ระบบการให้คะแนนอัตโนมัติ โดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม

นาย ศิวนันทน์ บุญประเสริฐ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2552

AUTOMATIC MARKING SYSTEM USING PROGRAM EDITING SEQUENCE

Mr. Siwanan Boonprasert

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2009

หัวข้อวิทยานิพนธ์	ระบบการให้คะแนนอัตโนมัติ โดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม
โดย	นายศิวนันทน์ บุญประเสริฐ
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล)

..... กรรมการ

(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการภายนอกมหาวิทยาลัย

(ดร.เนื่องวงศ์ ทวยเจริญ)

ศิวนันทน์ บุญประเสริฐ : ระบบการให้คะแนนอัตโนมัติโดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม (Automatic Marking System Using Program Editing Sequence) อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.สมชาย ประสิทธิ์จูตระกูล, 160 หน้า.

งานวิจัยนี้นำเสนอระบบอัตโนมัติในการตรวจให้คะแนน สำหรับการเรียนการสอนในชั้นเรียนปฏิบัติการการทำโปรแกรมคอมพิวเตอร์ โดยปรกติการตรวจให้คะแนนใช้ชุดคำสั่งทดสอบตรวจผลการดำเนินการของโปรแกรม เพื่อประเมินว่าโปรแกรมทำงานถูกต้องหรือไม่ สำหรับโปรแกรมที่ไม่ผ่านการทดสอบของชุดคำสั่ง ไม่สามารถประเมินให้คะแนนได้ ระบบควรนำรหัสต้นฉบับของโปรแกรมมาวิเคราะห์เพื่อให้คะแนนบางส่วน ระบบที่นำเสนอนี้นำลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมมาวิเคราะห์แทนที่จะใช้รหัสต้นฉบับท้ายสุด การบันทึกลำดับการเปลี่ยนแปลงของรหัสต้นฉบับระหว่างการเขียนโปรแกรมอาศัยโปรแกรมเสริมที่ได้พัฒนาขึ้นและติดตั้งให้กับโปรแกรมแก้ไขรหัสต้นฉบับ ลำดับการเปลี่ยนแปลงของรหัสต้นฉบับที่บันทึกได้จะถูกนำมาเปรียบเทียบความละม้ายกับชุดของเฉลยที่มี เพื่อหาคู่ของรหัสต้นฉบับกับชุดเฉลยที่ละม้ายที่สุด แล้วนำมาคำนวนให้คะแนน โดยอาศัยค่าความละม้ายและส่วนเบี่ยงเบนของค่าความซับซ้อนของโปรแกรม ผลการทดลองแสดงให้เห็นว่าระบบอัตโนมัติให้คะแนนที่ได้ผลลัพธ์ที่เหมาะสมกับโปรแกรมของผู้เรียนที่ไม่สามารถตรวจสอบได้โดยชุดคำสั่งทดสอบ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา 2552.....

4970605121 : MAJOR COMPUTER ENGINEERING

KEYWORDS : MARKING / GRADING / AUTOMATIC MARKING SYSTEM / AUTOMATICGRADING SYSTEM

SIWANAN BOONPRASERT : AUTOMATIC MARKING SYSTEM USING PROGRAM EDITING SEQUENCE. THESIS ADVISOR : ASSOC. PROF. SOMCHAI PRASITJUTRAKUL, Ph.D., 160 pp.

This thesis presents an automatic system for marking online computer programming sessions. Marking is normally done by performing a functional testing script on student programs. For any program which does not pass the testing script, its source code should be statically analyzed to give some partial credit. Rather than analyzing a final source code of the programming session, the system considers all editings of student's source code. A plugin module for source-code editor was developed for storing all source code editings during a programming session. The editing sequence are approximately pairwise-matched with a predefined set of solutions. Finally, marking is done based on the most similarly-matched pair of student's source code and solution. A combination of similarity and complexity distance metrices are used to mark the source code. Experimental results showed that the system automatically gives reasonable partial credits to functionally-failed programs.

Department : Computer Engineering Student's Signature .....

Field of Study : Computer Engineering Advisor's Signature .....

Academic Year : 2009

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของ รศ.ดร. สมชาย ประสิทธิ์จูตระกูล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้ให้คำแนะนำ ข้อคิดเห็น และแนวทางในการวิจัยมาด้วยดีตลอด

ขอขอบคุณพี่ ๆ น้อง ๆ และ เพื่อน ที่ช่วยเหลือให้คำแนะนำ และให้กำลังใจแก่ผู้วิจัยตลอดมา ท้ายนี้ผู้วิจัยใครขอกราบขอบพระคุณบิดา มารดา ซึ่งสนับสนุน ช่วยเหลือและให้กำลังใจแก่ผู้วิจัยเสมอมาจนสำเร็จการศึกษา

สารบัญ

หน้า

บทคัดย่อภาษาไทย..... ง

บทคัดย่อภาษาอังกฤษ..... จ

กิตติกรรมประกาศ..... ฉ

สารบัญ..... ช

สารบัญตาราง..... ญ

สารบัญภาพ..... ฏ

บทที่ 1 บทนำ.....

1.1 ความเป็นมาและความสำคัญของปัญหา..... 1

1.2 วัตถุประสงค์ของการวิจัย..... 3

1.3 ขอบเขตของการวิจัย..... 3

1.4 คำจำกัดความที่ใช้ในการวิจัย..... 4

1.5 ประโยชน์ที่คาดว่าจะได้รับ..... 4

1.6 วิธีดำเนินการวิจัย..... 5

1.7 ลำดับขั้นตอนในการเสนอผลการวิจัย..... 5

1.8 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์..... 6

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง..... 7

2.1 แนวคิดและทฤษฎี..... 7

2.1.1 ส่วนของการวิเคราะห์โปรแกรม..... 7

2.1.1.1 การวิเคราะห์เชิงสถิต..... 7

2.1.1.2 การวิเคราะห์เชิงพลวัต..... 8

2.1.2 ส่วนของการเปรียบเทียบความละม้าย..... 8

2.1.2.1 สายอักขระ (string)..... 8

2.1.2.2 โทเค็น (token)..... 9

2.1.2.3 ต้นไม้ (tree)..... 12

2.1.2.4 กราฟ (graph)..... 13

2.1.3 ส่วนของการวัดและประเมินผลคะแนน..... 14

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง..... 18

2.3 เครื่องมือที่ใช้ในการวิจัย..... 22

หน้า

2.3.1 โปรแกรม Eclipse IDE..... 22
2.3.2 โปรแกรม Plaggie..... 26
2.3.3 ไลบรารี ANTLR (Another Tool for Language Recognition)..... 29
บทที่ 3 วิธีดำเนินการวิจัย..... 32
3.1 ภาพรวมของระบบ..... 32
3.2 การพัฒนาโปรแกรมเสริม Javasnapshot เพื่อบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม..... 35
3.2.1 คุณสมบัติของโปรแกรมเสริม Javasnapshot..... 35
3.2.1.1 การกำหนดพฤติกรรมในการเก็บข้อมูล..... 36
3.2.1.2 การเริ่มต้นบันทึกและการหยุดบันทึกข้อมูล..... 37
3.2.2 จุดเชื่อมต่อของโปรแกรม Eclipse ที่ใช้งาน..... 37
3.2.3 คลาสของระบบในการบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม..... 39
3.3 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย..... 41
3.3.1 การทดสอบโปรแกรม Plaggie..... 42
3.3.2 คลาสของระบบในการเปรียบเทียบความละม้าย..... 44
3.4 การวิเคราะห์รหัสต้นฉบับของโปรแกรมกับชุดเฉลยเพื่อให้คะแนน..... 44
3.4.1 คลาสของระบบในการคำนวนให้คะแนน..... 45
3.4.2 สมการการคำนวนให้คะแนน..... 46
3.5 กลุ่มประชากร..... 47
3.6 การเก็บรวบรวมข้อมูล..... 47
3.7 การวิเคราะห์ข้อมูล..... 48
3.8 การประเมินผลคะแนน..... 50
บทที่ 4 ผลการวิเคราะห์ข้อมูล..... 51
4.1 การวิเคราะห์ข้อมูล..... 51
4.1.1 ผลการเก็บบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม..... 51
4.1.1.1 การบันทึกตามช่วงเวลา..... 51
4.1.1.2 การบันทึกรายตัวอักษร..... 53

หน้า

4.1.1.3 การบันทึกรายคำสั่ง..... 53
4.1.2 ผลการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย..... 54
4.1.3 ผลการเปรียบเทียบคะแนน..... 59
4.2 สรุปผลการวิเคราะห์..... 67
บทที่ 5 สรุปผลการวิจัย และข้อเสนอแนะ..... 69
5.1 สรุปผลการวิจัย..... 69
5.2 ข้อเสนอแนะ..... 70
รายการอ้างอิง..... 73
ภาพผนวก..... 75
ภาคผนวก ก..... 76
ภาคผนวก ข..... 100
ภาคผนวก ค..... 106
ภาคผนวก ง..... 111
ภาคผนวก จ..... 145
ประวัติผู้เขียนวิทยานิพนธ์..... 160

สารบัญตาราง

		หน้า
ตารางที่ 2.1	ตัวอย่างการทำงานของอัลกอริทึม Longest Common Subsequence…	9
ตารางที่ 2.2	มาตรวัดของฮอลสตีด……..	18
ตารางที่ 2.3	ประเภทของโทเค็นในโปรแกรม Plaggie…….	27
ตารางที่ 2.4	ผลการตัดแบ่งรหัสต้นฉบับของโปรแกรมเป็นโทเค็นด้วยโปรแกรม Plaggie.....	28
ตารางที่ 2.5	ประเภทของโทเค็นของภาษาจาวาที่ได้จากการตัดคำด้วยไลบรารี ANTLR……	29
ตารางที่ 3.1	คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot	39
ตารางที่ 3.2	คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.handlers……	39
ตารางที่ 3.3	คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.jobs……	40
ตารางที่ 3.4	คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.lexer……	40
ตารางที่ 3.5	คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.parser……	40
ตารางที่ 3.6	คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.preferences……..	40
ตารางที่ 3.7	ผลการทดสอบผลของการเปลี่ยนแปลงรหัสต้นฉบับของโปรแกรมที่มีผลต่อเนื้อหาแต่ไม่มีผลต่อแนวทางการแก้ไขปัญหา……	42
ตารางที่ 3.8	ผลการทดสอบการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมด้วย Plaggie……	43
ตารางที่ 3.9	คลาสของระบบในส่วนการเปรียบเทียบความละม้าย ใน package javasnapshot.analyze……..	44
ตารางที่ 3.10	คลาสของระบบในส่วนการเปรียบเทียบความละม้าย ใน package javasnapshot.analyze.engine……	44
ตารางที่ 3.11	คลาสของระบบในส่วนการคำนวนให้คะแนน ใน package	

javasnapshot.mark..... 45
ตารางที่ 3.12 คลาสของระบบในส่วนการคำนวนให้คะแนน ใน package javasnapshot.estimate..... 46
ตารางที่ 4.1 ผลการทดสอบของแบบทดสอบที่ 1 ของผู้ที่ทำไม่เสร็จภายในเวลา..... 60
ตารางที่ 4.2 ผลการทดสอบของแบบทดสอบที่ 2 ของผู้ที่ทำไม่เสร็จภายในเวลา..... 60
ตารางที่ 4.3 ผลการทดสอบของแบบทดสอบที่ 3 ของผู้ที่ทำไม่เสร็จภายในเวลา..... 61
ตารางที่ 4.4 ผลการทดสอบของแบบทดสอบที่ 4 ของผู้ที่ทำไม่เสร็จภายในเวลา..... 62
ตารางที่ 4.5 ผลการทดสอบของแบบทดสอบที่ 5 ของผู้ที่ทำไม่เสร็จภายในเวลา..... 62
ตารางที่ 4.6 สรุปผลการคำนวนเปอร์เซ็นต์ของคะแนนของผู้ที่ทำแบบทดสอบไม่สำเร็จ..... 63

สารบัญภาพ

		หน้า
ภาพที่ 2.1	ตัวอย่างการตัดแบ่งโทเค็นจากรหัสต้นฉบับของโปรแกรม.....	10
ภาพที่ 2.2	รหัสเทียมของอัลกอริทึม Greedy string tiling.....	11
ภาพที่ 2.3	สูตรการคำนวนค่าความละม้าย.....	12
ภาพที่ 2.4	ตัวอย่างการเปลี่ยนแปลงจากโครงสร้างต้นไม้ (a) ไปยังต้นไม้ (c)	12
ภาพที่ 2.5	ตัวอย่าง program dependency graph.....	13
ภาพที่ 2.6	การแปลงรหัสต้นฉบับของโปรแกรมให้อยู่ในโครงสร้างกราฟ.....	15
ภาพที่ 2.7	The “black box” model of the traditional oracle.....	20
ภาพที่ 2.8	แผนผังการทำงานของระบบอัตโนมัติในการตรวจการลอกกันของโปรแกรมคอมพิวเตอร์ของนักเรียน.....	21
ภาพที่ 2.9	ขั้นตอนการทำงานของระบบ Coogle.....	22
ภาพที่ 2.10	หน้าต่างการทำงานของโปรแกรม Eclipse IDE.....	23
ภาพที่ 2.11	สถาปัตยกรรมของโปรแกรมเสริมภายในโปรแกรม Eclipse.....	23
ภาพที่ 2.12	ตัวอย่างการประกาศการเชื่อมต่อโปรแกรมเสริมของโปรแกรม Eclipse.....	24
ภาพที่ 2.13	การตั้งค่ารายละเอียดต่าง ๆ ของโปรแกรมเสริมที่พัฒนาขึ้นด้วยโปรแกรม Eclipse.....	25
ภาพที่ 2.14	การกำหนดรายละเอียดการเชื่อมต่อของโปรแกรมเสริมที่พัฒนาขึ้น.....	25
ภาพที่ 2.15	ตัวอย่างรหัสต้นฉบับของโปรแกรมที่นำมาตัดโทเค็นด้วยโปรแกรม Plaggie.....	28
ภาพที่ 3.1	ตัวอย่างรหัสต้นฉบับของโปรแกรมคำนวนเส้นรอบวงและพื้นที่ ณ เวลา T วินาที.....	33
ภาพที่ 3.2	ตัวอย่างรหัสต้นฉบับของโปรแกรมคำนวนเส้นรอบวงและพื้นที่ ขณะเสร็จสมบูรณ์.....	33
ภาพที่ 3.3	แนวคิดในการทำงานของระบบการตรวจข้อสอบอัตโนมัติ โดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม.....	34
ภาพที่ 3.4	โปรแกรมเสริมที่ใช้เก็บข้อมูลที่ติดตั้งเพิ่มเติมลงในโปรแกรม Eclipse.....	35
ภาพที่ 3.5	ปุ่มฟังก์ชั่นการทำงานของโปรแกรมเสริม.....	35
ภาพที่ 3.6	หน้าต่างกำหนดค่าการทำงานของโปรแกรมเสริม.....	37
ภาพที่ 3.7	จุดเชื่อมต่อที่มีการใช้งานของโปรแกรมเสริม Javasnapshot ที่พัฒนาขึ้น.....	38

ภาพที่ 3.8 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย..... 41
ภาพที่ 3.9 การวิเคราะห์รหัสต้นฉบับของโปรแกรมกับชุดเฉลยเพื่อให้คะแนน..... 45
ภาพที่ 3.10 ลำดับการเปลี่ยนแปลงของโปรแกรมที่บันทึกได้ หลังจากเสร็จสิ้นการทดสอบ..... 48
ภาพที่ 3.11 การวิเคราะห์ความละม้ายของลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่บันทึกได้ กับชุดเฉลย..... 49
ภาพที่ 3.12 ผลการเปรียบเทียบความละม้ายได้รหัสต้นฉบับของโปรแกรม และชุดเฉลยที่ละม้ายกันมากที่สุด เพื่อนำไปคำนวนให้คะแนน..... 49
ภาพที่ 4.1 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 1..... 51
ภาพที่ 4.2 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 2..... 52
ภาพที่ 4.3 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 3..... 52
ภาพที่ 4.4 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 4..... 52
ภาพที่ 4.5 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 1..... 53
ภาพที่ 4.6 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 2..... 53
ภาพที่ 4.7 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 3..... 54
ภาพที่ 4.8 ส่วนของความละม้ายระหว่างรหัสต้นฉบับของโปรแกรมกับชุดเฉลย..... 55
ภาพที่ 4.9 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบคนที่ 1 กับชุดเฉลยทั้งหมดของแบบทดสอบที่ 1..... 55
ภาพที่ 4.10 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบคนที่ 10 กับชุดเฉลยของแบบทดสอบที่ 1..... 56
ภาพที่ 4.11 รหัสต้นฉบับของโปรแกรมลำดับที่ 35 ของผู้ทดสอบคนที่ 10 (แบบทดสอบที่ 1)..... 57
ภาพที่ 4.12 รหัสต้นฉบับของโปรแกรมลำดับที่ 78 ของผู้ทดสอบคนที่ 10 (แบบทดสอบที่ 1)..... 57
ภาพที่ 4.13 รูปแบบของความละม้ายที่สามารถเกิดขึ้น ในการเปรียบเทียบรหัสต้นฉบับของโปรแกรม กับชุดเฉลย..... 58
ภาพที่ 4.14 รหัสต้นฉบับของผู้ทดสอบคนที่ 6 ของแบบทดสอบที่ 1 ณ ลำดับที่ 11..... 65
ภาพที่ 4.15 รหัสต้นฉบับของชุดเฉลยชุดที่ 4 ของแบบทดสอบที่ 1..... 65
ภาพที่ 4.16 รหัสต้นฉบับของผู้ทดสอบคนที่ 10 ของแบบทดสอบที่ 4 ณ ลำดับที่ 11..... 66

ภาพที่ 4.17 รหัสต้นฉบับของชุดเฉลยชุดที่ 6 ของแบบทดสอบที่ 4..... 66

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในการตรวจให้คะแนนโปรแกรมที่ผู้เรียนพัฒนาขึ้นมาในระหว่างการทำปฏิบัติการ (lab) หรือระหว่างการสอบกลางภาคและปลายภาคนั้น งานที่ผู้เรียนส่งที่เป็นแบบอัตนัย ทำให้อาจารย์และผู้ช่วยสอนมีปริมาณงานที่จะต้องตรวจเป็นจำนวนมาก ต้องใช้เวลาค่อนข้างมากสำหรับการตรวจงาน และได้กลายเป็นข้อจำกัด ทำให้อาจารย์ไม่สามารถให้งานแก่นักเรียนมากเกินกว่ากำลังที่จะใช้ตรวจได้ เป็นข้อจำกัดที่สำคัญต่อการพัฒนาการเรียนการสอน ซึ่งเป็นส่วนสำคัญมากในการเรียนการสอนในวิชาการด้านการพัฒนาโปรแกรมของผู้เรียนในภาคปฏิบัติการพัฒนาโปรแกรม ถ้าเราสามารถนำคอมพิวเตอร์เข้ามาช่วยในการตรวจงานของนักเรียนในส่วนนี้ได้อย่างมีประสิทธิภาพน่าจะเป็นประโยชน์มากในการเรียนการสอนของวิชาการทำโปรแกรมคอมพิวเตอร์

ในปัจจุบันระบบการตรวจให้คะแนนอัตโนมัติ ได้เข้ามามีส่วนอย่างมากในการเรียนการสอน เพื่อช่วยอำนวยความสะดวกแก่อาจารย์และผู้ช่วยสอน ในการให้งานและตรวจงานภาคปฏิบัติในการสอนได้มากขึ้น การให้คะแนนของระบบตรวจให้คะแนนอัตโนมัติในปัจจุบันมีลักษณะการตรวจและให้คะแนนอยู่เป็น 2 แบบที่สำคัญ คือ

1. การให้คะแนนแบบดูผลลัพธ์ (black box) โดยการทดสอบการทำงานของโปรแกรมเทียบกับผลลัพธ์จากข้อมูลทดสอบ
2. การให้คะแนนแบบดูจากรหัสต้นฉบับของโปรแกรม (white box) โดยดูตามเงื่อนไขการตรวจ ตามที่ได้กำหนดไว้ เช่น ดูจากวิธีแก้ปัญหาว่าถูกต้องหรือไม่ จากความสวยงาม จากความคิดสร้างสรรค์ เป็นต้น

ปัจจุบันการตรวจให้คะแนนในงานภาคปฏิบัติการเขียนโปรแกรมในปัจจุบันของคณะวิศวกรรมศาสตร์[1] เป็นการให้คะแนนแบบดูผลลัพธ์ โดยมีระบบตรวจให้คะแนนโดยเปรียบเทียบผลของการทำงานของโปรแกรมกับผลเฉลย หรือกรณีทดสอบที่เตรียมไว้ ถ้าไม่ถูกต้องก็จะไม่ได้คะแนน ซึ่งถ้าหากดูจากรหัสต้นฉบับของโปรแกรม ก็อาจได้คะแนนไปบ้าง สำหรับกรณีที่เขียนโปรแกรมลงกระดาษคำตอบ แต่ถ้าดูที่รหัสต้นฉบับของโปรแกรม ณ เวลาสุดท้ายเพียงอย่างเดียวก็อาจไม่ดี ในกรณีที่ทำข้อสอบบนเครื่องคอมพิวเตอร์ สามารถที่จะเพิ่มความสามารถของระบบ ให้สามารถบันทึกการเปลี่ยนแปลงของโปรแกรมได้ ตั้งแต่เริ่มถึงสิ้นสุด ซึ่งในที่นี้ขอเรียกว่าลำดับการเปลี่ยนแปลงของโปรแกรม (program editing sequence) เช่น ในการพัฒนาโปรแกรมของ

นักเรียนขณะทำการพัฒนาโปรแกรมเมื่อเวลาผ่านไป อาจเริ่มต้นและพัฒนาไปในแนวทางที่ถูกต้อง ดังตัวอย่างเช่น

โจทย์ : จงเขียนเมท็อดรับตัวแปรแถวลำดับ 2 มิติ และหาผลรวมของตัวเลขทั้งหมดในแถวลำดับ 2 มิติ

```
public int sumArray2(int[][] data){
  int sum = 0;

  return sum;
}
```

งานของนาย ก เมื่อเวลา T1

```
public int sumArray2(int[][] data){
  int sum = 0;
  for(int i=0 ; i< data.length ; i++){
    for(int j=0 ; j< data [] ; j++){
      sum = sum + data [];
    }
  }
  return sum;
}
```

งานของนาย ก เมื่อเวลา T2

เมื่อเวลาผ่านไปอาจมีการเปลี่ยนแปลงไปในแนวทางอื่น เมื่อนาย ก ติดปัญหา เช่นไม่รู้ว่าจะวนแถวลำดับในมิติที่ 2 อย่างไร จึงลองลบออกและเปลี่ยนวิธีการเขียน จนกระทั่งหมดเวลา

```
public int sumArray2(int[][] data){
  int sum = 0;
  for(int i=0 ; i< data.length ; i++){

  }
  return sum;
}
```

งานของนาย ก เมื่อเวลา T3

```
public int sumArray2(int[][] data){
  int sum = 0;
  for(int i=0 ; i< data.length ; i++){
    sum = sum + data [i];
  }
  return sum;
}
```

งานของนาย ก เมื่อเวลา T4 (หมดเวลา)

```
public int sumArray2(int[][] data){
  int sum = 0;
  for(int i=0 ; i< data.length ; i++){
    for(int j=0 ; j< data [i].length ; j++){
      sum = sum + data [i][j];
    }
  }
  return sum;
}
```

รหัสต้นฉบับของเฉลย

จะเห็นได้ว่าถ้าเปรียบเทียบเมื่อเวลา T4 กับเฉลย ก็จะรู้ได้ชัดว่าผู้เรียนคนนี้อาจไม่รู้จักแถวลำดับ 2 มิติ แต่ถ้าสามารถย้อนกลับไปดูได้ จะเห็นว่าเมื่อเวลา T2 ผู้เรียนรู้จักแถวลำดับ 2 มิติ แต่ไม่รู้ว่าจะเข้าถึงในมิติที่ 2 ได้อย่างไร จึงเห็นได้ว่าถ้าดูจากผลลัพธ์สุดท้ายเพียงอย่างเดียว อาจทำให้ประเมินความสามารถของนักเรียนได้ไม่ดีเท่าที่ควร

ในการพัฒนาระบบการให้คะแนนโดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม จะเห็นได้ว่าในงาน ๆ หนึ่งนักเรียนอาจมีการเปลี่ยนแปลงตัวรหัสต้นฉบับของโปรแกรมที่พัฒนาขึ้นได้ตลอดเวลา ทั้งเพิ่ม ลบ แก้ไข หรือเปลี่ยนแปลงแนวทางของโปรแกรม ย่อมทำให้ได้ข้อมูลที่ใช้ในการเปรียบเทียบ และใช้ในการตรวจมีปริมาณมากขึ้น ซึ่งโปรแกรมจะเข้ามาช่วยในการตรวจเพื่อลดภาระงานที่เกิดขึ้น และให้คุณภาพในการตรวจที่ดีกว่า สามารถประเมินความรู้ความเข้าใจของผู้เรียนได้ดีขึ้นกว่าการตรวจโดยดูผลลัพธ์สุดท้าย ซึ่งคือจุดประสงค์ของงานวิจัยครั้งนี้

1.2 วัตถุประสงค์ของการวิจัย

เพื่อศึกษาการให้คะแนนโดยดูตามลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่ทำให้ได้คุณภาพในการตรวจและประเมินผลงานของผู้เรียนได้เทียบเท่าหรือดีกว่าวิธีการตรวจโดยดูผลลัพธ์สุดท้าย และนำมาพัฒนาเป็นระบบการให้คะแนนอัตโนมัติ เพื่อลดภาระงานของอาจารย์และผู้สอน

1.3 ขอบเขตของการวิจัย

1. ข้อมูลที่นำมาศึกษาจะนำแบบทดสอบจากการเรียนการสอนในวิชาการทำโปรแกรมคอมพิวเตอร์ (2110101) ของนิสิตคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2552
2. การเก็บข้อมูลระหว่างการทำปฏิบัติการจะเก็บข้อมูลการทำงานผ่านระบบ JLab หรือสร้างข้อมูลจำลอง หรือเก็บข้อมูลจากกลุ่มทดสอบ
3. ระบบการให้คะแนนจะทำการศึกษาค้นคว้าบนพื้นฐานของระบบการให้คะแนนที่ใช้อยู่ในปัจจุบันมาเป็นเกณฑ์ประกอบในการให้คะแนนแก่ผู้เรียน
4. การตรวจให้คะแนนจะจำกัดเฉพาะในส่วนของการตรวจรหัสต้นฉบับของโปรแกรมเฉพาะความถูกต้องของโปรแกรม และให้ตามความพยายามที่ทำ สำหรับกรณีที่ทำไม่สำเร็จ จะไม่รวมการให้คะแนนในส่วนของรูปแบบของรหัสต้นฉบับของโปรแกรม ความคิดสร้างสรรค์ หรือการพิจารณาให้คะแนนในด้านอื่น ๆ

5. การประเมินผลของคะแนนที่ระบบคำนวณให้ จะให้อาจารย์ผู้สอนเป็นผู้ประเมินการยอมรับได้ของการให้คะแนนที่ได้ หรือนำคะแนนที่ได้มาเทียบกับการให้คะแนนของอาจารย์ผู้ทำการตรวจ ซึ่งจะต้องไม่แตกต่างกันอย่างมีนัยสำคัญ

1.4 คำจำกัดความที่ใช้ในการวิจัย

ระบบการให้คะแนนอัตโนมัติ หมายถึง ระบบที่พัฒนาเพื่อใช้ในการตรวจผลงานภาคปฏิบัติ (Lab) ในวิชาการทำโปรแกรมคอมพิวเตอร์ สำหรับนิสิตปีที่ 1 คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

รหัสต้นฉบับของโปรแกรม หมายถึง ข้อมูลรหัสต้นฉบับของโปรแกรมภาษาจาวา ที่ผู้เรียนเขียนขึ้นระหว่างทำปฏิบัติการ เพื่อพัฒนาโปรแกรมตามที่โจทย์กำหนด

การเปลี่ยนแปลงของโปรแกรม หมายถึง การเปลี่ยนแปลงที่เกิดขึ้นกับรหัสต้นฉบับของโปรแกรมที่ผู้เรียนพัฒนาขึ้น โดยจะเกิดการเปลี่ยนแปลงจากการเขียนเพิ่ม แก้ไข หรือลบออก

ลำดับการเปลี่ยนแปลงของโปรแกรม หมายถึง ลำดับของข้อมูลรหัสต้นฉบับของโปรแกรมที่ผู้เรียนพัฒนาขณะทำปฏิบัติการ (Lab) ที่จะถูกเก็บข้อมูลไว้เมื่อเกิดการเปลี่ยนแปลงของโปรแกรมขึ้น ตั้งแต่เริ่มต้นพัฒนาจนสิ้นสุดการทำปฏิบัติการ เพื่อนำมาใช้การศึกษาวิจัยและประเมินผลผู้เรียน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่ได้จากการศึกษาพัฒนาระบบการช่วยให้คะแนนแก่ผู้เรียนในการเรียนการสอนวิชาการทำโปรแกรมคอมพิวเตอร์ (2110101) จะช่วยให้สามารถพัฒนาคุณภาพการตรวจของอาจารย์ได้ดีกว่าการตรวจเฉพาะผลลัพธ์สุดท้ายเพียงอย่างเดียว และช่วยให้อาจารย์ประหยัดเวลาในการตรวจข้อสอบของผู้เรียน ทำให้อาจารย์มีเวลาในการเตรียมการสอน และสามารถให้งานแก่ผู้เรียนได้มากขึ้นโดยไม่กระทบต่อกำลังงานของอาจารย์ในการที่จะต้องมาตรวจข้อสอบหรืองานที่ได้มอบหมายให้ผู้เรียนทำ และนำเสนอผลจากการศึกษาทดลองหารูปแบบในการเก็บข้อมูลการเปลี่ยนแปลงของการพัฒนาโปรแกรม รวมถึงวิธีที่เหมาะสมในการที่จะนำมาใช้ในการเปรียบเทียบข้อมูล และนำเสนอแนวคิดและผลการศึกษาการให้คะแนนแบบเปรียบเทียบกับผู้อื่นเพื่อเป็นแนวทางในการพัฒนาระบบการให้คะแนนแบบอัตโนมัติที่จะปรับปรุงพัฒนาต่อไปสำหรับการเรียน

การสอนวิชาการทำโปรแกรมคอมพิวเตอร์ (2110101) หรือนำไปประยุกต์กับการพัฒนาในภาษาอื่น ๆ ต่อไป

1.6 วิธีดำเนินการวิจัย

1. ศึกษาแนวคิดและทฤษฎีที่เกี่ยวข้อง
2. ศึกษางานวิจัยที่เกี่ยวข้อง
3. ศึกษารูปแบบการให้คะแนนที่ใช้อยู่ในปัจจุบัน และค้นหารูปแบบการให้คะแนนจากข้อมูลการตรวจข้อสอบที่มีเก็บไว้
4. ค้นคว้าหารูปแบบที่เหมาะสมในการเก็บข้อมูลการเปลี่ยนแปลงของโปรแกรม
5. ค้นคว้าหาวิธีการที่เหมาะสมในการเปรียบเทียบข้อมูลการเปลี่ยนแปลงของโปรแกรม
6. พัฒนาระบบต้นแบบเพื่อทดสอบรูปแบบและวิธีการที่ได้ค้นคว้ามา
7. วิเคราะห์ผลที่ได้จากการพัฒนาและทดสอบระบบต้นแบบ
8. พัฒนาและปรับปรุงระบบ
9. วิเคราะห์และสรุปผลการทำงาน
10. จัดทำรายงานวิทยานิพนธ์

1.7 ลำดับขั้นตอนในการเสนอผลการวิจัย

การนำเสนอผลการวิจัยครั้งนี้ เสนอผลตามลำดับดังต่อไปนี้

บทที่ 1 บทนำ จะนำเสนอความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ ขอบเขตของการวิจัย คำจำกัดความที่ใช้ในการวิจัย ประโยชน์ที่คาดว่าจะได้รับ วิธีการดำเนินการวิจัย และผลงานที่ตีพิมพ์จากวิทยานิพนธ์

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง จะนำเสนอแนวคิดและทฤษฎีที่เกี่ยวข้อง และเอกสารงานวิจัยในปัจจุบันที่เกี่ยวข้องกับงานวิจัยนี้

บทที่ 3 วิธีดำเนินการวิจัย จะนำเสนอรายละเอียดของขั้นตอนในการวิจัยแต่ละขั้นที่ได้ทำตามที่ออกแบบไว้

บทที่ 4 ผลการวิเคราะห์ข้อมูล จะนำเสนอผลการวิเคราะห์ข้อมูลที่ได้จากการดำเนินการวิจัยตามวิธีดำเนินการวิจัยที่ได้ออกแบบไว้

บทที่ 5 สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ จะนำเสนอผลสรุปของการศึกษาวิจัย และอภิปรายผล และนำเสนอข้อเสนอแนะหลังจากที่ได้ทำการวิจัยเสร็จสิ้น

1.8 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการและนำเสนอในงานประชุมวิชาการ ในหัวข้อ “Automatic Marking System for Formative Assessment” โดย ศิวนันทน์ บุญประเสริฐ และ มานิต บุญประเสริฐ ในงานประชุมวิชาการ “13th UNESCO-APEID International Conference on Education and World Bank-KERIS High Level Seminar on ICT in Education” จัดโดย UNESCO Asia-Pacific Regional Bureau for Education ร่วมกับ National Commission of the People’s Republic of China for UNESCO และ The World Bank และ Korea Educational Research and Information Service(KERIS) ณ โรงแรม Zhejiang International เมืองหางโจว สาธารณรัฐประชาชนจีน ในระหว่างวันที่ 15-17 พฤศจิกายน 2552

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การวิจัยเรื่องระบบการให้คะแนนอัตโนมัติ โดยการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรมนี้ ผู้วิจัยได้ศึกษาแนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง โดยจะนำเสนอเนื้อหาออกเป็น 3 ส่วน คือ แนวคิดและทฤษฎี เอกสารและงานวิจัยที่เกี่ยวข้อง และเครื่องมือที่เกี่ยวข้องในงานวิจัย ดังนี้

2.1 แนวคิดและทฤษฎี

แนวคิดของระบบการให้คะแนนอัตโนมัติ โดยการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม จะเก็บข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมระหว่างที่ผู้เรียนทำปฏิบัติการ และนำชุดรหัสต้นฉบับของโปรแกรมที่บันทึกได้มาทำการเปรียบเทียบความละม้ายกับชุดเฉลยตามลำดับกันไปจนครบทุกลำดับที่เก็บไว้และครบทุกชุดเฉลยที่มี ซึ่งจะได้คู่ของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายที่สุดออกมา และนำผลที่ได้มาประเมินค่าความพยายามที่ใช้ในการพัฒนารหัสต้นฉบับของโปรแกรมที่เลือกได้และชุดเฉลยด้วยมาตรวัดทางวิศวกรรมซอฟต์แวร์ และนำมาคำนวนให้คะแนน โดยจะมีทฤษฎีที่เกี่ยวข้องในแต่ละส่วนดังนี้

2.1.1 ส่วนของการวิเคราะห์รหัสต้นฉบับของโปรแกรม

การวิเคราะห์รหัสต้นฉบับของโปรแกรม จะแบ่งแนวทางในการวิเคราะห์ออกเป็น 2 แบบ คือ การวิเคราะห์เชิงสถิต และการวิเคราะห์เชิงพลวัต โดยมีรายละเอียดดังนี้

2.1.1.1 การวิเคราะห์เชิงสถิต (Static analysis)

การวิเคราะห์เชิงสถิตเป็นการวิเคราะห์เพื่อศึกษาลักษณะของสิ่งที่สนใจจากรหัสต้นฉบับของโปรแกรมคอมพิวเตอร์ แบ่งแนวทางการวิเคราะห์ลักษณะออกเป็น 3 รูปแบบ คือ

- การวิเคราะห์เชิงข้อความ (String based analysis)
- การวิเคราะห์เชิงโครงสร้าง / วากยสัมพันธ์ (Structure/syntax based analysis)
- การวิเคราะห์เชิงความหมาย (Semantic based analysis)

2.1.1.2 การวิเคราะห์เชิงพลวัต (Dynamic analysis)

การวิเคราะห์เชิงพลวัตเป็นการวิเคราะห์เพื่อศึกษาลักษณะของสิ่งที่สนใจในขณะที่โปรแกรมกำลังดำเนินงาน

2.1.2 ส่วนของการเปรียบเทียบความละม้าย

การเปรียบเทียบความละม้ายของโปรแกรมสามารถนำวิธีที่ใช้ในการตรวจสอบการลอกโปรแกรมกัน (Plagiarism detection) มาใช้ในการเปรียบเทียบ ซึ่งในปัจจุบันสามารถจำแนกออกเป็น 4 รูปแบบ ประกอบด้วย สายอักขระ (String) โทเค็น (Token) ต้นไม้ (Tree) และกราฟ (Graph) มีรายละเอียดดังนี้

2.1.2.1 สายอักขระ (string)

วิธีนี้เป็นการนำรหัสต้นฉบับของโปรแกรมมาตัดหมายเหตุ (Comment) ช่องว่าง (Blank space) และส่วนที่ไม่ต้องการนำเข้ามาเปรียบเทียบ จะได้สายอักขระที่ใช้เปรียบเทียบ และนำมาเปรียบเทียบเพื่อหาลำดับของสายอักขระที่ตรงกัน โดยอัลกอริทึมที่ใช้ในการเปรียบเทียบ คือ

- อัลกอริทึมสำหรับการหา Longest Common Subsequence (LCS)

อัลกอริทึมนี้เป็นการหาลำดับย่อยร่วมกันที่ยาวที่สุด โดยลำดับย่อย คือ ลำดับของอักขระที่เป็นสมาชิกของสายอักขระที่พิจารณา ซึ่งไม่จำเป็นที่จะต้องอยู่ติดกัน

บทนิยาม : กำหนดให้ $X = (x_1, x_{2, \dots,} x_i)$ และ $Y = (y_1, y_2, \dots, y_j)$

$$\mathrm{LCS}(X_{1\dots i}, Y_{1\dots j}) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(X_{1\dots i-1}, Y_{1\dots j-1}) + x_i & \text{if } x_i = y_j \\ \max(\mathrm{LCS}(X_{1\dots i}, Y_{1\dots j-1}), \mathrm{LCS}(X_{1\dots i-1}, Y_{1\dots j})) & \text{otherwise} \end{cases}$$

ในการทำงานของอัลกอริทึมนี้ เราจะใช้แถวลำดับ 2 มิติ มาช่วยในการหาคำตอบ โดยจะเปรียบเทียบไปจากซ้ายไปขวา และจากบนลงล่าง โดย L(i , j) จะหาได้จาก L(i-1 , j-1) L(i , j-1) และ L(i-1 , j) ตามบทนิยามที่กำหนดไว้ไปจนสิ้นสุด ซึ่งจากตัวอย่างการทำงานของอัลกอริทึมในตารางที่ 2.1 จะได้ “AJBU” เป็นลำดับย่อยร่วมกันที่ยาวที่สุดของ “AFJBXYU” กับ “MAJBDMU”

ตารางที่ 2.1 ตัวอย่างการทำงานของอัลกอริทึม Longest Common Subsequence

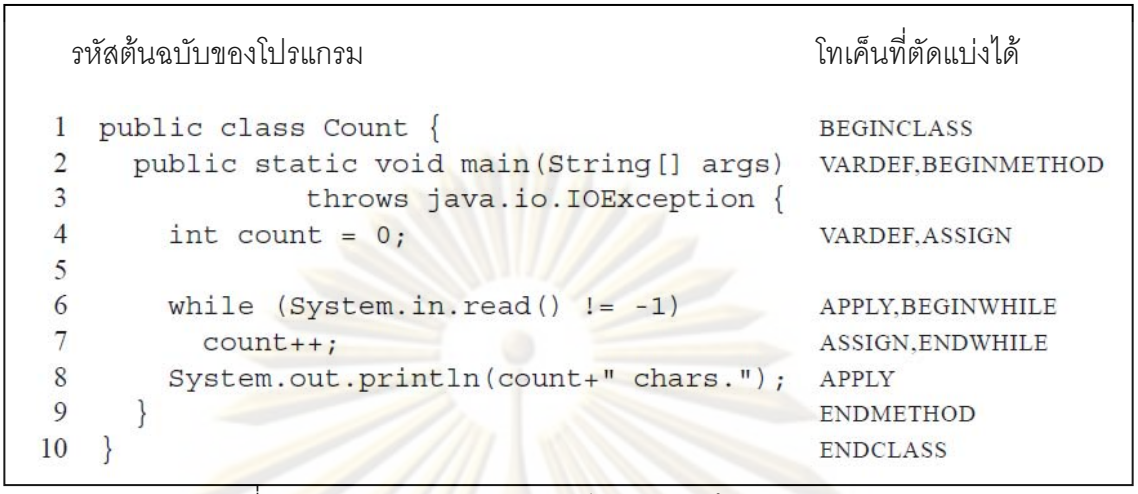
		0	1	2	3	4	5	6	7
			A	F	J	B	X	Y	U
0									
1	M								
2	A		A	A	A	A	A	A	A
3	J		A	A	AJ	AJ	AJ	AJ	AJ
4	B		A	A	AJ	AJB	AJB	AJB	AJB
5	D		A	A	AJ	AJB	AJB	AJB	AJB
6	M		A	A	AJ	AJB	AJB	AJB	AJB
7	U		A	A	AJ	AJB	AJB	AJB	AJBU

2.1.2.2 โทเค็น (token)

วิธีนี้เป็นการนำรหัสต้นฉบับของโปรแกรมมาสร้างเป็นโทเค็น (token) ซึ่งจะตัดแบ่งตาม คำหลัก (Keyword) สัญลักษณ์ (Symbol) ตัวแปร (Identifier) ซึ่งการทำโทเค็นจะเป็นการตัดสายอักขระออกเป็นคำย่อย ๆ เรียงกันตามลำดับ แล้วจึงนำลำดับคำย่อย ๆ นั้นมาเปรียบเทียบกัน นอกจากนี้อาจนำรูปแบบของลำดับของโทเค็นที่เก็บไว้เป็นต้นแบบหรือลายพิมพ์ (fingerprint) เข้ามาช่วยในการเปรียบเทียบ เพื่อให้สามารถเปรียบเทียบได้เร็วขึ้น โดยอัลกอริทึมที่ใช้ในการเปรียบเทียบหาความละม้าย คือ

- อัลกอริทึม Running Karp-Rabin Matching and Greedy string tiling (RKR-GST) [2]

อัลกอริทึมนี้เป็นการเปรียบเทียบระหว่างโทเค็นจากรหัสต้นฉบับของโปรแกรม 2 ชุด เพื่อหาจำนวนโทเค็นที่ละม้ายกัน โดยโทเค็นที่ได้จากการตัดแบ่งรหัสต้นฉบับของโปรแกรมออกเป็นคำย่อย ๆ แสดงดังตัวอย่างในภาพที่ 2.1 ดังนี้



	รหัสต้นฉบับของโปรแกรม	โทเค็นที่ตัดแบ่งได้
1	public class Count {	BEGINCLASS
2	public static void main(String[] args)	VARDEF,BEGINMETHOD
3	throws java.io.IOException {	
4	int count = 0;	VARDEF,ASSIGN
5		
6	while (System.in.read() != -1)	APPLY,BEGINWHILE
7	count++;	ASSIGN,ENDWHILE
8	System.out.println(count+" chars.");	APPLY
9	}	ENDMETHOD
10	}	ENDCLASS

ภาพที่ 2.1 ตัวอย่างการตัดแบ่งโทเค็นจากรหัสต้นฉบับของโปรแกรม

จากโทเค็นที่ได้จากคู่ของรหัสต้นฉบับของโปรแกรมที่จะทำการเปรียบเทียบความละม้าย โดยกำหนดให้รหัสต้นฉบับของโปรแกรมแรกแทนด้วย A และรหัสต้นฉบับของโปรแกรมหลังแทนด้วย B จะนำมาเปรียบเทียบโดยมีข้อกำหนดในการเปรียบเทียบ คือ

- โทเค็นใด ๆ ใน A สามารถจับคู่กับโทเค็นใน B ได้เพียง 1 ครั้งเท่านั้น ซึ่งผลของข้อกำหนดนี้ทำให้หลังจากที่ถูกจับคู่ไปแล้ว จะไม่มีการใช้โทเค็นนี้ของ A ไปจับคู่ซ้ำอีกกับโทเค็นใน B
- โทเค็นใด ๆ ใน A สามารถใช้จับคู่ได้ในทุก ๆ ตำแหน่งในสายโทเค็นของ B ซึ่งผลของข้อกำหนดนี้ทำให้การสลับเปลี่ยนตำแหน่งหรือเปลี่ยนบรรทัดของคำสั่งในรหัสต้นฉบับของโปรแกรมไม่มีผลต่อการเปรียบเทียบความละม้าย
- ถ้าพบสายโทเค็นที่สั้นและยาวที่มีส่วนร่วมกัน สายโทเค็นที่ยาวกว่าควรจะถูกเลือก

ในการทำการเปรียบเทียบโทเค็นของอัลกอริทึมนี้ จะแบ่งออกเป็น 2 ขั้นตอน คือ ขั้นตอนแรกทำการค้นหาโทเค็นร่วมที่ยาวที่สุดระหว่าง A และ B และขั้นตอนที่สองทำเครื่องหมาย (Mark) ให้กับโทเค็นร่วมที่พบในขั้นตอนแรกเพื่อไม่ให้ถูกนำไปใช้ในการค้นหาอีกซึ่งเป็นไปตามข้อกำหนด จากนั้นทำวนซ้ำทั้ง 2 ขั้นตอนไปเรื่อย ๆ จนกระทั่งเปรียบเทียบแล้วไม่พบส่วนที่เหมือนกันอีกจึงหยุดการทำงาน จากขั้นตอนการทำงานนี้แสดงเป็นรหัสเทียม (Pseudo code) [3] ได้ดังภาพที่ 2.2 ดังนี้

```
0   Greedy-String-Tiling(String A, String B) {
1       tiles = {};
2       do {
3           maxmatch = MinimumMatchLength;
4           matches = {};
5           Forall unmarked tokens A_a in A {
6               Forall unmarked tokens B_b in B {
7                   j = 0;
8                   while (A_{a+j} == B_{b+j} &&
9                           unmarked(A_{a+j}) && unmarked(B_{b+j}))
10                      j++;
11                  if (j == maxmatch)
12                      matches = matches ⊕ match(a, b, j);
13                  else if (j > maxmatch) {
14                      matches = {match(a, b, j)};
15                      maxmatch = j;
16                  }
17              }
18          }
19          Forall match(a, b, maxmatch) ∈ matches {
20              For j = 0 ... (maxmatch − 1) {
21                  mark(A_{a+j});
22                  mark(B_{b+j});
23              }
24              tiles = tiles ∪ match(a, b, maxmatch);
25          }
26      } while (maxmatch > MinimumMatchLength);
27      return tiles;
28  }
```

ภาพที่ 2.2 รหัสเทียมของอัลกอริทึม Greedy string tiling

จากอัลกอริทึมของ Greedy string tiling ในภาพที่ 2.2 จะถูกปรับปรุงประสิทธิภาพของอัลกอริทึมด้วยแนวความคิดของ Karp-Rabin โดยจะนำโทเค็นมาทำแฮช (Hash) เพื่อช่วยในการเปรียบเทียบ โดยจะทำแฮชทั้งในส่วนของโทเค็นของ A และโทเค็นของ B และนำแฮชที่ได้มาใส่ลงในตารางแฮช (Hash table) ถ้าแฮชของโทเค็นของ A และของ B ชุดไหนใส่ลงในตารางแฮชแล้วตรงกัน ก็จะนำโทเค็นชุดนั้นมาเปรียบเทียบต่อเพื่อดูว่าถ้าขยายสายโทเค็นต่อออกไปแล้วจะยังคงเหมือนกันอยู่หรือไม่ ทำให้ประสิทธิภาพของอัลกอริทึมในการเปรียบเทียบเป็นลักษณะเชิงเส้น (linear) หรือมี Big O เป็น $O(n)$ โดยกรณีที่แย่ที่สุด ประสิทธิภาพ

ของอัลกอริทึมจะยังคงเป็น $O(n^3)$ โดยค่าความละม้ายที่ได้หลังจากเปรียบเทียบจับคู่โทเค็นระหว่าง A และ B แล้ว จะคำนวนจากสูตรดังภาพที่ 2.3 ดังนี้

$$SIM(A,B) = \frac{2 \cdot coverage\,(tiles)}{|A| + |B|}$$

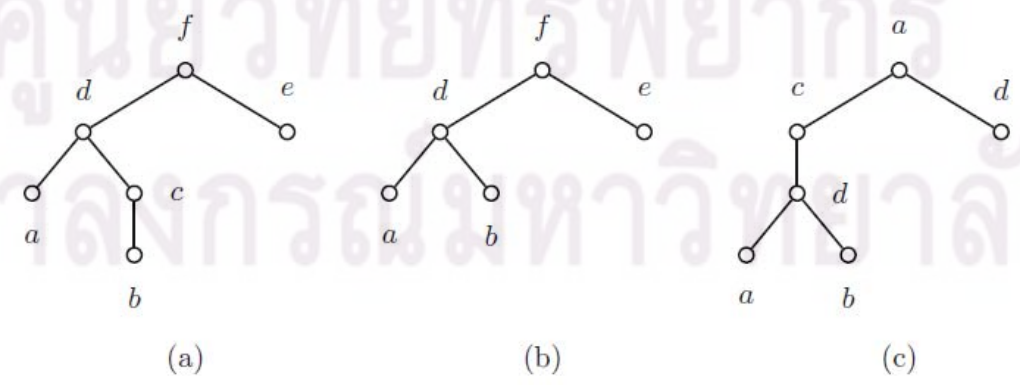
$$coverage\,(tiles) = \sum_{match(a,\,b,\,length)\ \in\ tiles} length$$

ภาพที่ 2.3 สูตรการคำนวนค่าความละม้าย

จากภาพที่ 2.3 ค่าความละม้ายระหว่าง A กับ B หาได้จากจำนวนโทเค็นที่พบร่วมกันทั้งหมดระหว่าง A กับ B คูณด้วย 2 และ หารด้วยผลรวมของจำนวนโทเค็นของ A และ B

2.1.2.3 ต้นไม้ (Tree)

วิธีนี้เป็นการนำรหัสต้นฉบับโปรแกรมมาแปลงให้อยู่ในรูปแบบของโครงสร้างต้นไม้วากยสัมพันธ์แบบนามธรรม (Abstract syntax tree) แล้วจึงนำมาเปรียบเทียบความละม้ายระหว่างโครงสร้างต้นไม้ที่ได้ โดยอัลกอริทึมที่ใช้ในการเปรียบเทียบ คือ อัลกอริทึม Tree edit distance อัลกอริทึมนี้หาว่า จะต้องใช้การเปลี่ยนแปลงกี่ขั้นตอนในการเปลี่ยนโครงสร้างต้นไม้หนึ่งไปให้เหมือนกับอีกโครงสร้างต้นไม้หนึ่ง โดยมีการเปลี่ยนแปลงที่เป็นไปได้คือ การเพิ่ม (Insertion) การแทนที่ (Substitution) และการลบ (Deletion) และในแต่ละประเภทของการเปลี่ยนแปลงจะมีการกำหนดต้นทุนของการเปลี่ยนแปลง (Cost function) โดยผลลัพธ์ที่ได้จะเป็นลำดับขั้นตอนที่ใช้ในการเปลี่ยนแปลง (Edit script) ตัวอย่างการเปลี่ยนแปลงแสดงดังภาพที่ 2.4



ภาพที่ 2.4 ตัวอย่างการเปลี่ยนแปลงจากโครงสร้างต้นไม้ (a) ไปยังต้นไม้ (c)

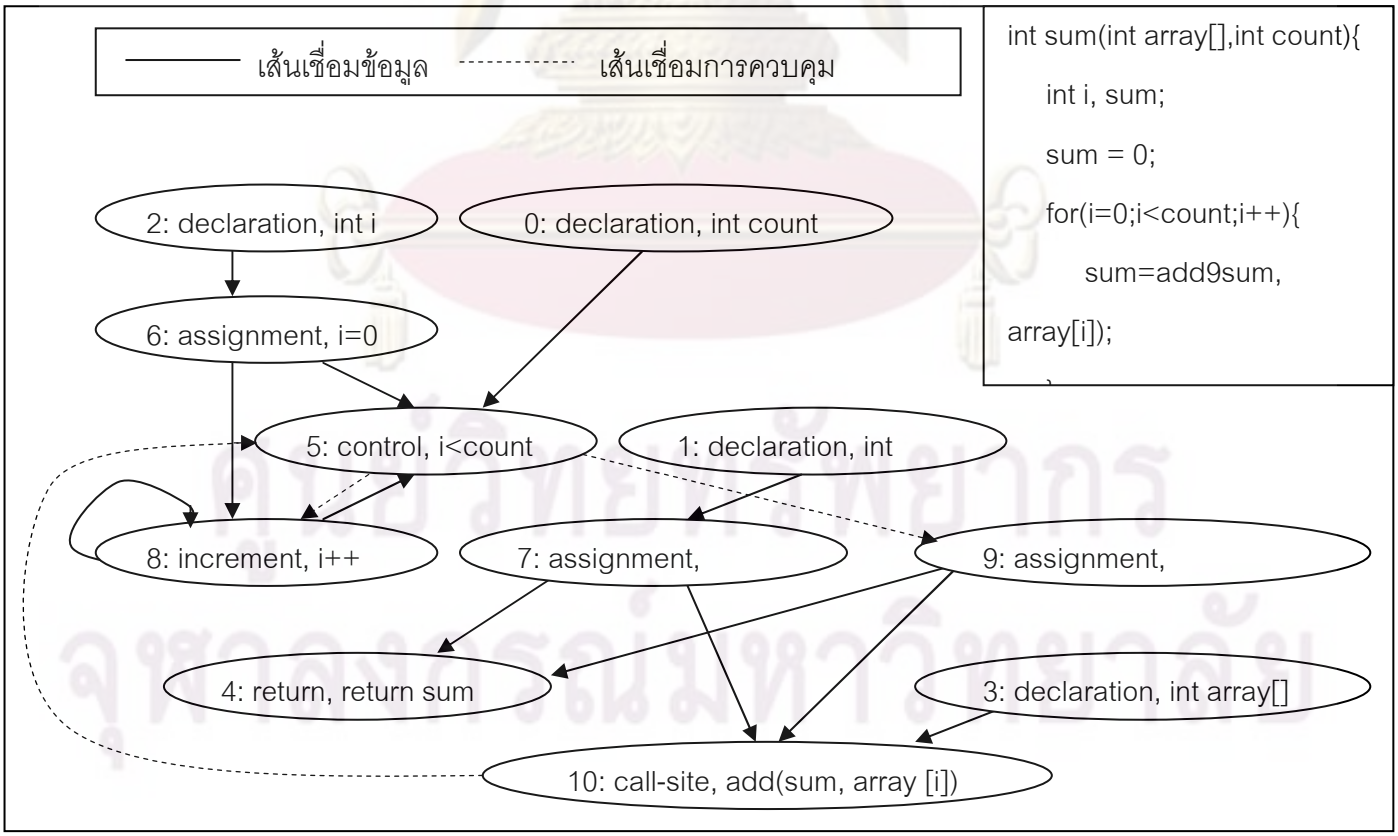
2.1.2.4 กราฟ (Graph)

วิธีนี้เป็นการนำรหัสต้นฉบับโปรแกรมมาแปลงให้อยู่ในรูปแบบของกราฟ (Graph) ตามลักษณะที่สนใจ แล้วจึงนำมาเปรียบเทียบความละม้ายระหว่างกราฟที่ได้ เช่น

- Program dependency graph (PDG)

  เป็นการใช้กราฟมาแทนโครงสร้างของโปรแกรมในรูปแบบที่เรียกว่า Program dependency graph ซึ่งจะมีรายละเอียดของกราฟดังนี้

  - จุดยอดของกราฟ (Vertex) แทนการเรียกใช้เมท็อด การประกาศค่าของตัวแปร และการกำหนดค่าของตัวแปร
  - เส้นเชื่อมข้อมูล (Data dependency edge) แทนการเชื่อมระหว่างจุดของกราฟที่แทนการทำงานที่อ้างอิงถึงตัวแปร
  - เส้นเชื่อมการควบคุม (Control dependency edge) แทนการเชื่อมระหว่างจุดของกราฟที่แทนการทำงานที่มีเงื่อนไข เช่น while หรือ for เป็นต้น เชื่อมต่อไปยังจุดที่การทำงานในเงื่อนไขนั้นเป็นจริง



ภาพที่ 2.5 ตัวอย่าง program dependency graph

2.1.3 ส่วนของการวัดและประเมินผลคะแนน

ในการประเมินผลการให้คะแนนมีการประเมินผลอยู่ 2 ลักษณะ [4] คือ การประเมินผลระหว่างเรียน (Formative) ซึ่งจะเน้นการนำผลไปใช้ในการพัฒนาผู้เรียนในระหว่างเรียน และการประเมินผลแบบหลังเรียน (Summative) ซึ่งจะเป็นการประเมินผู้เรียนว่ามีระดับความรู้ความเข้าใจต่อบทเรียนแค่ไหน การวัดและให้คะแนนแบบหลังเรียนนี้จะมีลักษณะการวัดตามเกณฑ์การประเมินผลที่ตั้งไว้ เช่น ลักษณะการออกแบบโปรแกรม การดำเนินการของโปรแกรม การตอบความต้องการของโจทย์ รูปแบบการเขียน และคำอธิบาย เป็นต้น

ในส่วนของการวัดและประเมินจากรหัสต้นฉบับของโปรแกรม นอกจากการตรวจวัดผลลัพธ์การดำเนินการของโปรแกรมแล้ว การนำมาตรวัดในการคำนวนความซับซ้อนของโปรแกรมมาใช้ประเมินรหัสต้นฉบับเป็นอีกวิธีหนึ่งที่สามารถนำมาใช้ประเมินเป็นผลคะแนนได้ โดยมีมาตรวัดความซับซ้อนของโปรแกรมที่น่าสนใจ [5] เช่น

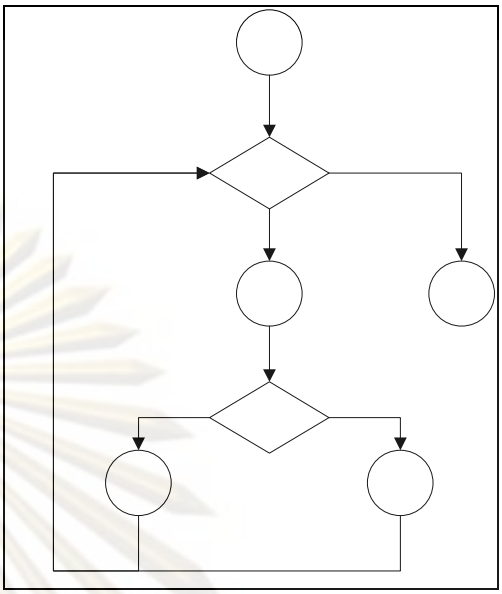
- มาตรวัดจำนวนบรรทัด (LOC)

มาตรวัดจำนวนบรรทัด เป็นการนับจำนวนบรรทัดที่ใช้ของรหัสต้นฉบับของโปรแกรม มาตรวัดนี้เป็นมาตรวัดเบื้องต้นที่บอกว่าโปรแกรมมีความซับซ้อนน้อยหรือมาก ถ้าจำนวนบรรทัดที่ใช้น้อย รหัสต้นฉบับของโปรแกรมมีแนวโน้มที่จะมีความซับซ้อนน้อย ถ้าจำนวนบรรทัดมากจะมีแนวโน้มที่มีความซับซ้อนมาก

- มาตรวัดความซับซ้อนของ McCabe's Cyclomatic Complexity (CC)

มาตรวัดความซับซ้อนของ McCabe's Cyclomatic Complexity พัฒนาขึ้นโดย Thomas J. McCabe ในปี พ.ศ. 2519 เพื่อใช้วัดความซับซ้อนของรหัสต้นฉบับของโปรแกรม โดยแมคเคบเสนอความคิดว่า ความยากในการเข้าใจรหัสต้นฉบับของโปรแกรมจะขึ้นอยู่กับกราฟกระแสการควบคุม (CFG) ของรหัสต้นฉบับของโปรแกรมนั้น การคำนวนค่าทำได้โดยแปลงรหัสต้นฉบับของโปรแกรมให้อยู่ในรูปของกราฟ (Control flow graph) ดังภาพที่ 2.6 ดังต่อไปนี้

```
public void evenOrOdd(int nums) {
        int i = nums;
        while (i<10) {
            System.out.print ("i = "+i);
            if (i%2 == 0) {
                System.out.println("even");
            } else {
                System.out.println("odd");
            }
        }
    }
```



ภาพที่ 2.6 การแปลงรหัสต้นฉบับของโปรแกรมให้อยู่ในโครงสร้างกราฟ

จากภาพที่ 2.6 การคำนวณค่า Cyclomatic complexity ทำได้โดยการนับจุด (Node) และเส้นเชื่อม (Edge) ที่ปรากฏ เพื่อใช้คำนวนความซับซ้อน ตามสมการดังนี้

$$M = E – N + 2P$$

โดยที่ M คือ ค่าของ Cyclomatic complexity

E คือ จำนวนเส้นเชื่อมในกราฟ

N คือ จำนวนจุดในกราฟ

P คือ จำนวนกราฟย่อยภายใน (Connected components)

- มาตรวัดความซับซ้อนของ Halstead

มาตรวัดความซับซ้อนของฮอลสตีด (Halstead) [5] พัฒนาขึ้นโดยศาสตราจารย์มอไรส์ ฮอลสตีด (Maurice Howard Halstead) ในปี พ.ศ. 2515 โดยฮอลสตีดตั้งสมมุติฐานว่าจำนวนจุดผิดในโปรแกรมสัมพันธ์กับจำนวนตัวดำเนินการ (Operator) และตัวถูกดำเนินการ (Operand) ในโปรแกรม ข้อสมมุติฐานนี้ถูกพิสูจน์โดยมีการทดสอบวัดจุดผิดของโปรแกรม (Bug) เทียบกับค่าจำนวนตัวดำเนินการและตัวถูกดำเนินการ ซึ่งผลการทดลองส่วนมากยอมรับได้ว่ามาตรวัดของฮอลสตีดถูกต้อง

ในการวัดความซับซ้อนจากรหัสต้นฉบับของโปรแกรม ฮอลสตีดได้เสนอมาตรวัดความซับซ้อนของโปรแกรมในหลาย ๆ ด้าน เช่น ความยาวของรหัสต้นฉบับของโปรแกรม (Program length) จำนวนคำศัพท์ที่รหัสต้นฉบับของโปรแกรมใช้ (Program vocabulary)

ปริมาตรของรหัสต้นฉบับของโปรแกรม (Volume) ระดับความยากของรหัสต้นฉบับของโปรแกรม (Program difficulty) และความพยายามที่ใช้ในการพัฒนารหัสต้นฉบับของโปรแกรม (Program effort) โดยในการคำนวนจะนับจำนวนตัวดำเนินการ (Operator) และตัวถูกดำเนินการ (Operand) ที่มีในรหัสต้นฉบับของโปรแกรมมาคำนวนค่าความซับซ้อน โดยมีนิยามดังนี้

n1 = จำนวนตัวดำเนินการที่ไม่ซ้ำกัน

n2 = จำนวนตัวถูกดำเนินการที่ไม่ซ้ำกัน

N1 = จำนวนตัวดำเนินการทั้งหมด

N2 = จำนวนตัวถูกดำเนินการทั้งหมด

ฮอลสตีดเสนอสูตรเพื่อวัดความซับซ้อนในด้านต่าง ๆ ที่น่าสนใจ จากพารามิเตอร์ทั้ง 4 ดังนี้

- ความยาวโปรแกรม (Length)

ฮอลสตีดตั้งข้อสังเกตว่าความยาวของโปรแกรมเป็นองค์ประกอบหนึ่งที่ทำให้โปรแกรมซับซ้อน โปรแกรมที่ยาวมาก มีแนวโน้มที่จะซับซ้อนกว่าโปรแกรมที่สั้น การคำนวนขนาดของโปรแกรมจะคำนวนจาก

$$N = N_1 + N_2$$

โดยที่ N คือ ขนาดของโปรแกรม

N_1 คือ จำนวนตัวดำเนินการทั้งหมด

N_2 คือ จำนวนตัวถูกดำเนินการทั้งหมด

- ปริมาตร (Volume)

ฮอลสตีดเสนอว่าปริมาตรของโปรแกรมหมายถึงจำนวนบิตที่ต้องใช้เพื่อแทนโปรแกรมนั้น เช่น ถ้าโปรแกรมมีความยาว N=4 และแต่ละคำใช้คำละ 2 บิต (00, 01, 10, 11) จะสามารถแทนโปรแกรมนั้นด้วย 8 บิต ดังนั้นถ้าโปรแกรมมีความยาว N และแต่ละคำในโปรแกรมนั้นต้องใช้ 4 บิต ปริมาตรของโปรแกรมจะเป็น 4 x N

จากค่า n_1 และ n_2 ทำให้เราทราบว่าในโปรแกรมหนึ่งจะมีค่าหรือสัญลักษณ์ที่ต่างกันอยู่ $n_1 + n_2$ ชนิด ดังนั้นแต่ละคำสามารถแทนด้วย $\log_2(n_1+n_2)$ บิต เช่น $n_1 + n_2 = 8$ หรือมีตัวดำเนินการและตัวถูกดำเนินการที่แตกต่างกันอยู่ 8 ตัว ซึ่งสามารถแทนด้วย $\log_2 8$ หรือ 3 บิต การคำนวนปริมาตรฮอลสตีดนิยามไว้ว่า

$$V = N \times \log_2(n_1 + n_2)$$

โดยที่ V คือ ปริมาตรของโปรแกรม

N คือ ขนาดของโปรแกรม

n_1 คือ จำนวนตัวดำเนินการทั้งหมด

n_2 คือ จำนวนตัวถูกดำเนินการทั้งหมด

- ระดับของการโปรแกรม (Program level)

ในการพัฒนาโปรแกรม ภาษาในการพัฒนาโปรแกรมที่เป็นภาษาชั้นสูงและต่ำ จะมีระดับของความเป็นนามธรรมต่างกัน โดยภาษาชั้นต่ำจะมีความเป็นนามธรรมน้อยกว่าภาษาชั้นสูง เนื่องจากภาษาระดับต่ำจะใช้ตัวดำเนินการที่พื้นฐานกว่า ซึ่งทำให้มีปริมาตรของโปรแกรมมาก และจะลดลงเมื่อระดับของภาษาที่ใช้พัฒนาสูงขึ้น จากนิยามนี้ฮอลสตีดจึงกำหนดความสัมพันธ์ว่า

$$L \times V = \text{ค่าคงที่} = V^*$$

โดยที่ L คือ ระดับของโปรแกรม

V คือ ปริมาตรของโปรแกรม

V^* คือ ปริมาตรศักยะ

ในการประมาณค่า L คำนวนได้จากสูตร

$$\tilde{L} = (2/n_1) \times (n_2/N_2)$$

โดยที่ $\tilde{L}$ คือ ค่าประมาณของระดับของโปรแกรม

N_2 คือ จำนวนของตัวถูกดำเนินการทั้งหมด

n_1 คือ จำนวนตัวดำเนินการที่ไม่ซ้ำกัน

n_2 คือ จำนวนตัวถูกดำเนินการที่ไม่ซ้ำกัน

สูตรการคำนวนระดับของการโปรแกรมนี้ มาจากแนวคิดที่ว่า ระดับของการโปรแกรมน่าจะลดลงถ้าจำนวนของตัวดำเนินการและตัวถูกดำเนินการทั้งหมดเพิ่มขึ้น และน่าจะเพิ่มขึ้นถ้าจำนวนชนิดตัวถูกดำเนินการต่าง ๆ เพิ่มขึ้น

ฮอลสตีดนิยามส่วนกลับของ L ว่าเป็นค่าความยากของโปรแกรม (Difficulty) หรือ D โดยมีแนวคิดว่าโปรแกรมเดียวกันที่เขียนด้วยภาษาชั้นต่ำจะมีค่าความยากในการพัฒนาโปรแกรมมากกว่าโปรแกรมที่เขียนด้วยภาษาชั้นสูง

- ความพยายาม (Effort)

ฮอลสตีดนิยามค่าความพยายามไว้ว่า แสดงถึงความพยายามของจิตใจในการเขียนโปรแกรมนั้น ๆ หรือในการอ่านทำความเข้าใจโปรแกรมนั้น ๆ ค่าความพยายามนี้ฮอลสตีดนิยามว่า

$$E = D \times V$$

โดยที่ E คือ ค่าความพยายาม

D คือ ค่าความยากของโปรแกรม

V คือ ปริมาตรของโปรแกรม

โดยสรุปมาตรวัดของฮอลสตีดที่น่าสนใจสำหรับงานวิจัยนี้ สรุปในตารางที่ 2.2

ตารางที่ 2.2 มาตรวัดของฮอลสตีด

มาตรวัด	สัญลักษณ์	สมการ
Program length	N	N = N1 + N2
Program vocabulary	n	n = n1 + n2
Volume	V	$V = N \times (\text{Log}_2\ n)$
Difficulty	D	D = (n1/2) x (N2/n2)
Effort	E	E = D x V

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1 การวิเคราะห์รหัสโปรแกรมคอมพิวเตอร์ : แนวทางและแผนกลยุทธ์

(Source Code Analysis : A Road Map) [6]

งานวิจัยนี้ได้กล่าวถึงความเป็นมา ความสำคัญของการวิเคราะห์รหัสโปรแกรมคอมพิวเตอร์ คุณลักษณะพื้นฐานของการวิเคราะห์รหัสโปรแกรมคอมพิวเตอร์ สรุปรูปแบบที่ใช้เป็นตัวแทนสำหรับเมื่อทำการวิเคราะห์รหัสโปรแกรมคอมพิวเตอร์ที่มีการใช้ในการศึกษาวิจัยในปัจจุบัน กลุ่มงานที่ได้นำไปประยุกต์ใช้ และได้รวบรวมและสรุปงานวิจัยที่ได้ทำการวิจัยทางด้านนี้แบ่งออกตามลักษณะแนวทางของงานวิจัย ทั้งในอดีต ปัจจุบัน ความท้าทายต่าง ๆ ที่กำลังเผชิญอยู่ของงานวิจัยในปัจจุบัน และได้ทำนายแนวโน้มและความท้าทายของงานวิจัยในอนาคต ในระยะ 10 ปี 20 ปี และ 50 ปี ต่อจากนี้ โดยจะระบุแนวโน้ม ปัญหา และอุปสรรคและข้อจำกัด รวมถึงผลกระทบจากการพัฒนาในด้านต่าง ๆ ที่อาจมีผลต่อลักษณะงานวิจัยในอนาคต

2.2.2 เงื่อนไขของการให้เกรดของการพัฒนาโปรแกรมของนักเรียน

(On Criteria for Grading Student Programs) [7]

งานวิจัยนี้ได้ศึกษาถึงเงื่อนไขในการให้คะแนนสำหรับการพัฒนารหัสโปรแกรมคอมพิวเตอร์ของนักเรียน โดยได้นำเสนอเงื่อนไขที่ใช้พิจารณาสำหรับการให้คะแนน 6 เงื่อนไข เรียงตามลำดับความสำคัญ ดังนี้

- การออกแบบโปรแกรม (Programming design)
- การดำเนินการของโปรแกรม (Programming execution)

- การตอบความต้องการของโจทย์ (Specification satisfaction)
- ลักษณะการเขียน (Coding style)
- การเขียนคำอธิบาย (Commenting)
- ความคิดสร้างสรรค์ (Creativity)

ทั้งนี้ได้นำเงื่อนไขที่กำหนดขึ้น ไปกำหนดเป็นเงื่อนไข และระบุรายละเอียดเงื่อนไขย่อยในแต่ละเงื่อนไขหลัก พร้อมคะแนนที่จะให้ในแต่ละส่วนแสดงให้กับนักเรียนตั้งแต่เริ่มเรียนวิชา เพื่อให้นักเรียนได้ทราบว่าหลักเกณฑ์การให้คะแนนเป็นอย่างไร ซึ่งผลที่ได้คือทำให้นักเรียนพึงพอใจในการเรียนการสอน และเข้าใจจุดมุ่งหมายในการเรียนได้ดีขึ้น มีการพัฒนาที่ดีซึ่งแสดงผลมาจากผลคะแนนที่ได้รับ

2.2.3 การสร้างส่วนต่างทางโครงสร้างของชุดรหัสโปรแกรมจาวา (Generating syntactical deltas from java source code) [8]

งานวิจัยนี้ได้กล่าวถึงการเปรียบเทียบส่วนต่างของชุดรหัสโปรแกรม ซึ่งโดยทั่วไปจะเปรียบเทียบบรรทัดต่อบรรทัด ว่าบรรทัดไหนต่างกัน แต่วิธีนี้มีข้อจำกัดคือไม่สามารถเปรียบเทียบในลักษณะเชิงโครงสร้างหรือเชิงสัญลักษณ์ได้ ซึ่งการเปรียบเทียบ 2 ลักษณะหลังนี้จะให้ความแม่นยำและให้ผลที่ดีกว่า งานวิจัยนี้จึงได้นำเสนอวิธีในการเปรียบเทียบโดยการนำต้นไม้ AST มาช่วยในการเปรียบเทียบระหว่าง 2 โปรแกรม โดยจะเก็บผลความแตกต่างของ 2 โปรแกรม ในลักษณะของชุดคำสั่ง (Edit script) ที่จะเปลี่ยนต้นไม้หนึ่งไปเป็นอีกต้นไม้หนึ่ง ผลของการเปรียบเทียบในเชิงไวยากรณ์ของภาษานี้โดยใช้โครงสร้างต้นไม้ AST จะให้ผลลัพธ์และความยืดหยุ่นที่ดีกว่าการเปรียบเทียบข้อความโดยตรง แต่จะมีข้อเสียคือจะผูกติดกับภาษานั้น ๆ ไม่สามารถนำไปใช้กับชุดรหัสโปรแกรมภาษาอื่นได้ และขึ้นอยู่กับข้อจำกัดของตัวแปรภาษา และเสนอแนวทางในการพัฒนาต่อ ว่าควรจะพัฒนาอัลกอริทึมในการเปรียบเทียบให้ดีขึ้นกว่านี้ รวมทั้งควรศึกษาระบุรูปแบบการจัดเก็บเอกสารของความแตกต่างของโปรแกรมให้เป็นรูปแบบมาตรฐาน

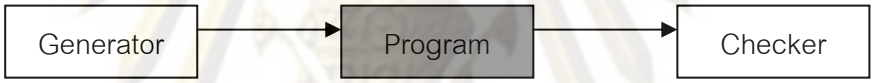
2.2.4 ระบบอัตโนมัติสำหรับการให้คะแนนแบบฝึกหัดการพัฒนาโปรแกรม (Automatic Grading of Programming Exercises) [9]

งานวิจัยนี้นำเสนอระบบการให้คะแนนอัตโนมัติสำหรับแบบฝึกหัดของการพัฒนาโปรแกรม โดยกล่าวถึงระบบการให้คะแนนอัตโนมัติที่มีอยู่ในปัจจุบัน จะใช้การเปรียบเทียบโปรแกรมที่พัฒนาขึ้นของนักเรียนกับโปรแกรมที่ทำเฉลยไว้แล้ว โดยจะมีวิธีการตรวจสอบโดยให้การวิเคราะห์เชิงสถิต (Static analysis) และการวิเคราะห์เชิงพลวัติ (Dynamic analysis) การวิเคราะห์เชิงสถิตจะใช้เมทริกซ์ในการเปรียบเทียบโปรแกรม ไม่ว่าจะเป็นความสามารถในการอ่านได้ของโปรแกรม ลักษณะการเขียนโปรแกรม หรือความยุ่งยากในการดูแลรักษา เป็นต้น ส่วนใน

การวิเคราะห์เชิงพลวัติจะใช้การดำเนินการของโปรแกรมโดยมีตัวทำนายจัดทำข้อมูลตัวอย่าง เพื่อมาทดสอบและเปรียบเทียบความถูกต้องกับผลลัพธ์ที่ได้

งานวิจัยนี้มีเป้าหมายเพื่อสร้างสภาวะแวดล้อมที่สามารถควบคุมได้สำหรับการประเมินทักษะทางการพัฒนาโปรแกรมของผู้เรียน โดยจะแบ่งเป็นส่วนสภาวะแวดล้อมสำหรับการแก้ปัญหา และส่วนสภาวะแวดล้อมสำหรับการตรวจสอบดูแล โดยจะสร้างผู้ทำนาย (Oracle) ขึ้นมาทำหน้าที่ในการสร้างข้อมูลทดสอบสำหรับแบบฝึกหัด และตรวจสอบผลลัพธ์ที่ได้ ดังตารางที่ 2.8 และให้คะแนนสำหรับแบบฝึกหัด และรายงานผลการผิดพลาดแก่นักเรียน

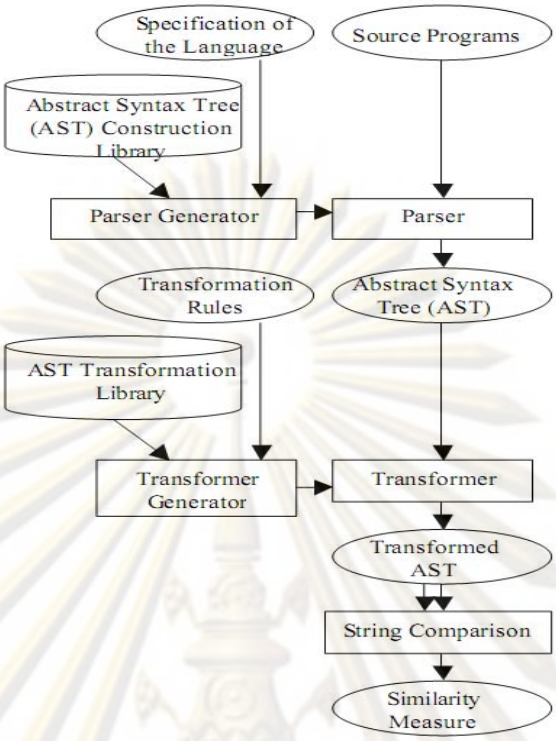
นอกจากนี้งานวิจัยนี้นำเสนอแนวทางพัฒนาต่อว่า ความท้าทายของการพัฒนาระบบให้คะแนนอัตโนมัติ จะอยู่ที่การสร้างผู้ทำนายสำหรับการให้คะแนนโดยอัตโนมัติ โดยจะเน้นไปในแนวทางเชิงพลวัติมากว่าเชิงสถิต และสรุปได้ว่า ข้อมูลด้านคุณภาพและความยุ่งยากในการดูแลรักษาที่ได้จากการวิเคราะห์รหัสโปรแกรมเชิงสถิตนั้นจะเหมาะสมกับโปรแกรมขนาดใหญ่ แต่จะไม่เหมาะสมกับโปรแกรมขนาดเล็กอย่างแบบฝึกหัด



ภาพที่ 2.7 The “black box” model of the traditional oracle

2.2.5 ระบบอัตโนมัติในการตรวจการลอกกันของโปรแกรมคอมพิวเตอร์ของนักเรียน (Automatic Generation of Plagiarism Detection Among Student Programs)[10]

งานวิจัยนี้ได้นำเสนอวิธีการในการตรวจเช็คการลอกงานกันของนักเรียน โดยนำเสนออัลกอริทึมในการทำงานแบบ 2 ขั้นตอน โดยขั้นแรกคือ การนำรหัสโปรแกรมผ่านตัวแจงส่วน (Parser) ให้อยู่ในรูปแบบต้นไม้ จากนั้นจะนำมาผ่านการปรับรูปแบบตามกฎเกณฑ์ที่ตั้งไว้ เพื่อเปลี่ยนให้อยู่ในรูปแบบข้อความและนำมาเปรียบเทียบด้วยอัลกอริทึมการจับคู่ข้อความที่ยาวที่สุด (Longest string matching) ซึ่งผลลัพธ์ที่ได้จะสามารถตรวจสอบการลอกกันได้ดีขึ้น โดยวิธีการที่นำเสนอนี้จะไม่ถูกหลอกโดยการเพิ่มคำบรรยาย การเปลี่ยนชื่อตัวแปร การเปลี่ยนย้ายตำแหน่งข้อความสั่ง การเพิ่มข้อความสั่งที่ไม่ได้ใช้ การเพิ่มข้อความสั่งที่ซ้ำซ้อน การปรับเปลี่ยนโครงสร้าง ซึ่งจะครอบคลุมมากกว่าวิธีการเปรียบเทียบโดยข้อความ หรือโดยโครงสร้าง / วายกสัมพันธ์ ดังแสดงขั้นตอนการทำงานในภาพที่ 2.8

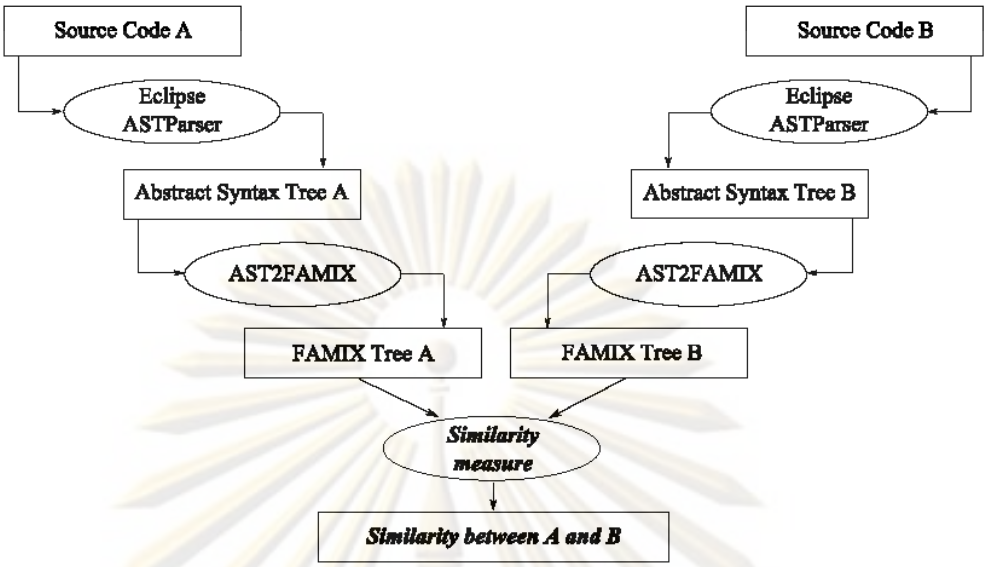


ภาพที่ 2.8 แผนผังการทำงานของระบบอัตโนมัติในการตรวจการลอกกันของโปรแกรมคอมพิวเตอร์ของนักเรียน

2.2.6 การตรวจสอบความคล้ายของคลาสของโปรแกรมภาษาจาวาด้วยอัลกอริทึมของต้นไม้

(Detecting Similar Java Classes using Tree Algorithms) [11]

งานวิจัยนี้ได้นำเสนอแนวคิดในการนำโครงสร้างข้อมูลแบบต้นไม้มาใช้ในการเปรียบเทียบคลาสของโปรแกรมภาษาจาวา โดยการศึกษาเปรียบเทียบอัลกอริทึมที่ใช้ตรวจหาความละม้ายของต้นไม้ 3 อัลกอริทึม คือ Bottom-up maximum common subtree isomorphism, Top-down maximum common subtree isomorphism และ Tree edit distance และพบว่าอัลกอริทึม Tree edit distance ให้ผลลัพธ์ในการตรวจสอบความละม้ายได้ดีที่สุด โดยใช้ข้อมูลทดสอบการเปลี่ยนแปลงที่เตรียมไว้คือ เพิ่ม Constructor เพิ่มตัวแปร เพิ่มการเรียกเมท็อด การแยกเมท็อด และการ Implement interface และพัฒนาระบบ Coogle ขึ้นมาใช้สำหรับเปรียบเทียบโดยมีการนำโมเดล FAMIX ซึ่งเป็นโมเดลที่ไม่ขึ้นกับภาษาของโปรแกรมมาใช้ในการเปรียบเทียบ โดยมีขั้นตอนการทำงานของระบบดังภาพที่ 2.9



ภาพที่ 2.9 ขั้นตอนการทำงานของระบบ Coogle

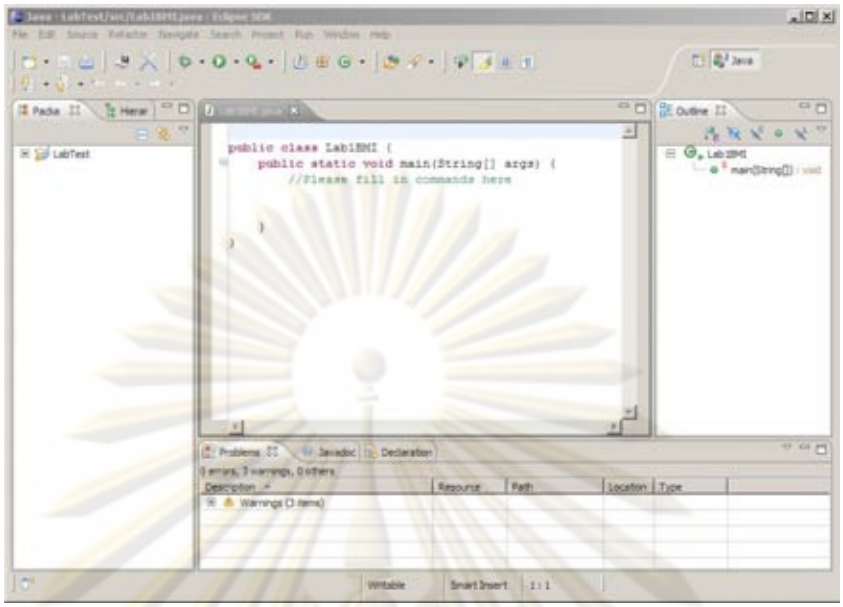
2.2.7 การตรวจสอบการลอกกันโดยการวิเคราะห์กราฟความขึ้นต่อกันของโปรแกรม (GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis) [12]

งานวิจัยนี้ได้นำเสนอเครื่องมือในการตรวจสอบการลอกโปรแกรมกัน โดยนำกราฟความขึ้นต่อกัน (PDG หรือ Program dependency graph [13]) มาใช้ในการตรวจสอบการลอกกัน ซึ่งงานวิจัยนี้อ้างว่าระบบตรวจสอบการลอกกันที่มีอยู่ในปัจจุบันเพียงพอต่อการใช้งานในระดับมหาวิทยาลัย แต่ยังคงไม่สามารถตรวจพบการลอกกันในบางรูปแบบที่สำคัญ ๆ โดยกราฟความขึ้นต่อกันของโปรแกรม แสดงถึงข้อมูลและชุดคำสั่งที่ขึ้นต่อกันภายในเมทอด และนำตัวกรองข้อมูลที่เก็บสถิติ (statistical lossy filter) มาช่วยในการเพิ่มประสิทธิภาพในการตรวจสอบ เพื่อให้สามารถนำไปใช้กับโปรแกรมขนาดใหญ่ได้อย่างมีประสิทธิภาพ

2.3 เครื่องมือที่ใช้ในการวิจัย

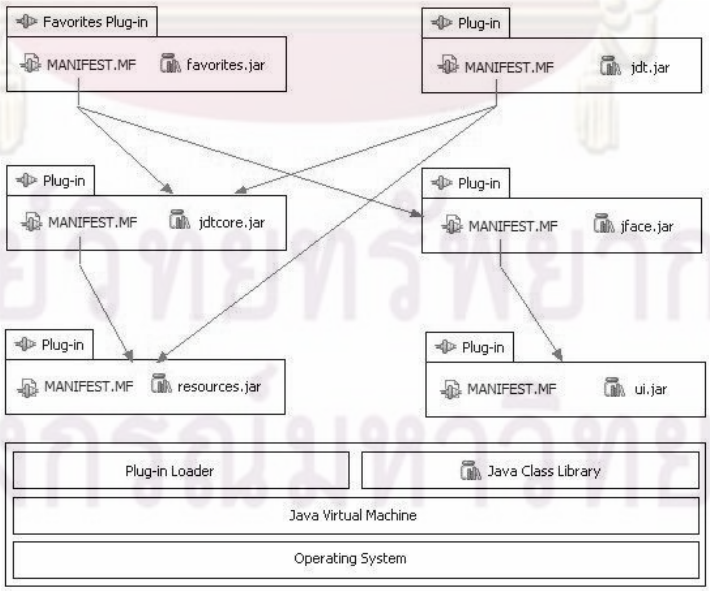
2.3.1 โปรแกรม Eclipse IDE

โปรแกรม Eclipse IDE (Integrated Development Environment) เป็นโปรแกรมที่ใช้ในการพัฒนาโปรแกรมภาษาจาวา ซึ่งเป็นสภาวะแวดล้อมหลักที่ให้ผู้ทดสอบใช้พัฒนาโปรแกรม ลักษณะหน้าต่างการทำงานของโปรแกรมแสดงดังภาพที่ 2.10



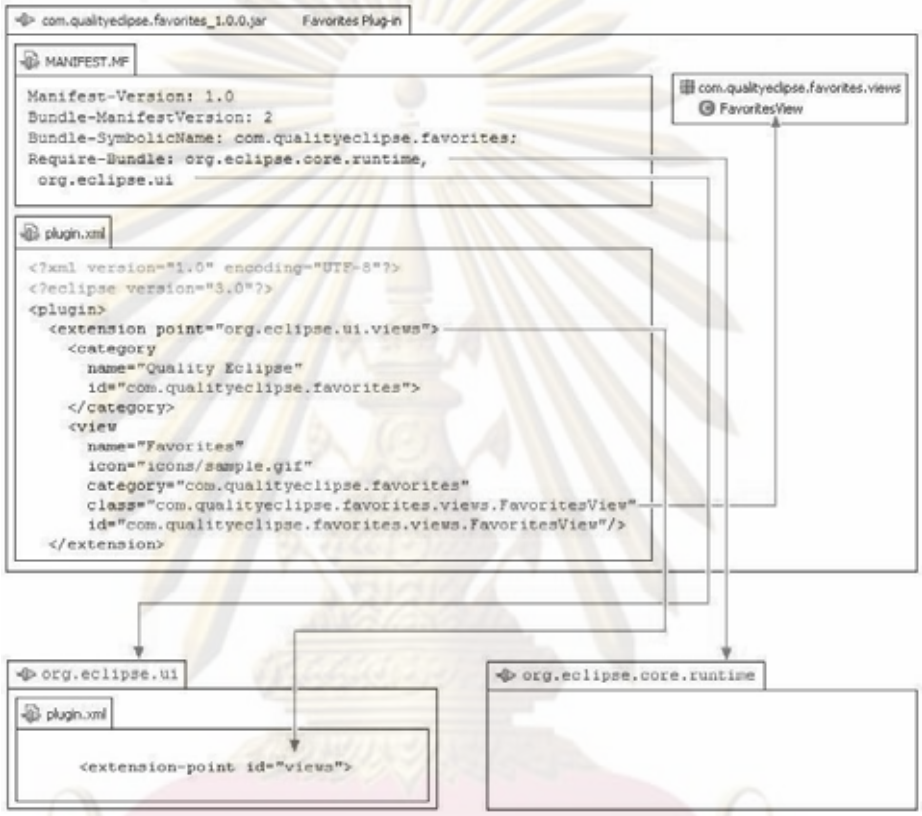
ภาพที่ 2.10 หน้าต่างการทำงานของโปรแกรม Eclipse IDE

โครงสร้างของโปรแกรม Eclipse IDE มีลักษณะเป็นเคอร์เนลขนาดเล็กที่ประกอบด้วยโปรแกรมที่ใช้เรียกโปรแกรมเสริมให้ทำงาน (Plugin loader) ซึ่งพัฒนาตามมาตรฐานของ OSGi R4 ในการใช้งาน โปรแกรมที่ใช้เรียกโปรแกรมเสริมให้ทำงานจะเรียกโปรแกรมเสริมย่อย ๆ จำนวนมากขึ้นมาทำงาน โดยโปรแกรมเสริมแต่ละตัวสามารถทำงานอิสระต่อกัน หรือขึ้นต่อกัน ขึ้นกับการออกแบบการทำงานของโปรแกรมเสริมนั้น ๆ โครงสร้างสถาปัตยกรรมของโปรแกรม Eclipse ที่ประกอบขึ้นจากโปรแกรมเสริมย่อย ๆ แสดงดังภาพที่ 2.11



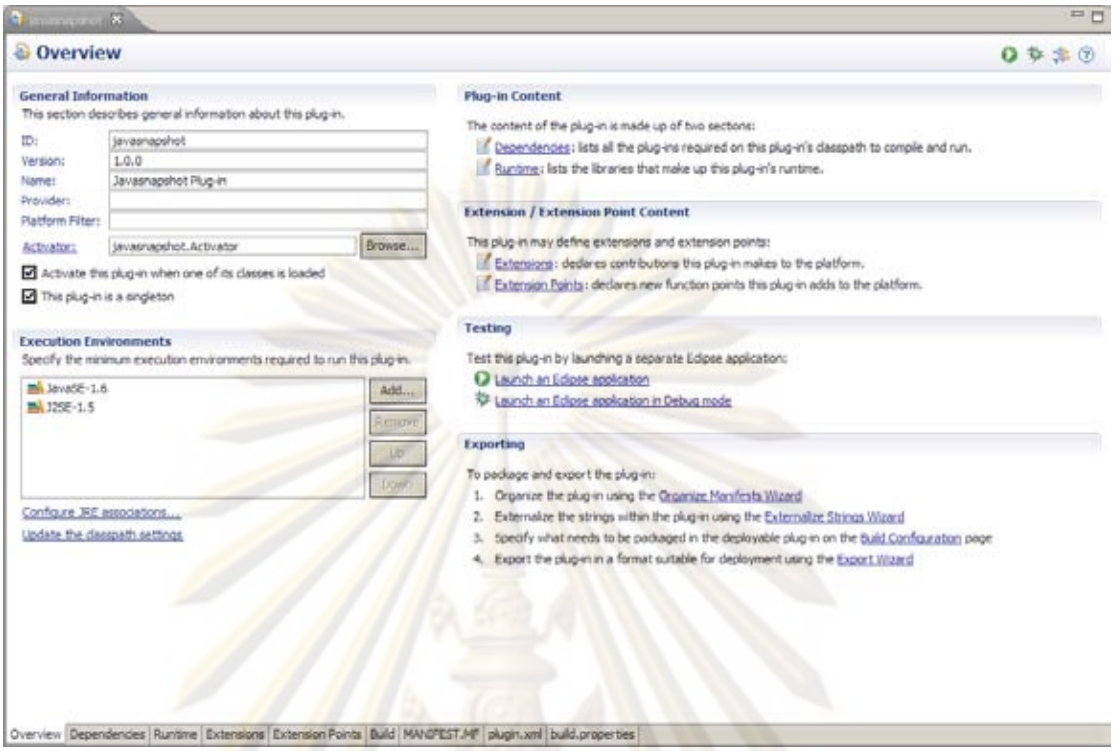
ภาพที่ 2.11 สถาปัตยกรรมของโปรแกรมเสริมภายในโปรแกรม Eclipse

จากสถาปัตยกรรมของโปรแกรม Eclipse ดังภาพที่ 2.11 ในการพัฒนาโปรแกรมเสริมเพื่อขยายความสามารถของระบบ โปรแกรม Eclipse จะเผยแพร่จุดเชื่อมต่อ (Extension points) จำนวนมากให้กับผู้พัฒนาภายนอก เพื่อให้สามารถพัฒนาโปรแกรมเสริมเพิ่มเติมได้ตามจุดต่าง ๆ ที่ต้องการ ดังตัวอย่างที่แสดงในภาพที่ 2.12

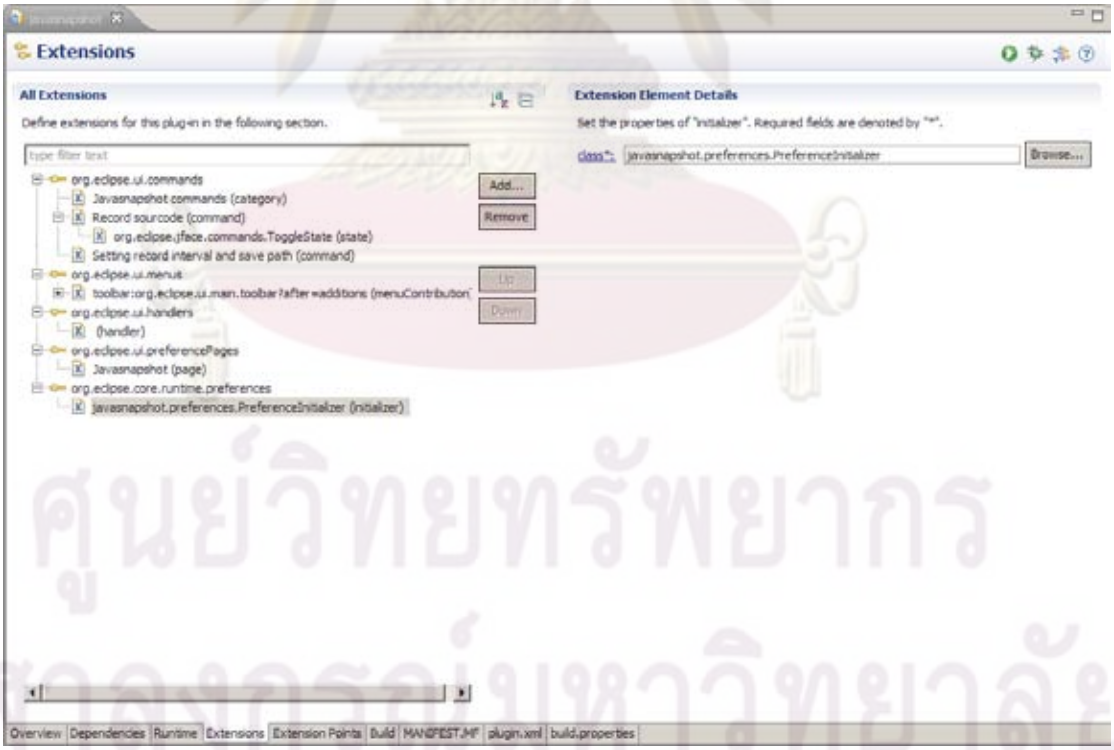


ภาพที่ 2.12 ตัวอย่างการประกาศการเชื่อมต่อโปรแกรมเสริมของโปรแกรม Eclipse

จากภาพที่ 2.12 ในการพัฒนาโปรแกรมเสริมเพื่อเชื่อมต่อเข้ากับโปรแกรม Eclipse ในส่วนของไฟล์ Plugin.xml จะใช้ประกาศโปรแกรมเสริม และจุดเชื่อมต่อที่จะใช้เชื่อมเข้ากับโปรแกรมของ Eclipse ซึ่งจากภาพที่ 2.12 แสดงให้เห็นว่าโปรแกรมเสริมนี้ เชื่อมต่อเข้ากับจุดเชื่อมต่อขยายที่ชื่อ org.eclipse.ui.views และโปรแกรมเสริมนี้ทำงานโดยเรียกใช้คลาสที่ชื่อ com.qualityeclipse.favorites.views.FavoritesView ส่วนของไฟล์ MANIFEST.MF จะเป็นไฟล์ที่ใช้ระบุข้อมูลต่าง ๆ ของโปรแกรมเสริมนี้ เช่น รายละเอียดของโปรแกรมเสริม รุ่นของจาวาที่ต้องการ ไลบรารีที่จะถูกเรียกใช้ในโปรแกรมเสริม เป็นต้น ซึ่งการพัฒนาโปรแกรมเสริมในโปรแกรม Eclipse โปรแกรม Eclipse ได้เตรียมเครื่องมือที่ช่วยในการพัฒนาไว้ให้ โดยมีหน้าต่างที่ใช้กำหนดค่าต่าง ๆ ของการสร้างโปรแกรมเสริม ดังตัวอย่างที่แสดงในภาพที่ 2.13 และภาพที่ 2.14



ภาพที่ 2.13 การตั้งค่ารายละเอียดต่าง ๆ ของโปรแกรมเสริมที่พัฒนาขึ้นด้วยโปรแกรม Eclipse



ภาพที่ 2.14 การกำหนดรายละเอียดการเชื่อมต่อของโปรแกรมเสริมที่พัฒนาขึ้น

จากภาพที่ 2.13 และ 2.14 แสดงตัวอย่างของหน้าต่างในขั้นตอนการพัฒนาโปรแกรมเสริม เพื่อกำหนดค่าต่าง ๆ ที่จำเป็นของโปรแกรมเสริม และภาพที่ 2.14 แสดงในส่วนของการประกาศจุดเชื่อมต่อที่เรียกใช้

2.3.2 โปรแกรม Plaggie

โปรแกรม Plaggie [14] เป็นโปรแกรมที่ใช้ในการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมของภาษาจาวา ที่พัฒนาขึ้นโดย Aleksi Ahtiainen ที่มหาวิทยาลัยเฮลซิงกิ (Helsinki University of Technology) ประเทศฟินแลนด์ เพื่อใช้ตรวจการลอกกัน (Plagiarism detection engine) ของการบ้านภาษาจาวาของผู้เรียนที่มหาวิทยาลัยเฮลซิงกิ และได้เผยแพร่ในงานสัมมนา Baltic Sea Conference on Computing Education Research - Koli Calling 2006 ครั้งที่ 6 ซึ่งจัดโดยความร่วมมือระหว่าง ACM SIGSCE และ UpCERG (Uppsala Computing Education Research Group, Department of Information Technology, Uppsala University, Sweden) และ CeTUSS (nationellt ämnesdidaktiskt Centrum för TeknikUndervisning i Studenternas Sammanhang)

ผู้พัฒนาโปรแกรม Plaggie เผยแพร่โปรแกรมให้ใช้งานในลักษณะซอฟต์แวร์เสรีภายใต้สิทธิการอนุญาตแบบ GNU General Public License ซึ่งโปรแกรมทั่วไปที่เป็นที่นิยมในการใช้ตรวจสอบการลอกกัน (Plagiarism detection) จากที่มีจำนวนการอ้างอิงถึงมากในงานวิจัยหรือบทความประเภทการตรวจการลอกกัน เช่น MOSS [15] หรือ JPlag [3] จะไม่เปิดให้บุคคลภายนอกนำเอาโปรแกรมของเขามาพัฒนาต่อเพื่อใช้วิจัยเพิ่มเติมได้ แต่จะเปิดให้บุคคลภายนอกใช้ได้โดยการสมัครสมาชิกและส่งข้อมูลที่ต้องการจะเปรียบเทียบเข้าไปที่ระบบของเขา จากนั้นระบบจะส่งผลการเปรียบเทียบกลับมาให้ ซึ่งวิธีนี้ไม่สะดวกต่อการนำมาใช้เป็นเครื่องมือทำการวิจัย ที่ต้องการการพัฒนาเพิ่มเติมตามรูปแบบอื่น ๆ ที่ต้องการทดสอบ และผู้ที่สามารถสมัครได้จะต้องมีตำแหน่งเป็นผู้สอนเท่านั้น ไม่เปิดให้บุคคลทั่วไปสมัครใช้งาน ซึ่งจะมีการตรวจสอบสถานะหลังจากสมัครแล้ว โปรแกรม Plaggie จึงให้ความยืดหยุ่นในการนำมาใช้ในการวิจัยมากกว่า

การทำงานของโปรแกรม Plaggie จะนำอัลกอริทึมในการเปรียบเทียบความละม้ายของโปรแกรม JPlag ซึ่งใช้อัลกอริทึม RKR-GST (Greedy String Tiling with Running Karp-Rabin Matching) [2] ซึ่งได้เผยแพร่ตัวอย่างรหัสเทียม (Pseudo code) ในบทความที่ตีพิมพ์ในงานวิจัยของ JPlag [3] มาพัฒนาและปรับปรุงเพิ่มเติม จึงให้ผลลัพธ์ในการเปรียบเทียบ

ความละม้ายที่เทียบเคียงกัน และผู้พัฒนาโปรแกรม Plaggie ได้พัฒนาเพิ่มเติมในส่วนของการแยกส่วน ที่เป็นแม่แบบของรหัสต้นฉบับของโปรแกรมออกจากส่วนของการเปรียบเทียบความละม้าย ซึ่งเหมาะต่อการนำมาใช้สำหรับการเปรียบเทียบแบบทดสอบภาษาจาวาที่ให้ผู้เรียนปฏิบัติในปัจจุบัน เพราะแบบทดสอบบางชุดในการทำปฏิบัติการของผู้เรียน จะมีส่วนของรหัสต้นฉบับของโปรแกรมบางส่วนขึ้นต้นไว้ให้ผู้เรียนอยู่แล้ว การเปรียบเทียบและให้คะแนนจึงไม่ควรนำส่วนที่ผู้สอนขึ้นต้นไว้ให้มาคำนวนให้คะแนนด้วย ซึ่งในการเปรียบเทียบความละม้าย โปรแกรม Plaggie กำหนดประเภทของโทเค็นดังตารางที่ 2.3

ตารางที่ 2.3 ประเภทของโทเค็นในโปรแกรม Plaggie

ABSTRACT_METHOD_DECLARATION	IF
ANONYMOUS_INNER_CLASS	IF_END
ANONYMOUS_INNER_CLASS_END	IMPORT_DECLARATION
ASSERT	INNER_CLASS_DECLARATION
ASSIGNMENT	INNER_CLASS_DECLARATION_END
BLOCK	INNER_ENUM_DECLARATION
BLOCK_END	INNER_ENUM_DECLARATION_END
BREAK	INNER_INTERFACE_DECLARATION
CASE	INNER_INTERFACE_DECLARATION_END
CATCH	INTERFACE_DECLARATION
CATCH_END	INTERFACE_DECLARATION_END
CLASS_DECLARATION	METHOD_DECLARATION
CLASS_DECLARATION_END	METHOD_DECLARATION_END
CONSTANT_DECLARATION	METHOD_INVOCATION
CONSTRUCTOR_DECLARATION	NEW
CONSTRUCTOR_DECLARATION_END	PACKAGE_DECLARATION
CONSTRUCTOR_INVOCATION_THIS	RETURN
CONSTRUCTOR_INVOCATION_SUPER	STATIC_INITIALIZATION
CONTINUE	SWITCH

DO	SWITCH_END
DO_END	SYNCHRONIZED
ELSE	SYNCHRONIZED_END
ELSE_END	THROW
ENUM_DECLARATION	TRY
ENUM_DECLARATION_END	TRY_END
FINALLY	VARIABLE_DECLARATION
FINALLY_END	WHILE
FOR	WHILE_END
FOR_END	

ผลการตัดแบ่งโทเค็นด้วยโปรแกรม Plaggie จากรหัสต้นฉบับของโปรแกรมดังภาพที่ 2.15 ได้ผลลัพธ์ดังตารางที่ 2.4

```
import java.util.Scanner;
  public class Area {
     public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("รัศมี = ");
        double r = sc.nextDouble();
        double area = Math.PI * r * r;
        System.out.println("พื้นที่ = "+area);
  }
}
```

ภาพที่ 2.15 ตัวอย่างรหัสต้นฉบับของโปรแกรมที่นำมาตัดโทเค็นด้วยโปรแกรม Plaggie

ตารางที่ 2.4 ผลการตัดแบ่งรหัสต้นฉบับของโปรแกรมเป็นโทเค็นด้วยโปรแกรม Plaggie

IMPORT_DECLARATION	import java.util.Scanner;
CLASS_DECLARATION	public class Area {
METHOD_DECLARATION	main(String[] args) {

VARIABLE_DECLARATION	sc = new Scanner(System.in)
NEW	new Scanner(System.in)
METHOD_INVOCATION	System.out.println(“รัศมี = ")
VARIABLE_DECLARATION	r = sc.nextDouble()
METHOD_INVOCATION	sc.nextDouble()
VARIABLE_DECLARATION	area = Math.PI * r * r
METHOD_INVOCATION	System.out.println(“รัศมี = "+area)
METHOD_DECLARATION_END	}
CLASS_DECLARATION_END	}

2.3.3 ไลบรารี ANTLR (Another Tool for Language Recognition)

ไลบรารี ANTLR เป็นเครื่องมือที่ใช้ในการวิเคราะห์ศัพท์ (Lexical Analysis) และแจงส่วน (Parsing) ตามไวยากรณ์ของภาษาที่กำหนด ในการวิเคราะห์รหัสต้นฉบับโปรแกรมของภาษาจาวา จะนำไลบรารี ANTLR และไวยากรณ์ภาษาจาวา 1.5 มาสร้างตัววิเคราะห์ศัพท์ (Lexer) และตัวแจงส่วน (Parser) เพื่อใช้ในการตรวจสอบความถูกต้องของรหัสต้นฉบับของโปรแกรมว่าถูกไวยากรณ์หรือไม่ และแบ่งรหัสต้นฉบับของโปรแกรมออกเป็นโทเค็นย่อย ๆ ตามประเภทของโทเค็นซึ่งแบ่งเป็น 114 ประเภท ดังแสดงในตารางที่ 2.5

ตารางที่ 2.5 ประเภทของโทเค็นของภาษาจาวาที่ได้จากการตัดคำด้วยไลบรารี ANTLR

<invalid>	char
<EOR>	byte
<DOWN>	short
<UP>	int
Identifier	long
ENUM	float
FloatingPointLiteral	double
CharacterLiteral	?
StringLiteral	super

HexLiteral	(
OctalLiteral	)
DecimalLiteral	…
ASSERT	this
HexDigit	null
IntegerTypeSuffix	true
Exponent	false
FloatTypeSuffix	@
EscapeSequence	default
UnicodeEscape	:
OctalEscape	if
letter	else
JavaIDDigit	for
WS	while
COMMENT	double
LINE_COMMENT	try
package	finally
;	switch
import	return
static	throws
.	break
*	continue
public	catch
protected	case
private	+=
abstract	-=
final	*=
strictfp	/=
class	&=

extends	|=
implements	^=
<	%=
,	||
>	&&
&	|
{	^
}	==
interface	!=
void	instanceof
[	+
]	-
throws	/
=	%
native	++
synchronized	--
transient	~
volatile	!
boolean	new

ในบทที่ 2 ผู้วิจัยได้ศึกษาเอกสารแนวคิดทฤษฎีการให้คะแนนอัตโนมัติแบบต่าง ๆ ตั้งแต่รูปแบบการวิเคราะห์รหัสต้นฉบับของโปรแกรม วิธีการต่าง ๆ ในการเทียบเคียงรหัสต้นฉบับของโปรแกรมกับชุดเฉลย และการประเมินรหัสต้นฉบับของโปรแกรมด้วยมาตรวัดทางวิศวกรรมซอฟต์แวร์ รวมทั้งได้ศึกษางานวิจัยและเครื่องมือที่เกี่ยวข้องในการวิจัย เพื่อเป็นความรู้พื้นฐานในการดำเนินการวิจัยในขั้นต่อไป

บทที่ 3

วิธีดำเนินการวิจัย

การออกแบบการวิจัยระบบการให้คะแนนอัตโนมัติ โดยการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม เพื่อศึกษาลักษณะของลำดับการเปลี่ยนแปลงของโปรแกรมของผู้เรียนว่ามีผลต่อการประเมินความรู้ความเข้าใจและการได้คะแนนของผู้เรียนอย่างไร และพัฒนาโปรแกรมเพื่อช่วยในการตรวจให้คะแนนอัตโนมัติ เพื่อวัตถุประสงค์ดังกล่าว ผู้วิจัยออกแบบการวิจัยและดำเนินการวิจัย ตามขั้นตอนการพัฒนาโปรแกรมดังต่อไปนี้

3.1 ภาพรวมของระบบ

ระบบการให้คะแนนอัตโนมัติ โดยการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม แบ่งการทำงานของระบบออกเป็น 3 ส่วน คือ

- ส่วนเก็บข้อมูล

ส่วนเก็บข้อมูลเป็นการเก็บข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมระหว่างที่ผู้เรียนทำปฏิบัติการ โดยจะตรวจจับการพิมพ์คำสั่งเพิ่ม ลบคำสั่ง หรือแก้ไขเปลี่ยนแปลงคำสั่งและบันทึกลงไฟล์ไว้

- ส่วนวิเคราะห์ความละม้าย

ส่วนวิเคราะห์ความละม้ายได้นำวิธีการของการตรวจสอบการลอกการบ้านหรือโปรแกรมกัน (Plagiarism detection) มาใช้ทำการเปรียบเทียบความละม้ายของลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่บันทึกได้จำนวน n ชุด ของผู้เรียน กับชุดเฉลยจำนวน m ชุด เพื่อหาคู่ของรหัสต้นฉบับของโปรแกรมของผู้เรียนกับชุดของเฉลยที่คล้ายที่สุด เพื่อนำมาใช้เปรียบเทียบให้คะแนนในส่วนต่อไป โดยมีแนวคิดในการนำคอมพิวเตอร์มาคำนวนให้คะแนนจากรหัสต้นฉบับของโปรแกรมชุดใด ๆ เช่น จากตัวอย่างรหัสต้นฉบับของโปรแกรมดังภาพที่ 3.1

```
public class Circle {
        public static void main(String[] args) {
                String s1 = "เส้นรอบวง = ";
                String s2 = "พื้นที่ = ";
                double radius = 5;
                double circumference, area;
                circumference = 2 * 3.14159 * radius;
        }
}
```

ภาพที่ 3.1 ตัวอย่างรหัสต้นฉบับของโปรแกรมคำนวนเส้นรอบวงและพื้นที่ ณ เวลา T วินาที

จากภาพที่ 3.1 ถ้าคอมพิวเตอร์ไม่รู้ว่าจุดสุดท้ายของรหัสต้นฉบับของโปรแกรมจะเป็นอย่างไร คอมพิวเตอร์จะไม่ทราบว่าจากรหัสต้นฉบับของโปรแกรมนี้จะต้องการคำสั่งเพิ่มอีกกี่คำสั่งถึงจะเสร็จสิ้น จึงไม่สามารถประเมินได้ว่ารหัสต้นฉบับของโปรแกรมดำเนินมาได้ไกลเท่าไหร่ ถ้าสามารถรู้ได้ว่ารหัสต้นฉบับของโปรแกรม ณ จุดสิ้นสุดเป็นดังภาพที่ 3.2

```
public class Circle {
        public static void main(String[] args) {
                String s1 = "เส้นรอบวง = ";
                String s2 = "พื้นที่ = ";
                double radius = 5;
                double circumference, area;
                circumference = 2 * 3.14159 * radius;
                area = 3.14159 * radius * radius;
                System.out.println(s1 + circumference);
                System.out.println(s2 + area);
        }
}
```

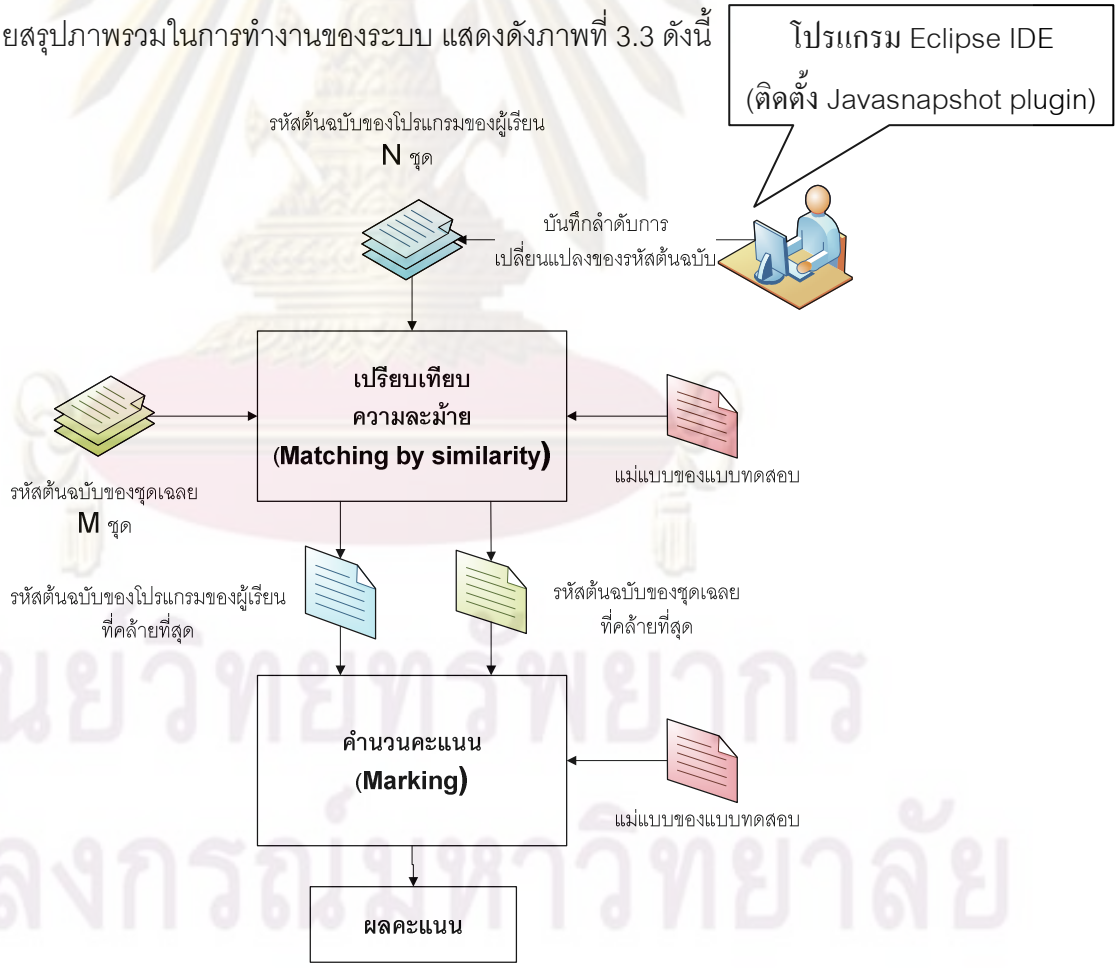
ภาพที่ 3.2 ตัวอย่างรหัสต้นฉบับของโปรแกรมคำนวนเส้นรอบวงและพื้นที่ ขณะเสร็จสมบูรณ์

จากภาพที่ 3.2 เมื่อดูจากรหัสต้นฉบับของโปรแกรมที่เสร็จสมบูรณ์ ทำให้รู้ว่าจากรหัสต้นฉบับของโปรแกรมในภาพที่ 3.1 เพิ่มคำสั่งอีก 3 บรรทัด โปรแกรมจะเสร็จสมบูรณ์ ดังนั้นถ้าสามารถทราบจุดหมายปลายทางของรหัสต้นฉบับของโปรแกรมแล้ว จะทำให้สามารถประเมินได้ว่ารหัสต้นฉบับของโปรแกรม ณ ขณะที่สนใจ ดำเนินมาไกลเพียงใดเมื่อเทียบกับรหัสต้นฉบับของโปรแกรมที่เสร็จสมบูรณ์

- ส่วนให้คะแนน

ส่วนการให้คะแนน หลังจากที่ได้คู่ของรหัสต้นฉบับของโปรแกรมของผู้เรียนกับชุดเฉลยที่ละม้ายที่สุด จะได้แนวทางที่ผู้เรียนใช้ในการแก้ไขปัญหา และทราบว่าในการแก้ปัญหาจะไปสิ้นสุดในรูปแบบใด จากนั้นจึงนำมาตรวัดทางด้านวิศวกรรมซอฟต์แวร์มาประเมินเนื้องานของรหัสต้นฉบับของโปรแกรมของผู้เรียน และชุดเฉลย เพื่อเทียบเป็นสัดส่วนให้คะแนน

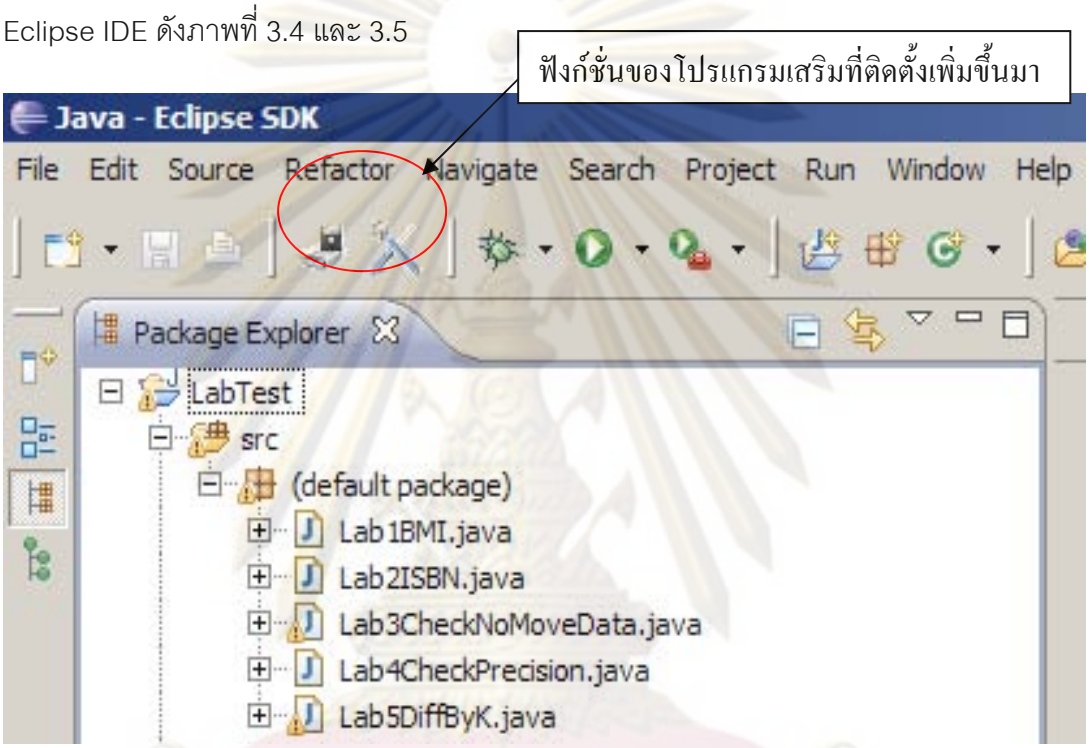
โดยสรุปภาพรวมในการทำงานของระบบ แสดงดังภาพที่ 3.3 ดังนี้



ภาพที่ 3.3 แนวคิดในการทำงานของระบบการตรวจข้อสอบอัตโนมัติ โดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม

3.2 การพัฒนาโปรแกรมเสริม Javasnapshot เพื่อบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม

การพัฒนาโปรแกรมเสริม Javasnapshot (plug-in) เพื่อบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม ได้พัฒนาโปรแกรมเสริมเพื่อติดตั้งเพิ่มลงในโปรแกรม Eclipse IDE เพื่อใช้ในการเก็บข้อมูลระหว่างการพัฒนาโปรแกรมของผู้ทดสอบ เมื่อติดตั้งแล้วจะปรากฏในโปรแกรม Eclipse IDE ดังภาพที่ 3.4 และ 3.5



ภาพที่ 3.4 โปรแกรมเสริมที่ใช้เก็บข้อมูลที่ติดตั้งเพิ่มเติมลงในโปรแกรม Eclipse

ภาพที่ 3.5 ปุ่มฟังก์ชั่นการทำงานของโปรแกรมเสริม

3.2.1 คุณสมบัติของโปรแกรมเสริม Javasnapshot

โปรแกรมเสริม Javasnapshot ที่พัฒนาขึ้นมีคุณสมบัติการทำงานดังต่อไปนี้

3.2.1.1 การกำหนดพฤติกรรมในการเก็บข้อมูล

การกำหนดค่าพื้นฐานและกำหนดพฤติกรรมของโปรแกรมเสริมในการเก็บข้อมูลระหว่างการพัฒนาโปรแกรมของผู้ทดสอบ สามารถกำหนดได้โดยกดที่ปุ่ม  ที่แถบเครื่องมือเพื่อกำหนดค่าต่าง ๆ ดังนี้

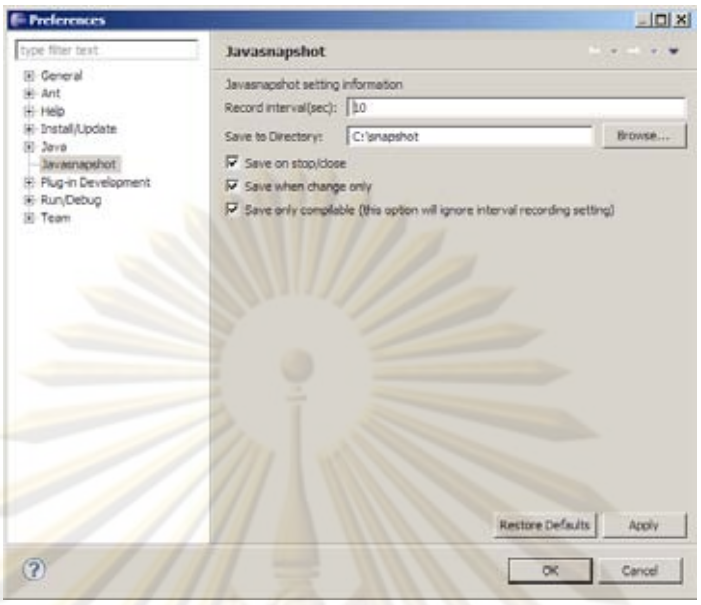
- กำหนดช่วงเวลาในการเก็บบันทึกข้อมูล โดยจะกำหนดในส่วนที่แสดงคำว่า Record interval (sec) ในหน้าจัดการ ดังภาพที่ 3.6 เพื่อให้บันทึกข้อมูลเก็บไว้เมื่อถึงทุก ๆ ช่วงเวลาที่กำหนด

- กำหนดตำแหน่งที่บันทึกข้อมูลเก็บไว้ โดยจะกำหนดในส่วนที่แสดงคำว่า Save to Directory ในหน้าจัดการ ดังภาพที่ 3.6 โดยระบบจะบันทึกข้อมูลเก็บไว้ในรูปไฟล์ โดยมีรูปแบบของชื่อไฟล์คือ ชื่อแบบทดสอบ ตามด้วยปี เดือน วัน และเวลาที่เก็บข้อมูล เช่น Lab1BMI-25521020-134720.java

- กำหนดให้บันทึกทุกครั้งเมื่อกดหยุดการทำงานหรือออกจากโปรแกรม Eclipse โดยจะกำหนดในส่วนที่แสดงคำว่า Save on stop/close ในหน้าจัดการ ดังภาพที่ 3.6 เพื่อบังคับให้บันทึกข้อมูลทุกครั้งที่หยุดการทำงาน หรือผู้ทดสอบลืมกดหยุดการทำงานและปิดโปรแกรมไป เพื่อให้บันทึกข้อมูลได้ครบถ้วนตั้งแต่เริ่มต้นจนกระทั่งผู้ทดสอบหยุดทำงาน

- กำหนดให้บันทึกเฉพาะเมื่อเกิดการเปลี่ยนแปลงของโปรแกรมเกิดขึ้น โดยจะกำหนดในส่วนที่แสดงคำว่า Save when change only ในหน้าจัดการ ดังภาพที่ 3.6 ซึ่งในกรณีที่เลือกตัวเลือกนี้ จะทำให้โปรแกรมจัดเก็บข้อมูลบันทึกรหัสต้นฉบับของโปรแกรมเฉพาะเมื่อมีการเปลี่ยนแปลงเกิดขึ้น ถึงแม้ว่าจะครบช่วงเวลาที่กำหนดไว้แล้วก็ตาม แต่ถ้าผู้ทดสอบไม่มีการทำงานใด ๆ ระหว่างช่วงเวลานั้น ก็จะไม่มีการบันทึกข้อมูลเพิ่มเติมเก็บไว้ เพราะข้อมูลจะซ้ำกับข้อมูลชุดที่แล้วที่เก็บเอาไว้

- กำหนดให้บันทึกเฉพาะรหัสต้นฉบับที่สามารถทำงานได้เท่านั้น โดยจะกำหนดในส่วนที่แสดงคำว่า Save only compilable (this option will ignore interval recording setting) ในหน้าจัดการ ดังภาพที่ 3.6



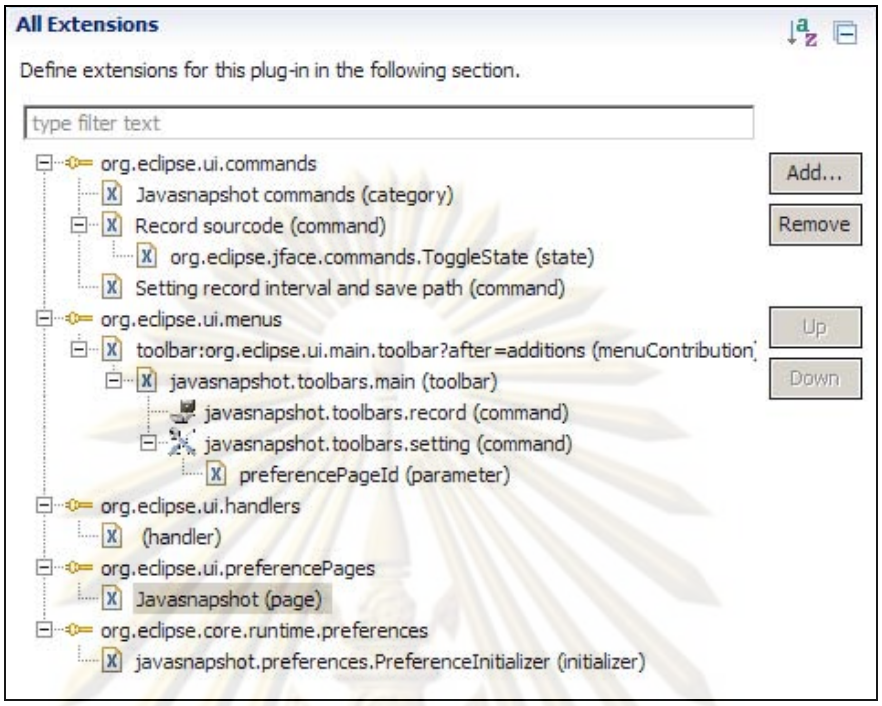
ภาพที่ 3.6 หน้าต่างกำหนดค่าการทำงานของโปรแกรมเสริม

3.2.1.2 การเริ่มต้นบันทึกและการหยุดบันทึกข้อมูล

ในการเริ่มต้นให้โปรแกรมเสริมที่ติดตั้งเพิ่มทำการบันทึกข้อมูลของผู้ทดสอบระหว่างการพัฒนาโปรแกรม ให้กดที่ปุ่ม  เพื่อเริ่มบันทึกและหลังจากที่กดแล้ว ปุ่มจะยุบลงไปมีลักษณะเป็น  ซึ่งจะแสดงว่าโปรแกรมบันทึกกำลังทำงานอยู่ หลังจากที่สิ้นสุดการทำงานแล้ว หรือผู้ทดสอบทำแบบทดสอบเสร็จแล้ว ก็ให้กดปุ่ม  อีกครั้งหนึ่ง เพื่อหยุดการบันทึกข้อมูล โดยหลังจากกดปุ่มแล้ว ปุ่มจะนูนขึ้นมาเหมือนตอนเริ่มต้นก่อนบันทึก คือ 

3.2.2 จุดเชื่อมต่อของโปรแกรม Eclipse ที่ใช้งาน

ในการพัฒนาโปรแกรมเสริม Javasnapshot ให้กับโปรแกรม Eclipse IDE เพื่อให้ทำงานได้ตามคุณสมบัติข้างต้น ได้ใช้จุดเชื่อมต่อของโปรแกรม Eclipse IDE ดังภาพที่ 3.7



ภาพที่ 3.7 จุดเชื่อมต่อที่มีการใช้งานของโปรแกรมเสริม Javasnapshot ที่พัฒนาขึ้น

จากภาพที่ 3.7 แสดงถึงจุดเชื่อมต่อที่เชื่อมเข้ากับโปรแกรม Eclipse IDE โดยมีรายละเอียดดังนี้

- จุดเชื่อมต่อ org.eclipse.ui.commands

ในส่วนนี้จะเป็นการประกาศคำสั่งของโปรแกรมเสริม Javasnapshot ที่มีให้เรียกใช้งานคือ การกำหนดพฤติกรรมการเก็บข้อมูล และการบันทึกข้อมูลระหว่างการพัฒนาโปรแกรม

- จุดเชื่อมต่อ org.eclipse.ui.menus

ในส่วนนี้จะเป็นการประกาศการเพิ่มเมนู การกำหนดพฤติกรรมการเก็บข้อมูล และการบันทึกข้อมูลระหว่างการพัฒนาโปรแกรม เข้ากับโปรแกรม Eclipse IDE ซึ่งแสดงดังภาพที่ 3.4 และ 3.5 และกำหนดคำสั่งในการทำงานให้กับเมนูแต่ละเมนู ตามคำสั่งที่ประกาศไว้ในจุดเชื่อมต่อ org.eclipse.ui.commands

- จุดเชื่อมต่อ org.eclipse.ui.handlers

ในส่วนนี้จะเป็นการประกาศการกำหนดพฤติกรรมที่เกิดขึ้นในส่วนหน้าจอการทำงาน (UI) ของโปรแกรม Eclipse IDE เพื่อดักจับพฤติกรรมที่ต้องการและให้เรียกใช้คำสั่งที่กำหนดไว้ทำงาน

- จุดเชื่อมต่อ org.eclipse.ui.preferencePages

ในส่วนนี้จะเป็นการประกาศการเพิ่มหน้าต่างการกำหนดพฤติกรรมในการเก็บข้อมูล ดังแสดงในภาพที่ 3.6 เข้ากับส่วนการกำหนดคุณสมบัติ (Preference) ของโปรแกรม Eclipse IDE

- จุดเชื่อมต่อ org.eclipse.core.runtime.preferences

ในส่วนนี้จะเป็นการประกาศการเพิ่มการดักจับพฤติกรรมเริ่มต้นทำงานของส่วนการกำหนดคุณสมบัติ (Preference) ของโปรแกรม Eclipse IDE เพื่ออ่านค่าข้อมูลของโปรแกรมเสริมที่กำหนดค่าเก็บไว้

3.2.3 คลาสของระบบที่ใช้ในการบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม

การทำงานของโปรแกรมเสริม Javasnapshot ในส่วนของการบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม ประกอบด้วยคลาสที่ใช้ทำงาน ซึ่งแบ่งออกเป็น package ต่าง ๆ ดังตารางที่ 3.1 – 3.6

ตารางที่ 3.1 คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot

Activator	ทำหน้าที่เป็นคลาสเริ่มต้นที่โปรแกรม Eclipse จะเรียกโปรแกรมเสริมให้ทำงาน
JavasnapshotLog	ทำหน้าที่เป็นคลาสที่โปรแกรม Eclipse จะเรียกใช้เมื่อมีการลงบันทึกหรือข้อผิดพลาดเกิดขึ้นในระหว่างการทำงานของโปรแกรมเสริม

ตารางที่ 3.2 คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.handlers

RecordHandler	ทำหน้าที่เป็นคลาสที่ดักจับพฤติกรรมในการทำงานของโปรแกรม Eclipse เพื่อส่งต่อให้กับส่วนของการบันทึกรหัสต้นฉบับของโปรแกรมขณะทำปฏิบัติการ
SettingHandler	ทำหน้าที่เป็นคลาสดักจับพฤติกรรมในการทำงานของ

	โปรแกรม Eclipse เพื่อส่งต่อให้กับส่วนของการกำหนดพฤติกรรมของโปรแกรมเสริม

ตารางที่ 3.3 คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.jobs

AbstractSnapshotJob	ทำหน้าที่เป็นคลาสแม่แบบของการพัฒนาคลาสที่ใช้บันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม
CompilableSnapshotJob	ทำหน้าที่เป็นคลาสที่ใช้ดักจับเหตุการณ์การเปลี่ยนแปลงที่เกิดขึ้นในเอกสารรหัสต้นฉบับของโปรแกรมที่กำลังพัฒนา นำมาตรวจเช็คความถูกต้องของไวยากรณ์ และบันทึกรหัสต้นฉบับของโปรแกรมเก็บไว้ เฉพาะการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่ถูกไวยากรณ์
SnapshotJob	ทำหน้าที่เป็นคลาสที่ใช้ดักจับเหตุการณ์การเปลี่ยนแปลงที่เกิดขึ้นในเอกสารรหัสต้นฉบับของโปรแกรมที่กำลังพัฒนา และบันทึกรหัสต้นฉบับของโปรแกรมเก็บไว้ตามช่วงเวลาที่กำหนด

ตารางที่ 3.4 คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.lexer

Java15Lexer	ทำหน้าที่เป็นคลาสที่ใช้ในการวิเคราะห์ศัพท์ของรหัสต้นฉบับของโปรแกรม ตามไวยากรณ์ภาษาจาวารุ่น 1.5
Java16Lexer	ทำหน้าที่เป็นคลาสที่ใช้ในการวิเคราะห์ศัพท์ของรหัสต้นฉบับของโปรแกรม ตามไวยากรณ์ภาษาจาวารุ่น 1.6

ตารางที่ 3.5 คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.parser

Java15Parser	ทำหน้าที่เป็นคลาสที่ใช้ในการแจงส่วน และตรวจสอบความถูกต้องของรหัสต้นฉบับของโปรแกรมตามไวยากรณ์ภาษาจาวารุ่น 1.5
Java16Parser	ทำหน้าที่เป็นคลาสที่ใช้ในการแจงส่วน และตรวจสอบความถูกต้องของรหัสต้นฉบับของโปรแกรมตามไวยากรณ์ภาษาจาวารุ่น 1.6

ตารางที่ 3.6 คลาสของโปรแกรมเสริม Javasnapshot ใน package javasnapshot.preferences

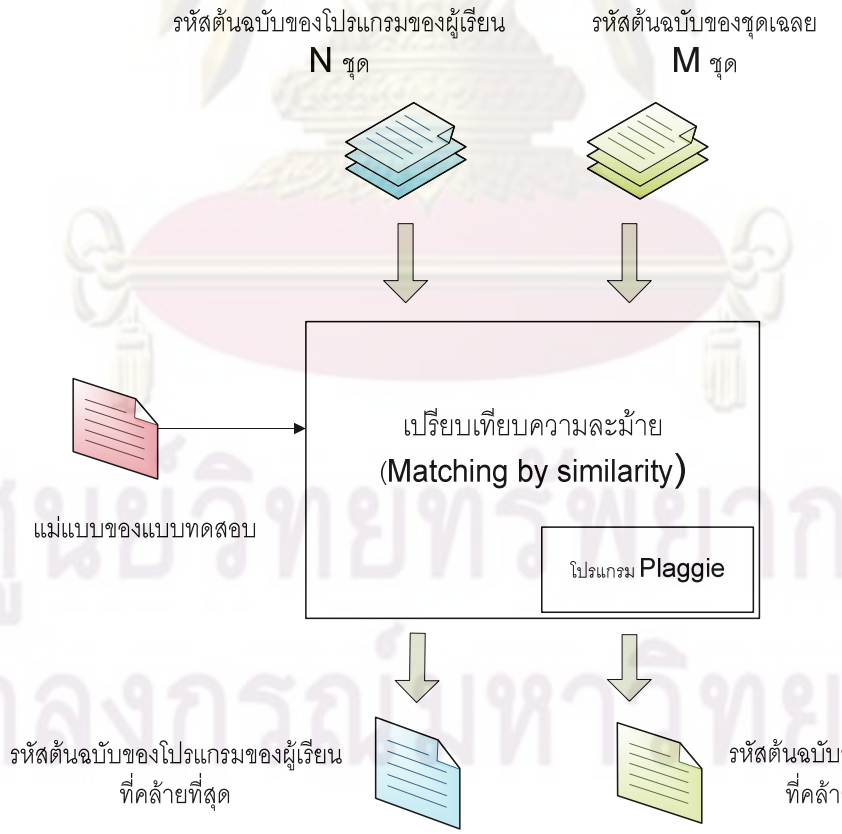
JavasnapshotPreferencePage	ทำหน้าที่เป็นคลาสที่ใช้ในการสร้างหน้าต่างที่ใช้ในการบันทึกการกำหนดพฤติกรรมในการทำงานของโปรแกรมเสริม ดัง

	แสดงในภาพที่ 3.6
PreferenceConstants	ทำหน้าที่เป็นคลาสที่เก็บค่าคงที่ที่ใช้อ้างอิงถึงพฤติกรรมต่าง ๆ ที่สามารถกำหนดได้ของโปรแกรมเสริม เช่น ระยะช่วงเวลา ตำแหน่งที่บันทึกข้อมูล เป็นต้น
PreferenceInitializer	ทำหน้าที่เป็นคลาสเริ่มต้นที่โปรแกรม Eclipse IDE จะเรียกใช้เมื่อเริ่มต้นการทำงานของระบบในส่วนของการกำหนดคุณสมบัติของโปรแกรม (Preference)

รายละเอียดของคลาสและการทำงาน ดูเพิ่มเติมได้ในภาคผนวก ข

3.3 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย

การพัฒนาเครื่องมือในการเปรียบเทียบความละม้ายระหว่างลำดับของรหัสต้นฉบับของโปรแกรมของผู้เรียนกับชุดเฉลย พัฒนาขึ้นด้วยโปรแกรมภาษาจาวา โดยมีลักษณะการทำงานดังภาพที่ 3.8



ภาพที่ 3.8 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย

ในการเปรียบเทียบความละม้ายได้ใช้โปรแกรม Plaggie เข้ามาช่วยในการเปรียบเทียบระหว่างลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย ค่าความละม้ายที่ได้จะไม่รวมเนื้อหาในส่วนแม่แบบของแบบทดสอบ ซึ่งการแยกเนื้อหาส่วนแม่แบบออกจากการเปรียบเทียบความละม้ายนี้เป็นคุณสมบัติหนึ่งของโปรแกรม Plaggie

3.3.1 การทดสอบโปรแกรม Plaggie

ในการทดสอบโปรแกรม Plaggie ผู้วิจัยได้ทดสอบผลกระทบของการเปลี่ยนแปลงรหัสต้นฉบับของโปรแกรมที่เพิ่มเข้าไปแล้วมีผลต่อเนื้อหาของรหัสต้นฉบับของโปรแกรม แต่ไม่มีผลต่อแนวทางการแก้ไขปัญหา เช่น การเพิ่มหมายเหตุ การเพิ่มบรรทัดว่าง การเว้นวรรค เป็นต้น ผลของความละม้ายที่ได้หลังจากที่เพิ่มการเปลี่ยนแปลงในรูปแบบต่าง ๆ แสดงดังตารางที่ 3.7

ตารางที่ 3.7 ผลการทดสอบผลของการเปลี่ยนแปลงรหัสต้นฉบับของโปรแกรมที่มีผลต่อเนื้อหาแต่ไม่มีผลต่อแนวทางการแก้ไขปัญหา

รูปแบบการทดสอบ	**ค่า % ความละม้ายที่ได้**
การเพิ่ม whitespace ตั้งแต่ 1 ตำแหน่งขึ้นไป	100
การเพิ่มหมายเหตุแบบ 1 บรรทัด “//” ตั้งแต่ 1 ตำแหน่งขึ้นไป	100
การเพิ่มหมายเหตุแบบหลายบรรทัด “/* */” ตั้งแต่ 1ตำแหน่งขึ้นไป	100
การเปลี่ยนชื่อตัวถูกดำเนินการ ตั้งแต่ 1 ตำแหน่งขึ้นไป	100
การสลับที่การบวก และลบ	100
การสลับที่การคูณ และหาร	100
การสลับที่การบวก ลบ คูณ และหาร	100
การเพิ่ม/ลด วงเล็บในการคำนวนตั้งแต่ 1 ตำแหน่งขึ้นไป	100

ผลของการทดสอบรูปแบบตามตารางที่ 3.7 ปรากฏว่าไม่มีผลต่อการเปรียบเทียบความละม้ายที่ได้ รหัสต้นฉบับของโปรแกรมก่อนการเปลี่ยนแปลงและหลังการเปลี่ยนแปลงตามลักษณะต่าง ๆ ที่ทดสอบให้ผลความละม้ายที่เหมือนกัน

ในการทดสอบโปรแกรม Plaggie เพื่อทดสอบผลของการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรม ได้ผลที่ดี ค่าความละม้ายที่เพิ่มขึ้นมีความสม่ำเสมอในทุก

ชุดทดสอบที่ได้ทำการทดสอบ ดังแสดงตัวอย่างการคำนวนค่าความละม้ายที่ได้ในแต่ละคำสั่งที่เพิ่มขึ้นตามตารางที่ 3.8 ดังนี้

ตารางที่ 3.8 ผลการทดสอบการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมด้วย Plaggie

ลำดับคำสั่ง	ชุดคำสั่งที่เขียน	% ความละม้าย
	public class Factorial {	
	public static void main(String[] args) {	
5	System.out.println(fac(3));	100%
	}	
1	public static int fac(int n){	43 %
2	if(n==0) return 1;	64 %
3	int f = fac(n-1);	79 %
4	return n*f;	86%
1	}	43 %
	}	

รหัสต้นฉบับของตัวอย่างในตารางที่ 3.8 เป็นรหัสต้นฉบับในการทำโปรแกรมการคำนวนแฟกทอเรียล ในส่วนพื้นสีเทาอ่อนจะแสดงส่วนของรหัสต้นฉบับขณะเริ่มต้นที่เริ่มทำการเปรียบเทียบความละม้าย ณ ขณะเริ่มต้นค่าเปอร์เซ็นต์ความละม้ายที่ได้มีค่าเท่ากับ 29 % โดยค่าเปอร์เซ็นต์ความละม้ายที่ได้เป็นค่าความละม้ายที่เปรียบเทียบกับรหัสต้นฉบับของโปรแกรมเมื่อเสร็จสมบูรณ์แล้ว ในที่นี้คือเมื่อพิมพ์ครบทุกคำสั่งที่แสดงในตารางที่ 3.8 เมื่อพัฒนาโปรแกรมโดยเพิ่มคำสั่งแต่ละคำสั่งเข้าไปตั้งแต่ลำดับที่ 1 จนถึงลำดับที่ 5 จนกระทั่งโปรแกรมเสร็จสมบูรณ์ ค่าความละม้ายที่ได้เป็นดังตารางที่ 3.8

ผลการทดสอบการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมด้วยโปรแกรม Plaggie ทั้งหมด ดูรายละเอียดได้ในภาคผนวก ค

3.3.2 คลาสของระบบในการเปรียบเทียบความละม้าย

การทำงานในการเปรียบเทียบความละม้ายดังภาพที่ 3.8 เมื่อนำมาพัฒนาเป็นระบบในการเปรียบเทียบความละม้าย ประกอบด้วยคลาสของการทำงานของโปรแกรมในส่วนการเปรียบเทียบความละม้าย ซึ่งแบ่งออกเป็น package ต่าง ๆ ดังตารางที่ 3.9 – 3.10

ตารางที่ 3.9 คลาสของระบบในส่วนการเปรียบเทียบความละม้าย ใน package javasnapshot.analyze

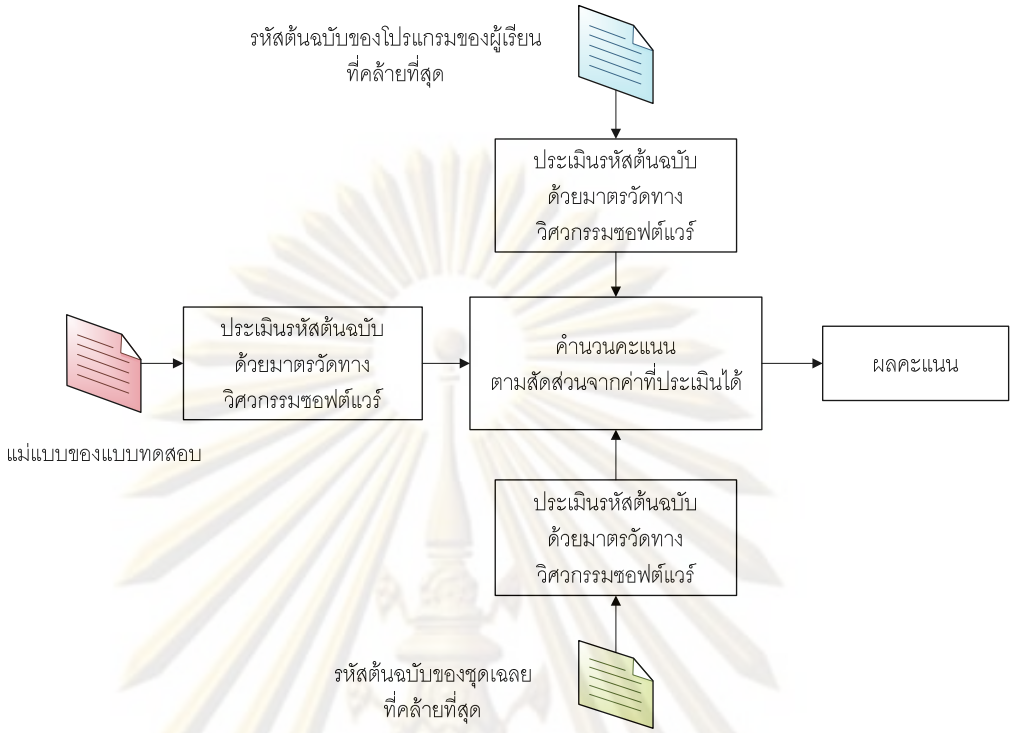
SimilarityAnalyzer	ทำหน้าที่เป็นคลาสหลักที่ใช้ในการเปรียบเทียบระหว่างลำดับของรหัสต้นฉบับของโปรแกรม กับชุดเฉลย และให้ผลลัพธ์กลับมาเป็นคลาส SimilarityResult ซึ่งเก็บผลลัพธ์ในการเปรียบเทียบของคู่ของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายที่สุด พร้อมกับค่าความละม้ายที่คำนวนได้
SimilarityResult	ทำหน้าที่เป็นคลาสที่เก็บผลลัพธ์ในการเปรียบเทียบของคู่ของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายที่สุด พร้อมกับค่าความละม้ายที่คำนวนได้

ตารางที่ 3.10 คลาสของระบบในส่วนการเปรียบเทียบความละม้าย ใน package javasnapshot.analyze.engine

ISimilarityEngine	ทำหน้าที่กำหนดส่วนต่อประสานในการทำงานของการเปรียบเทียบความละม้าย
PlaggieSimEngine	ทำหน้าที่เป็นคลาสที่ใช้ในการเปรียบเทียบความละม้ายโดยใช้โปรแกรม Plaggie

3.4 การวิเคราะห์รหัสต้นฉบับของโปรแกรมกับชุดเฉลยเพื่อให้คะแนน

การพัฒนาเครื่องมือในการวิเคราะห์รหัสต้นฉบับของโปรแกรมกับชุดเฉลยเพื่อให้คะแนนพัฒนาขึ้นด้วยโปรแกรมภาษาจาวา โดยมีลักษณะการทำงานดังภาพที่ 3.9



ภาพที่ 3.9 การวิเคราะห์รหัสต้นฉบับของโปรแกรมกับชุดเฉลยเพื่อให้คะแนน

ในการวิเคราะห์รหัสต้นฉบับเพื่อประเมินด้วยมาตรวัดทางวิศวกรรมซอฟต์แวร์ ใช้มาตรวัดค่าความพยายามของฮอลสตีดมาประเมินรหัสต้นฉบับของโปรแกรมของผู้เรียน ชุดเฉลย และแม่แบบของแบบทดสอบ โดยใช้ไลบรารีของ ANTLR มาช่วยในการวิเคราะห์ศัพท์ (Lexical analysis) และแจงส่วน (Parsing) เพื่อตัดคำจากรหัสต้นฉบับออกเป็นส่วน ๆ ตามประเภทของโทเค็นดังตารางที่ 2.4 เพื่อนับจำนวนตัวดำเนินการและตัวถูกดำเนินการ เพื่อใช้ในการคำนวนตามมาตรวัดของฮอลสตีด ผลการทดสอบการคำนวนค่าความพยายามตามมาตรวัดของฮอลสตีด ดูรายละเอียดได้ในภาคผนวก ฉ

3.4.1 คลาสของระบบในการคำนวนให้คะแนน

การทำงานในการคำนวนให้คะแนนดังภาพที่ 3.9 เมื่อนำมาพัฒนาเป็นระบบการให้คะแนน ประกอบด้วยคลาสของการทำงานของโปรแกรมในส่วนการคำนวนให้คะแนน ซึ่งแบ่งออกเป็น package ต่าง ๆ ดังตารางที่ 3.11 – 3.12

ตารางที่ 3.11 คลาสของระบบในส่วนการคำนวนให้คะแนน ใน package javasnapshot.mark

Marker	ทำหน้าที่เป็นคลาสหลักในการคำนวนให้คะแนน โดยการนำลำดับของรหัสต้นฉบับ และชุดเฉลย มาหารหัสต้นฉบับของ

	โปรแกรมและชุดเฉลยที่ละม้ายที่สุดด้วยโปรแกรมเปรียบเทียบความละม้ายที่พัฒนาขึ้นมา และนำรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายมาประเมินด้วยมาตรวัดทางวิศวกรรมซอฟต์แวร์และคำนวนให้คะแนน

ตารางที่ 3.12 คลาสของระบบในส่วนการคำนวนให้คะแนน ใน package javasnapshot.estimate

IEstimator	ทำหน้าที่กำหนดส่วนต่อประสานในการทำงานของการประเมินรหัสต้นฉบับของโปรแกรม
HalsteadEffortEstimator	ทำหน้าที่เป็นคลาสที่ใช้ในการประเมินรหัสต้นฉบับของโปรแกรมตามค่าความพยายามตามมาตรวัดของฮอลสตีด

3.4.2 สมการการคำนวนให้คะแนน

การคำนวนให้คะแนน โดยการนำรหัสต้นฉบับของโปรแกรม รหัสต้นฉบับของชุดเฉลย และรหัสต้นฉบับของแม่แบบของแบบทดสอบ มาประเมินด้วยมาตรวัดของฮอลสตีด และให้คะแนนตามสูตรการคำนวนดังนี้

$$P = \alpha \cdot S + (1 - \alpha) \cdot \mathrm{E}$$

โดยที่ P เป็นเปอร์เซ็นต์ของคะแนนที่ได้

$\propto$ คือ ความน่าจะเป็น หรือโอกาสที่จะเกิด (probability)

S คือ ค่าเปอร์เซ็นต์ความละม้าย

E คือ ค่าเปอร์เซ็นต์ของความพยายามที่คำนวนได้

โดยที่ค่าเปอร์เซ็นต์ของความพยายาม คำนวนได้จาก

$$E = 1 - \frac{|E(x) - E(s)|}{Max(E(s), E(x))}$$

โดยที่ E เป็นเปอร์เซ็นต์ของความพยายามที่คำนวนได้

E(x) เป็นค่าความพยายามของฮอลสตีดที่คำนวนได้จากรหัสต้นฉบับของผู้เรียนที่ละม้ายที่สุด (snapshot)

E(s) เป็นค่าความพยายามของฮอลสตีดที่คำนวนได้จาก
ชุดเฉลยที่ละม้ายที่สุด (solution)

ผลของการคำนวนให้คะแนน ดูได้จากผลการทดลองในบทที่ 4

3.5 กลุ่มประชากร

กลุ่มประชากรของผู้ที่ทำการทดสอบ จะเป็นผู้ที่เคยผ่านการเรียนรู้โปรแกรมภาษาจาวามาก่อน ซึ่งได้แก่ กลุ่มผู้เรียนที่ผ่านการเรียนวิชาการทำโปรแกรมคอมพิวเตอร์ (2110101) ซึ่งเป็นวิชาเรียนของนิสิตชั้นปีที่ 1 ของคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 หรือกลุ่มผู้ที่เรียนการเขียนโปรแกรมภาษาจาวาหรือใช้งานภาษาจาวามาก่อน

3.6 การเก็บรวบรวมข้อมูล

การเก็บรวบรวมข้อมูล ผู้วิจัยเลือกแบบทดสอบจากแบบฝึกหัดในการทำปฏิบัติการ (lab) ของวิชาการทำโปรแกรมคอมพิวเตอร์ (2110101) ของคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปี พ.ศ. 2552 มาใช้ทำการทดสอบจำนวน 5 แบบทดสอบ โดยแบ่งระดับความยากง่ายของโจทย์ที่ใช้ทดสอบตามลำดับขั้นของการเรียนรู้ของบลูม (Bloom Taxonomy) [16] ได้เป็นระดับความรู้ (Knowledge) 1 ข้อ ระดับความเข้าใจ (Comprehension) 3 ข้อ และระดับการนำไปใช้ (Application) 1 ข้อ ซึ่งผู้เข้าทดสอบจะดำเนินการทดสอบตามลำดับขั้นตอนดังนี้

1. ผู้เข้าทดสอบจะได้รับโจทย์ทดสอบ 5 ข้อ และอ่านทำความเข้าใจโจทย์ก่อนเริ่มทำ

2. ผู้เข้าทดสอบเริ่มทำโจทย์โดยใช้โปรแกรม Eclipse ที่ติดตั้งโปรแกรมเสริมเพื่อใช้เก็บข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม โดยกดเริ่มบันทึกเมื่อเริ่มทำโจทย์แต่ละข้อ และกดหยุดการบันทึกเมื่อทำเสร็จหรือหมดเวลาในแต่ละข้อ และกดเริ่มบันทึกเมื่อเริ่มข้อต่อไป ทำแบบนี้ไปจนผู้เข้าทดสอบทำโจทย์ครบทุกข้อ ในกรณีที่ผู้เข้าทดสอบยังไม่เคยชินกับโปรแกรม Eclipse ผู้วิจัยจะอธิบายการใช้งานเบื้องต้นที่จำเป็นในการใช้ทำโจทย์ต่อผู้เข้าทดสอบ

3. ผู้เข้าทดสอบทำแบบทดสอบแต่ละข้อ โดยมีอิสระในการเลือกที่จะทำโจทย์ข้อใดก่อนก็ได้ การทำโจทย์แต่ละข้อผู้เข้าทดสอบจะถูกจับเวลาที่ใช้ตามที่โจทย์กำหนด หลังจากหมดเวลาในแต่ละข้อ ถ้ายังทำไม่เสร็จและคิดว่าสามารถที่จะทำเสร็จได้ ก็จะปล่อยให้ทำต่อไปจนเสร็จ

4. หลังจากที่ผู้เข้าทดสอบทำแบบทดสอบเสร็จแล้ว ผู้เข้าทดสอบจะต้องส่งชุดของรหัสต้นฉบับของโปรแกรม ที่เกิดการเปลี่ยนแปลงทั้งหมดระหว่างการทำโปรแกรมของผู้เข้าทดสอบ ที่ถูกเก็บบันทึกลงไฟล์ แยกเป็นแต่ละครั้งของการเปลี่ยนแปลงที่เกิดขึ้นตามลำดับไว้ให้กับผู้วิจัย เพื่อนำไปวิเคราะห์ข้อมูลต่อไป

ผู้เรียน / ผู้ทดสอบ

รหัสต้นฉบับของโปรแกรมของผู้เรียน N ชุด

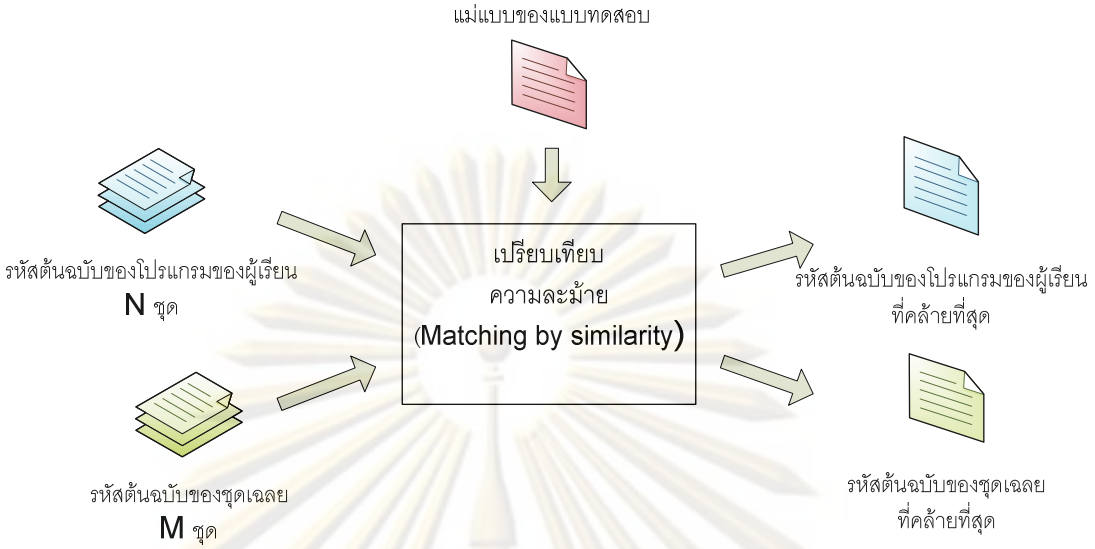
ภาพที่ 3.10 ลำดับการเปลี่ยนแปลงของโปรแกรมที่บันทึกได้ หลังจากเสร็จสิ้นการทดสอบ

3.7 การวิเคราะห์ข้อมูล

การวิเคราะห์ข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่ผู้เรียนทำแบบทดสอบ หลังจากที่สิ้นสุดการทำแบบทดสอบในแต่ละข้อแล้ว จะนำข้อมูลที่บันทึกไว้มาวิเคราะห์ตามลำดับขั้นตอนดังนี้

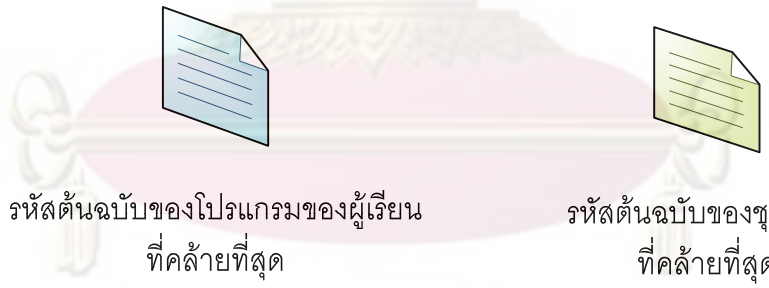
1. วิเคราะห์ความละม้ายของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย

หลังจากที่เก็บข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่ผู้ทดสอบทำการทดสอบในแต่ละข้อแล้ว จะนำชุดข้อมูลที่บันทึกได้มาทำการวิเคราะห์ความละม้ายของรหัสต้นฉบับของโปรแกรม โดยใช้โปรแกรม Plaggie ช่วยในการวิเคราะห์หาค่าความละม้ายของรหัสต้นฉบับของโปรแกรมของผู้เรียนเทียบกับชุดเฉลยที่มี สำหรับแบบทดสอบในแต่ละข้อ เพื่อหารหัสต้นฉบับของโปรแกรมที่เปรียบเทียบแล้วดีที่สุด ออกมาจากชุดของรหัสต้นฉบับของโปรแกรมทั้งหมดที่เก็บบันทึกได้



ภาพที่ 3.11 การวิเคราะห์ความละม้ายของลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่บันทึกได้ กับชุดเฉลย

ขั้นตอนในการเปรียบเทียบความละม้าย เป็นการนำรหัสต้นฉบับของผู้เรียน n ชุด มาเปรียบเทียบ กับชุดเฉลย m ชุด ที่มี จำนวนครั้งในการเปรียบเทียบจะเท่ากับ n x m เพื่อหาคู่ของรหัสต้นฉบับของผู้เรียนกับชุดเฉลยที่ละม้ายกันที่สุด และนำไปวิเคราะห์ให้คะแนนในขั้นต่อไป



ภาพที่ 3.12 ผลการเปรียบเทียบความละม้ายได้รหัสต้นฉบับของโปรแกรม และชุดเฉลยชุดที่ละม้ายกันมากที่สุด เพื่อนำไปคำนวนให้คะแนน

2. วิเคราะห์รหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่ละม้ายมากที่สุดเพื่อให้คะแนน

ในการวิเคราะห์เพื่อให้คะแนน จะใช้เครื่องมือที่สร้างไว้มาประเมินรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายมากที่สุด ได้เป็นค่าตัวเลขเปอร์เซ็นต์ของคะแนนที่ได้รับ

3.8 การประเมินผลคะแนน

การประเมินผลคะแนนที่คำนวนได้ อาจารย์ผู้สอนหรือผู้ทรงคุณวุฒิจะเป็นผู้ประเมินคะแนนที่ได้จากรหัสต้นฉบับของโปรแกรมของผู้เรียนที่ดีที่สุดเมื่อเทียบกับชุดเฉลยที่คล้ายมากที่สุดว่า ผลคะแนนที่ได้เหมาะสมหรือไม่

บทที่ 4

ผลการวิเคราะห์ข้อมูล

ผลการวิเคราะห์ข้อมูลจะประกอบด้วย ผลการวิเคราะห์การเก็บบันทึกข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมระหว่างทำปฏิบัติการ ผลการเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลยเพื่อหาคู่ของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายมากที่สุดโดยใช้โปรแกรม Plaggie และผลการคำนวนค่าความพยายามตามทฤษฎีของฮอลสตีดจากคู่ของรหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่ละม้ายมากที่สุดเพื่อให้คะแนน

ผลการวิเคราะห์ข้อมูลจะวิเคราะห์จากชุดของลำดับต้นฉบับของโปรแกรมทั้งหมด 70 ชุด จากกลุ่มตัวอย่าง 16 คน ซึ่งประกอบด้วย ผู้ที่กำลังศึกษาระดับปริญญาตรี 6 คน ซึ่งแบ่งเป็นนิสิตจุฬาลงกรณ์มหาวิทยาลัย 3 คน และนักศึกษามหาวิทยาลัยรังสิต 3 คน กำลังศึกษาระดับปริญญาโท 3 คน กำลังศึกษาระดับปริญญาเอก 2 คน และผู้ที่ทำงานแล้ว 5 คน ผู้ทดสอบทุกคนผ่านการเรียนวิชาการทำโปรแกรมคอมพิวเตอร์ หรือวิชาที่มีเนื้อหาเกี่ยวกับการพัฒนาโปรแกรมคอมพิวเตอร์มาแล้ว ผลการวิเคราะห์ข้อมูลที่ได้ในแต่ละส่วน เป็นดังนี้

4.1 การวิเคราะห์ข้อมูล

4.1.1 ผลการเก็บบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม

การบันทึกข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม มีรูปแบบการบันทึกการเปลี่ยนแปลง ดังนี้

4.1.1.1 การบันทึกตามช่วงเวลา

การบันทึกตามช่วงเวลา จะบันทึกรหัสต้นฉบับของโปรแกรมระหว่างที่ผู้ปฏิบัติทำการทดสอบเก็บไว้ทุก ๆ ช่วงเวลา เช่น ทุก ๆ 5 วินาที หรือ 10 วินาที ตัวอย่างผลการบันทึกที่ได้เป็นดัง ภาพที่ 4.1 - 4.4 ดังนี้

```
public class TestMult
```

ภาพที่ 4.1 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 1

```
public class TestMultiply {
        public static void mai
}
```

ภาพที่ 4.2 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 2

```
public class TestMultiply {
        public static void main(String[] args){
                System.out.prin
        }
}
```

ภาพที่ 4.3 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 3

```
public class TestMultiply {
        public static void main(String[] args){
                System.out.println(4*5);
        }
}
```

ภาพที่ 4.4 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 4

จากตัวอย่างดังภาพที่ 4.1 – 4.4 พบว่าข้อมูลที่บันทึกได้แสดงความคิดของผู้ปฏิบัติได้ในระดับหนึ่ง แต่ในบางลำดับจะแสดงได้ไม่ชัดเจนว่าผู้ปฏิบัติต้องการจะทำอะไร เช่น ในภาพที่ 4.3 จะไม่รู้ว่าผู้ปฏิบัติกำลังจะสั่งพิมพ์อะไร ผู้ปฏิบัติต้องการจะสั่งพิมพ์ในบรรทัดเดิม ซึ่งจะใช้คำสั่ง print() หรือสั่งพิมพ์แล้วขึ้นบรรทัดใหม่ ซึ่งจะใช้คำสั่ง println() จนกระทั่งถึงภาพที่ 4.4 จึงจะทราบว่าผู้ปฏิบัติคิดอะไรและต้องการจะเขียนอะไร ซึ่งจากรหัสต้นฉบับของโปรแกรมที่บันทึกได้ตามช่วงเวลานี้ จะได้ข้อมูลที่มีทั้งที่แสดงถึงสิ่งที่ผู้ปฏิบัติกำลังกระทำได้อย่างชัดเจน และที่แสดงได้ไม่ชัดเจน

การปรับเปลี่ยนช่วงเวลาให้สั้นลงหรือยาวขึ้น ให้ผลลัพธ์ที่ไม่แตกต่างกัน ผลที่ได้สรุปได้ว่า ลักษณะของข้อมูลที่บันทึกได้ นอกจากจะขึ้นอยู่กับช่วงเวลาที่จัดเก็บแล้ว ยังขึ้นอยู่กับลักษณะการทำงานของผู้ปฏิบัติด้วย สำหรับผู้ปฏิบัติที่ถนัด ก็จะพิมพ์เร็ว ผู้ที่ไม่ค่อยถนัดหรือไม่เข้าใจ ก็จะพิมพ์ได้ช้า หรือมีการเว้นช่วงห่างระหว่างใช้ความคิดไม่เท่ากัน การบันทึกตามช่วงเวลาจึงไม่เหมาะสมสำหรับนำไปใช้กับกลุ่มคนจำนวนมาก และไม่เหมาะสมที่จะนำมาใช้แสดงในแต่ละขณะว่า ผู้ปฏิบัติกำลังตั้งใจจะทำอะไร

4.1.1.2 **การบันทึกรายตัวอักษร**

การบันทึกรายตัวอักษร จะบันทึกรหัสต้นฉบับของโปรแกรมระหว่างที่ผู้ปฏิบัติทำการทดสอบเก็บไว้ในทุก ๆ ครั้งที่มีการกดพิมพ์ตัวอักษรลงไปแต่ละตัว แก้ไข หรือลบออก ผลที่ได้สรุปได้ว่า ผลการบันทึกมีลักษณะคล้ายกับการบันทึกตามช่วงเวลา แต่จะมีจำนวนลำดับของรหัสต้นฉบับของโปรแกรมที่บันทึกได้จำนวนมาก ซึ่งประกอบด้วยรหัสต้นฉบับของโปรแกรมที่แสดงได้อย่างชัดเจนว่าผู้ปฏิบัติกำลังต้องการจะทำอะไร และแสดงได้ไม่ชัดเจน การบันทึกรายตัวอักษรจึงไม่เหมาะสมสำหรับนำไปใช้กับกลุ่มคนจำนวนมาก และผลเสียของจำนวนของรหัสต้นฉบับของโปรแกรมที่บันทึกได้มากขึ้นจะเพิ่มความยุ่งยากในการเลือกข้อมูลเพื่อนำไปเปรียบเทียบความละม้ายของรหัสต้นฉบับกับชุดเฉลยในขั้นต่อไป

4.1.1.3 **การบันทึกรายคำสั่ง**

การบันทึกรายคำสั่ง จะบันทึกรหัสต้นฉบับของโปรแกรมระหว่างที่ผู้ปฏิบัติทำการทดสอบเก็บไว้ในทุก ๆ รายคำสั่งที่ผู้ปฏิบัติพิมพ์คำสั่งถูกต้อง ตัวอย่างผลการบันทึกที่ได้เป็นดังภาพที่ 4.5 - 4.7 ดังนี้

```
public class TestMultiply{
}
```

ภาพที่ 4.5 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 1

```
public class TestMultiply {
    public static void main(String[] args){
    }
}
```

ภาพที่ 4.6 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 2

```
public class TestMultiply {
        public static void main(String[] args){
                System.out.println(4*5);
        }
}
```

ภาพที่ 4.7 รหัสต้นฉบับของโปรแกรมที่บันทึกได้ลำดับที่ 3

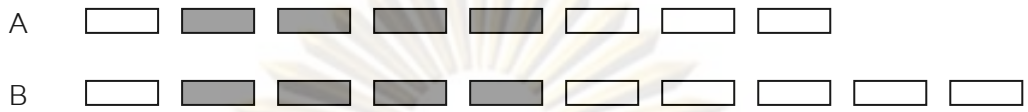
จากตัวอย่างดังภาพที่ 4.5 – 4.7 พบว่าในแต่ละลำดับของการเปลี่ยนแปลงของโปรแกรมที่บันทึกได้ แสดงถึงความตั้งใจของผู้ปฏิบัติได้อย่างครบถ้วน ซึ่งเมื่อเทียบกับการบันทึกตามช่วงเวลาในภาพที่ 4.3 และ 4.4 การบันทึกรหัสต้นฉบับของโปรแกรมในภาพที่ 4.4 ก็เพียงพอที่จะแทนความตั้งใจของผู้ปฏิบัติในภาพที่ 4.3 ได้ ดังนั้นการเปลี่ยนแปลงขณะทำปฏิบัติการในลักษณะที่ยังพิมพ์คำสั่งไม่ครบ เช่น ไม่ได้พิมพ์เครื่องหมาย “ ; ” เพื่อจบคำสั่งไม่ควรบันทึกรหัสต้นฉบับของโปรแกรมเก็บไว้ เพราะจะได้รหัสต้นฉบับของโปรแกรมที่แสดงความตั้งใจของผู้ปฏิบัติได้ไม่ชัดเจน

โดยสรุปการพัฒนาระบบเพื่อบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม จึงควรบันทึกรหัสต้นฉบับของโปรแกรมทุกครั้ง เมื่อผู้ปฏิบัติพิมพ์คำสั่งแต่ละคำสั่งเสร็จสิ้นแล้ว ข้อมูลที่บันทึกได้จะแสดงถึงความตั้งใจที่ชัดเจนในแต่ละคำสั่งว่าผู้ปฏิบัติต้องการจะทำอะไร ข้อมูลที่บันทึกได้มีปริมาณไม่มากเมื่อเทียบกับการบันทึกตามช่วงเวลา หรือเทียบกับการบันทึกรายตัวอักษร จำนวนและข้อมูลลำดับของรหัสต้นฉบับของโปรแกรมที่บันทึกได้เพียงพอและเหมาะสมที่จะนำไปใช้ในการเปรียบเทียบความละม้ายกับชุดเฉลยในขั้นตอนต่อไป

4.1.2 ผลการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย

ในการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมที่บันทึกไว้โดยกลุ่มตัวอย่างที่ได้รับแบบทดสอบ 5 ข้อ แบ่งตามระดับความยากง่ายตามระดับการเรียนรู้ของบลูม (Bloom Taxanomy) [13] เป็น ระดับความรู้ (knowledge) สำหรับแบบทดสอบชุดที่ 1 ระดับความเข้าใจ (comprehension) สำหรับแบบทดสอบชุดที่ 2 ถึง 4 และระดับการนำไปใช้ สำหรับแบบทดสอบชุดที่ 5 รายละเอียดของแบบทดสอบและชุดเฉลยดูได้ในภาคผนวก ก

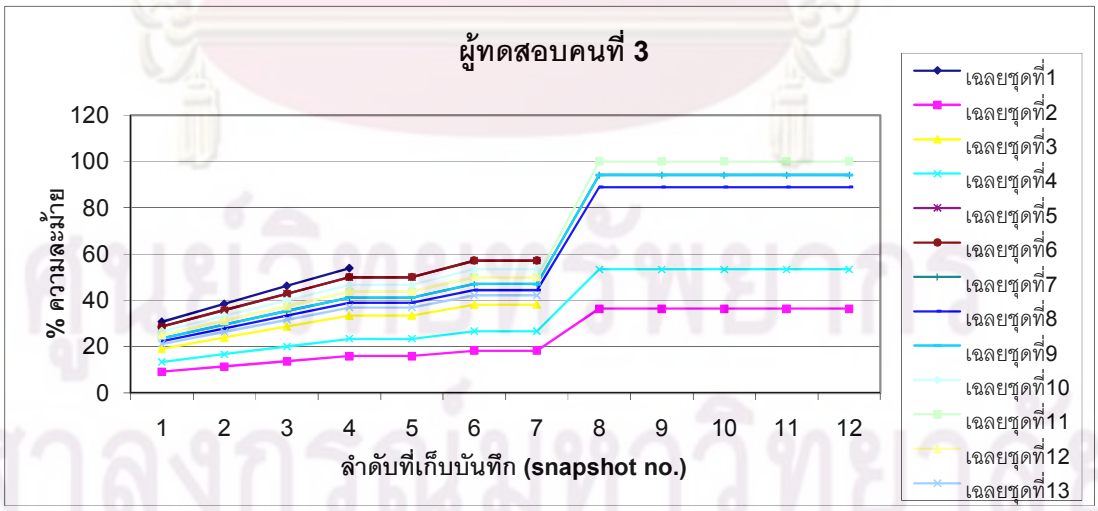
การเปรียบเทียบความละม้ายโดยใช้ Plaggie ผลที่ได้จากการเปรียบเทียบความละม้ายระหว่างรหัสต้นฉบับของโปรแกรม กับชุดเฉลย จะได้ค่าความละม้าย 2 ค่า คือ ค่าความละม้ายของรหัสต้นฉบับของโปรแกรมเมื่อเทียบกับชุดเฉลย (solution base) และ ค่าความละม้ายของชุดเฉลยเมื่อเทียบกับรหัสต้นฉบับของโปรแกรม (snapshot base) ดังแสดงในภาพที่ 4.8



ภาพที่ 4.8 ส่วนของความละม้ายระหว่างรหัสต้นฉบับของโปรแกรมกับชุดเฉลย

จากภาพที่ 4.8 กำหนดให้ A แทนรหัสต้นฉบับของโปรแกรมของผู้เรียน และ B แทนรหัสต้นฉบับของชุดเฉลย พื้นที่สีเทาแสดงถึงส่วนที่ละม้ายกันระหว่าง A กับ B ค่าความละม้ายของรหัสต้นฉบับของโปรแกรมเมื่อเทียบกับชุดเฉลย (solution base) คือ จำนวนโทเค็นของ A ที่ละม้าย เทียบกับจำนวนโทเค็นทั้งหมดใน B ซึ่งจากภาพที่ 4.8 จะได้เท่ากับ 4/10 และค่าความละม้ายของชุดเฉลยเมื่อเทียบกับรหัสต้นฉบับของโปรแกรม (snapshot base) คือ จำนวนโทเค็นของ B ที่ละม้าย เทียบกับจำนวนโทเค็นทั้งหมดใน A ซึ่งจากภาพที่ 4.8 จะได้เท่ากับ 4/8

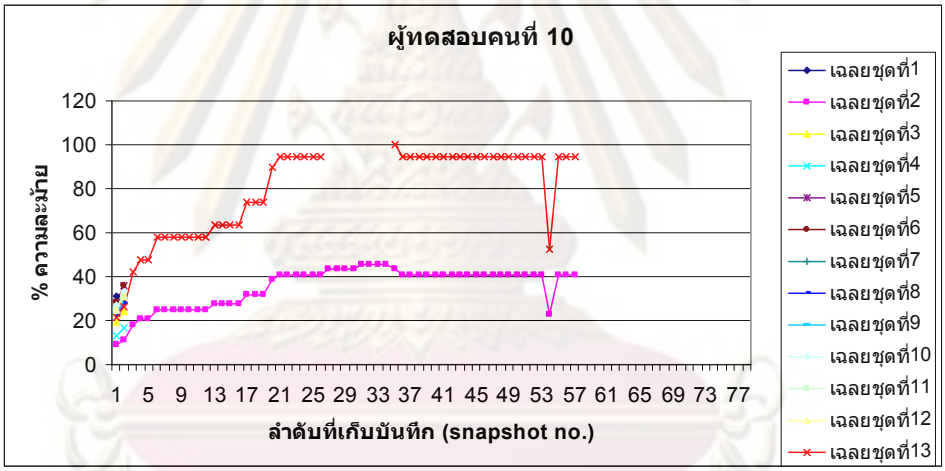
เมื่อนำลำดับของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบที่ทำการทดสอบ กับชุดเฉลยมาเปรียบเทียบแยกเป็นรายแบบทดสอบ เช่น รหัสต้นฉบับของผู้ทดสอบคนที่ 3 และนำมาวาดกราฟ ตัวอย่างของกราฟที่ได้มีลักษณะดังภาพที่ 4.9



ภาพที่ 4.9 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบคนที่ 3 กับชุดเฉลยทั้งหมดของแบบทดสอบที่ 1

กราฟข้างต้นแสดงผลการเปรียบเทียบความละม้าย โดยแกนนอนแสดงลำดับที่เกิดการบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม และแกนตั้งแสดงค่าเปอร์เซ็นต์ความละม้าย ผลที่ได้เมื่อนำรหัสต้นฉบับของโปรแกรมในแต่ละลำดับมาเทียบกับชุดเฉลยแต่ละชุดแสดงให้เห็นถึงวิธีคิด หรือเส้นทางของชุดเฉลยที่ผู้ทดสอบใช้ในการแก้ปัญหา จากตัวอย่าง ผู้ทดสอบทำแบบทดสอบตามแนวทางของเฉลยชุดที่ 11 ซึ่งแสดงจาก ค่าเปอร์เซ็นต์ความละม้ายที่เพิ่มขึ้นจนเป็นร้อยเปอร์เซ็นต์

เมื่อตรวจสอบผลของการตรวจตามลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่บันทึกได้ พบว่า ณ ลำดับที่บันทึกสุดท้ายในบางการทดสอบของผู้ทดสอบไม่ใช่งานที่ดีที่สุด หรือไม่ใช่งานที่ทำได้ถูกต้องตรงตามที่โจทย์กำหนด เช่น ในการทำแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 10 ผลการเปรียบเทียบความละม้ายปรากฏตามภาพที่ 4.10



ภาพที่ 4.10 การเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบคนที่ 10 กับชุดเฉลยของแบบทดสอบที่ 1

จากภาพที่ 4.10 ค่าเปอร์เซ็นต์ความละม้ายจากกราฟแสดงให้เห็นว่า งานสุดท้ายของผู้ทดสอบคือ ลำดับที่ 78 ที่ผู้ทดสอบคนที่ 10 ส่ง ไม่ใช่งานที่ได้ความละม้ายมากที่สุด แต่เป็นลำดับที่ 35 ที่ได้ค่าความละม้ายเท่ากับร้อยเปอร์เซ็นต์ เมื่อตรวจสอบรหัสต้นฉบับของโปรแกรมในลำดับที่ 35 และ 78ปรากฏเนื้อหาของรหัสต้นฉบับของโปรแกรม ดังนี้

```
import java.io.*;
public class Lab1BMI {
	public static void main(String[] args) throws Exception{
		//Please fill in commands here
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		System.out.println("weight (kg.) : ");
		double w=Double.parseDouble(br.readLine());
		System.out.println("heigth (kg.) : ");
		double h=Double.parseDouble(br.readLine());
		System.out.println(w/Math.pow(h, 2));
	}
}
```

ภาพที่ 4.11 รหัสต้นฉบับของโปรแกรมลำดับที่ 35 ของผู้ทดสอบคนที่ 10 (แบบทดสอบที่ 1)

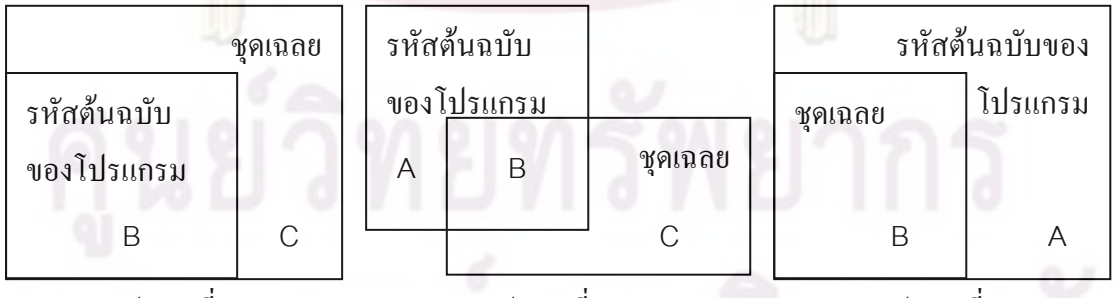
```
import java.io.*;
public class Lab1BMI {
	public static void main(String[] args) throws Exception{
		//Please fill in commands here
		BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
		System.out.println("weight (kg.) : ");
		int w=Integer.parseInt(br.readLine());
		System.out.println("heigth (kg.) : ");
		int h=Integer.parseInt(br.readLine());
		double ibm=w/Math.pow(h, 2);

		if(ibm<18.5)System.out.println("u r thin");
		if(ibm>=25)System.out.println("u r flat");
		if(ibm>=18.5&&ibm<=25)System.out.println("u r smart");
	}
}
```

ภาพที่ 4.12 รหัสต้นฉบับของโปรแกรมลำดับที่ 78 ของผู้ทดสอบคนที่ 10 (แบบทดสอบที่ 1)

เมื่อตรวจดูรหัสต้นฉบับของโปรแกรมของผู้ทดสอบคนที่ 10 ในลำดับที่ 35 และ 78 พบว่า ผู้ทดสอบทำเสร็จตั้งแต่ลำดับที่ 35 ซึ่งแสดงว่าผู้ทดสอบมีความเข้าใจ สามารถแก้ไขปัญหาของโจทย์ได้ แต่ได้ทำต่อเพิ่มเติมเกินกว่าที่โจทย์กำหนด ทำให้ค่าเปอร์เซ็นต์ความละม้ายลดลง ถ้าดูแต่ผลลัพธ์สุดท้ายระบบตรวจข้อสอบอัตโนมัติจะตรวจงานชิ้นนี้ไม่ผ่าน เพราะว่าผลจากการทำงานของโปรแกรมได้ผลไม่ตรงตามรูปแบบที่โปรแกรมตรวจสอบตรวจ แต่ถ้าดูในลำดับการเปลี่ยนแปลงของโปรแกรมและเลือกลำดับที่ละม้ายที่สุดออกมา โปรแกรมตรวจผลลัพธ์จะตรวจผ่าน เพราะว่าผลที่ได้จากการทำงานของลำดับที่ 35 จะได้ผลตรงกัน ซึ่งทำให้ผู้สอนไม่ต้องเสียเวลาตรวจข้อนี้ด้วยตนเองอีกครั้งหนึ่ง

นอกจากนี้ผลของการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม บอกให้ทราบถึงพฤติกรรมในการพัฒนาโปรแกรมของผู้ทดสอบ เช่น ผู้ทดสอบใช้แนวคิดหรือวิธีแก้ไขปัญหาในรูปแบบไหน มีการเปลี่ยนแปลงแนวทางการแก้ไขปัญหาหรือไม่ หรือผู้ทดสอบมีความเข้าใจมากน้อยเพียงใดในการทำโจทย์ ซึ่งแสดงจากค่าเปอร์เซ็นต์ความละม้ายที่เพิ่มขึ้นเรื่อย ๆ ตามลำดับการบันทึกการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่เพิ่มขึ้น แสดงว่าผู้ทดสอบมีความเข้าใจว่าจะเขียนอย่างไรในแต่ละขั้น การเพิ่มคำสั่งเข้าไปจึงเพิ่มค่าเปอร์เซ็นต์ความละม้ายในทุก ๆ คำสั่งที่เพิ่ม เป็นต้น ซึ่งผลของการเปรียบเทียบของลำดับของรหัสต้นฉบับโปรแกรม กับชุดเฉลย ทั้งหมดในแต่ละแบบทดสอบ แยกเป็นรายบุคคล รวม 70 ชุด ดูรายละเอียดเพิ่มเติมได้ในภาคผนวก ง

จากผลการทดลองเปรียบเทียบความละม้าย สามารถสรุปลักษณะของความละม้ายได้ 3 รูปแบบ ดังแสดงในภาพที่ 4.13



รูปแบบที่ 1 รูปแบบที่ 2 รูปแบบที่ 3

ภาพที่ 4.13 รูปแบบของความละม้ายที่สามารถเกิดขึ้น ในการเปรียบเทียบรหัสต้นฉบับของโปรแกรม กับชุดเฉลย

รูปแบบที่ 1 คือลักษณะซึ่งแสดงว่า ณ ลำดับที่เปรียบเทียบ รหัสต้นฉบับของโปรแกรมที่ผู้ทดสอบพัฒนา อยู่ภายใต้แนวทางของชุดเฉลยชุดนี้ทั้งหมด ไม่มีชุดคำสั่งชุดไหนที่ผู้ทดสอบพิมพ์ลงไปแล้วไม่ปรากฏในรหัสต้นฉบับของชุดเฉลย

รูปแบบที่ 2 คือลักษณะที่แสดงว่า ณ ลำดับที่เปรียบเทียบ รหัสต้นฉบับของโปรแกรมที่ผู้ทดสอบพัฒนา ชุดคำสั่งบางส่วนมีปรากฏในรหัสต้นฉบับของชุดเฉลย และมีชุดคำสั่งบางส่วนที่ผู้ทดสอบพัฒนาที่ไม่ปรากฏในรหัสต้นฉบับของชุดเฉลย

รูปแบบที่ 3 คือลักษณะที่แสดงว่า ณ ลำดับที่เปรียบเทียบ ชุดคำสั่งของรหัสต้นฉบับของชุดเฉลยทั้งหมด ปรากฏในรหัสต้นฉบับของโปรแกรมของผู้ทดสอบ และมีชุดคำสั่งเพิ่มเติมที่ผู้ทดสอบเพิ่มเข้าไปเกินกว่าชุดคำสั่งที่ปรากฏในชุดเฉลย ในกรณีนี้ เช่น หลังจากที่ผู้ทดสอบทำเสร็จแล้ว ผู้ทดสอบใส่ชุดคำสั่งเพื่อสั่งพิมพ์ค่าของตัวแปรบางตัวออกมาดูเพื่อตรวจสอบ หรือทำเกินกว่าที่โจทย์กำหนด การปฏิบัติลักษณะนี้ทำให้มีชุดคำสั่งในรหัสต้นฉบับของโปรแกรมของผู้ทดสอบเกินไปกว่าชุดคำสั่งที่มีในชุดเฉลย เป็นต้น

จากการวิเคราะห์ลักษณะของความละม้ายทั้ง 3 รูปแบบ มีเพียงรูปแบบที่ 1 เท่านั้นที่สามารถนำไปให้คะแนนอัตโนมัติในขั้นต่อไปได้ เนื่องจากในรูปแบบที่ 2 และ 3 นั้น เมื่อนำไปประเมินรหัสต้นฉบับของโปรแกรม และชุดเฉลย ด้วยมาตรวัดทางวิศวกรรมซอฟต์แวร์ และเทียบเป็นสัดส่วนเพื่อให้คะแนน จะมีชุดคำสั่งอื่น ๆ ที่ปรากฏในรหัสต้นฉบับของโปรแกมของผู้ทดสอบที่นอกเหนือจากชุดเฉลย ซึ่งเมื่อประเมินเป็นค่าความพยายามที่ใช้ในการพัฒนาจากมาตรวัดของฮอลสตีดแล้ว ชุดคำสั่งในส่วนที่ไม่ละม้ายนี้จะถูกประเมินค่าด้วย ทำให้ได้ค่าความพยายามของรหัสต้นฉบับของโปรแกรมที่เกินความจริง

4.1.3 **ผลการเปรียบเทียบคะแนน**

ในการเปรียบเทียบเพื่อให้คะแนน หลังจากที่เปรียบเทียบในขั้นตอนที่ 2 จะได้คู่ของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่คล้ายที่สุด และนำมาคำนวนคะแนน โดยมีขอบเขตในการพิจารณาให้คะแนนเฉพาะผู้ที่ทำแบบทดสอบไม่เสร็จในเวลา ซึ่งหมายถึงผู้ที่ทำไม่ทันเวลา และผู้ที่ทำทันเวลาแต่เมื่อตรวจสอบแล้วปรากฏว่าทำผิด โดยผลสรุปการเปรียบเทียบความละม้ายตามลำดับการเปลี่ยนแปลงของโปรแกรม แสดงในตารางที่ 4.1 – 4.5 ดังต่อไปนี้

ตารางที่ 4.1 ผลการทดสอบแบบทดสอบที่ 1 ของผู้ที่ทำไม่เสร็จภายในเวลา

ผู้ทดสอบ	จำนวนลำดับที่บันทึกได้	ตรวจตามลำดับการเปลี่ยนแปลง			ตรวจผลลัพธ์สุดท้าย	
		ลำดับที่ละม้ายมากสุด	ชุดเฉลยที่ละม้าย	% ความละม้าย	ชุดเฉลยที่ละม้าย	% ความละม้าย
คนที่ 2	23	23	1	54	1	54
คนที่ 5	6	6	6	50	6	50
คนที่ 6	11	11	4	40	4	40
คนที่ 8	16	6	6	93	6	93
คนที่ 9	5	5	6	71	6	71
คนที่ 10	21	21	13	95	13	95
คนที่ 14	12	12	13	84	13	84

ในแบบทดสอบที่ 1 มีผู้ทำเสร็จ 9 คน และทำไม่เสร็จจำนวน 7 คน ผลการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม และการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย ให้ผลเหมือนกันคือ พบลำดับที่มีค่าเปอร์เซ็นต์ความละม้ายสูงสุดที่ลำดับของรหัสต้นฉบับของโปรแกรมลำดับสุดท้าย

ตารางที่ 4.2 ผลการทดสอบแบบทดสอบที่ 2 ของผู้ที่ทำไม่เสร็จภายในเวลา

ผู้ทดสอบ	จำนวนลำดับที่บันทึกได้	ตรวจตามลำดับการเปลี่ยนแปลง			ตรวจผลลัพธ์สุดท้าย	
		ลำดับที่ละม้ายมากสุด	ชุดเฉลยที่ละม้าย	% ความละม้าย	ชุดเฉลยที่ละม้าย	% ความละม้าย
คนที่ 1	30	30	1	75	1	75
คนที่ 3	33	33	2	97	2	97
คนที่ 5	25	22	12	81	2	75
คนที่ 6	12	12	2	54	2	54
คนที่ 7	41	34	12	83	5	73
คนที่ 8	15	15	8	54	8	54
คนที่ 9	8	8	8	34	8	34
คนที่ 10	27	27	8	72	8	72
คนที่ 11	42	42	9	58	9	58
คนที่ 13	79	79	2	63	2	63

คนที่ 14	27	27	12	33	12	33
คนที่ 15	33	33	13	89	13	89
คนที่ 16	27	27	14	72	14	72

ในแบบทดสอบที่ 2 มีผู้ทำเสร็จ 1 คน ไม่ได้ทำ 2 คน และทำไม่เสร็จจำนวน 13 คน ผลการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม และการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย พบผู้ที่มีผลการตรวจเหมือนกัน คือลำดับของรหัสต้นฉบับของโปรแกรมที่ละม้ายที่สุดเป็นลำดับสุดท้ายจำนวน 11 คน และผู้ที่ได้ผลไม่เหมือนกัน 2 คน คือ ผู้ทดสอบคนที่ 5 และ 7 ซึ่งผลการตรวจตามลำดับการเปลี่ยนแปลงพบค่าเปอร์เซ็นต์ความละม้ายที่สูงกว่า ค่าเปอร์เซ็นต์ความละม้ายของลำดับสุดท้าย

ตารางที่ 4.3 ผลการทดสอบแบบทดสอบที่ 3 ของผู้ที่ทำไม่เสร็จภายในเวลา

ผู้ทดสอบ	จำนวนลำดับที่บันทึกได้	ตรวจตามลำดับการเปลี่ยนแปลง			ตรวจผลลัพธ์สุดท้าย	
		ลำดับที่ละม้ายมากสุด	ชุดเฉลยที่ละม้าย	% ความละม้าย	ชุดเฉลยที่ละม้าย	% ความละม้าย
คนที่ 8	14	14	1	86	1	86
คนที่ 9	9	9	2	85	2	85
คนที่ 10	12	12	5	96	5	96
คนที่ 11	16	16	7	88	7	88
คนที่ 13	19	12	6	94	7	89

ในแบบทดสอบที่ 3 มีผู้ทำเสร็จ 9 คน ไม่ได้ทำ 2 คน และทำไม่เสร็จจำนวน 5 คน ผลการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม และการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย พบผู้ที่มีผลการตรวจเหมือนกัน คือลำดับของรหัสต้นฉบับของโปรแกรมที่ละม้ายที่สุดเป็นลำดับสุดท้ายจำนวน 4 คน และผู้ที่ได้ผลไม่เหมือนกัน 1 คน คือ ผู้ทดสอบคนที่ 13 ซึ่งผลการตรวจตามลำดับการเปลี่ยนแปลงพบค่าเปอร์เซ็นต์ความละม้ายที่สูงกว่าค่าเปอร์เซ็นต์ความละม้ายของลำดับสุดท้าย

ตารางที่ 4.4 ผลการทดสอบแบบทดสอบที่ 4 ของผู้ที่ทำไม่เสร็จภายในเวลา

ผู้ทดสอบ	จำนวนลำดับที่บันทึกได้	ตรวจตามลำดับการเปลี่ยนแปลง			ตรวจผลลัพธ์สุดท้าย	
		ลำดับที่ละม้ายมากสุด	ชุดเฉลยที่ละม้าย	% ความละม้าย	ชุดเฉลยที่ละม้าย	% ความละม้าย
คนที่ 8	19	10	4	80	4	76
คนที่ 9	5	3	1	75	1	75
คนที่ 10	14	11	6	92	6	92
คนที่ 14	3	3	7	96	7	96
คนที่ 16	5	5	8	78	8	78

ในแบบทดสอบที่ 4 มีผู้ทำเสร็จ 8 คน ไม่ได้ทำ 3 คน และทำไม่เสร็จจำนวน 5 คน ผลการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม และการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย พบผู้ที่มีผลการตรวจเหมือนกัน คือลำดับของรหัสต้นฉบับของโปรแกรมที่ละม้ายที่สุดเป็นลำดับสุดท้ายจำนวน 3 คน และผู้ที่ได้ผลไม่เหมือนกัน 1 คน คือ ผู้ทดสอบคนที่ 8 ซึ่งผลการตรวจตามลำดับการเปลี่ยนแปลงพบค่าเปอร์เซ็นต์ความละม้ายที่สูงกว่า ค่าเปอร์เซ็นต์ความละม้ายของลำดับสุดท้าย

ตารางที่ 4.5 ผลการทดสอบแบบทดสอบที่ 5 ของผู้ที่ทำไม่เสร็จภายในเวลา

ผู้ทดสอบ	จำนวนลำดับที่บันทึกได้	ตรวจตามลำดับการเปลี่ยนแปลง			ตรวจผลลัพธ์สุดท้าย	
		ลำดับที่ละม้ายมากสุด	ชุดเฉลยที่ละม้าย	% ความละม้าย	ชุดเฉลยที่ละม้าย	% ความละม้าย
คนที่ 8	10	10	6	79	6	79
คนที่ 9	5	2	6	70	7	69
คนที่ 10	8	8	3	83	3	83

ในแบบทดสอบที่ 5 มีผู้ทำเสร็จ 10 คน ไม่ได้ทำ 3 คน และทำไม่เสร็จจำนวน 3 คน ผลการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม และการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย พบผู้ที่มีผลการตรวจเหมือนกัน คือลำดับของรหัสต้นฉบับของโปรแกรมที่ละม้ายที่สุดเป็นลำดับสุดท้ายจำนวน 2 คน และผู้ที่ได้ผลไม่เหมือนกัน 1 คน คือ ผู้ทดสอบคนที่ 9 ซึ่งผลการตรวจตามลำดับการเปลี่ยนแปลงพบค่าเปอร์เซ็นต์ความละม้ายที่สูงกว่า ค่าเปอร์เซ็นต์ความละม้ายของลำดับสุดท้าย

เมื่อนำคู่ของลำดับของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายที่สุดมาวิเคราะห์เนื้อหาของรหัสต้นฉบับของโปรแกรม ประเมินค่าความพยายามตามมาตรวัดของฮอลสตีด และคำนวนให้คะแนนตามสมการ ดังต่อไปนี้

$$P = \alpha \cdot S + (1 - \alpha) \cdot E$$

โดยที่ P เป็นเปอร์เซ็นต์ของคะแนนที่ได้

$\propto$ คือ ความน่าจะเป็น หรือโอกาสที่จะเกิด (probability)

S คือ ค่าเปอร์เซ็นต์ความละม้าย

E คือ ค่าเปอร์เซ็นต์ของความพยายามที่คำนวนได้

โดยที่ค่าเปอร์เซ็นต์ของความพยายาม คำนวนได้จาก

$$E = 1 - \frac{|E(x) - E(s)|}{Max(E(s), E(x))}$$

โดยที่ E เป็นเปอร์เซ็นต์ของความพยายามที่คำนวนได้

E(x) เป็นค่าความพยายามของฮอลสตีดที่คำนวนได้จากรหัสต้นฉบับของผู้เรียนที่ละม้ายที่สุด (snapshot)

E(s) เป็นค่าความพยายามของฮอลสตีดที่คำนวนได้จากชุดเฉลยที่ละม้ายที่สุด (solution)

จากสมการข้างต้น ผลการคำนวนเปอร์เซ็นต์ของคะแนนจากผู้ที่ทำไม่เสร็จ ได้ผลดังนี้

ตารางที่ 4.6 สรุปผลการคำนวนเปอร์เซ็นต์ของคะแนนของผู้ที่ทำแบบทดสอบไม่เสร็จ

ผู้ทดสอบ	**แบบทดสอบ 1**			**แบบทดสอบ 2**			**แบบทดสอบ 3**			**แบบทดสอบ 4**			**แบบทดสอบ 5**		
	S	E	P	S	E	P	S	E	P	S	E	P	S	E	P
คนที่ 1	[redacted]			73	52	61	[redacted]								
คนที่ 2	33	15	22	[redacted]											

คนที่ 3				96	72	82									
คนที่ 4															
คนที่ 5	30	30	30	79	56	65									
คนที่ 6	31	22	26	56	21	35									
คนที่ 7				81	73	76									
คนที่ 8	90	37	58	50	17	30	54	78	68	44	42	43	33	21	26
คนที่ 9	60	19	35	29	6	15	50	32	39	25	17	20	56	66	62
คนที่ 10	93	96	95	69	45	55	88	95	92	75	60	66	40	6	21
คนที่ 11				56	34	43	71	27	45						
คนที่ 12															
คนที่ 13				60	36	46	84	80	82						
คนที่ 14	80	40	56	27	4	13				61	75	69			
คนที่ 15				89	83	85									
คนที่ 16				70	55	61				29	42	37			

จากผลของคะแนนที่แสดงในตารางที่ 4.6 เช่น ค่าเปอร์เซ็นต์คะแนนของผู้ทดสอบคนที่ 6 ของแบบทดสอบที่ 1 ได้ 26 % มีรายละเอียดของลำดับของรหัสต้นฉบับของโปรแกรม และชุดเฉลยที่ละม้ายที่สุด เป็นดังนี้

```
public class Lab1BMI {
  public static float getBMI(float height, float weight) {
    return (weight/(height*height));
  }
  public static void main(String[] args) {
    //Please fill in commands here
    double height = 1.6;
    float weight = 71;
    System.out.println(String.valueOf(getBMI(height,weight)));
  }
}
```

ภาพที่ 4.14 รหัสต้นฉบับของผู้ทดสอบคนที่ 6 ของแบบทดสอบที่ 1 ณ ลำดับที่ 11

```
import java.util.Scanner;
public class Lab1BMI {
  public static double getBMI(double height, double weight) {
    if ((height > 0) && (weight > 0))
      return (weight/(height*height));
    return 0;
  }
  public static double scanKeyboardInput(String InputDescString) {
    Scanner sc = new Scanner(System.in);
    if (sc.hasNext())
      return sc.nextDouble();
    return 0;
  }
  public static void main(String[] args) {
    //Please fill in commands here
    double height = 1.6;
    double weight = 71;

    height = scanKeyboardInput("Height = ");
    weight = scanKeyboardInput("Weight = ");

    System.out.println(String.valueOf(getBMI(height,weight)));
  }
}
```

ภาพที่ 4.15 รหัสต้นฉบับของชุดเฉลยชุดที่ 4 ของแบบทดสอบที่ 1

จากภาพที่ 4.14 และ 4.15 รหัสต้นฉบับของโปรแกรม ณ ลำดับที่ 11 แสดงให้เห็นว่า ผู้ทดสอบคนที่ 6 เขียนโครงสร้างของโปรแกรม โดยประกาศตัวแปร height และ weight และสร้างเมทอด getBMI และคำสั่งพิมพ์ผลลัพธ์ของการคำนวนค่า BMI โดยเรียกเมทอด getBMI ที่สร้างขึ้น ซึ่งเมื่อเทียบกับผลเฉลยจะมีรายละเอียดคำสั่งที่มากกว่า คือ มีการเรียกรับค่าตัวแปร height และ weight จากเมทอด scanKeyboardInput ที่สร้างขึ้นมาเพื่อใช้รับข้อมูลเข้า และมีการปรับการตรวจสอบค่า height และ weight ที่ส่งเข้าไปยัง getBMI เพื่อป้องกันข้อผิดพลาดที่เกิดจากการหารด้วยค่า 0 จากความแตกต่างของรหัสต้นฉบับของโปรแกรมทั้งสอง ได้ค่าเปอร์เซ็นต์

คะแนนที่ได้เป็น 26 % ซึ่งยอมรับได้เมื่อดูจากผลของการทำงานที่ต้องทำเพิ่มจนกระทั่งสิ้นสุดโปรแกรมตามชุดเฉลยที่ 4

กรณีที่ค่าเปอร์เซ็นต์คะแนนอยู่ในช่วงกลาง ๆ เช่น ค่าคะแนนของผู้ทดสอบที่ 10 ของแบบทดสอบที่ 4 ได้ 66 % มีรายละเอียดของลำดับของรหัสต้นฉบับของโปรแกรม และชุดเฉลยที่ละม้ายที่สุด เป็นดังนี้

```
import java.util.Scanner;
public class Lab4CheckPrecision {
  public static void main(String[] args) {
    System.out.println("Please select k value :");
    Scanner sc = new Scanner(System.in);
    int k = sc.nextInt();
    System.out.println("Nearest positive integer which give wrong
precision value is " + foo(k));
  }
  public static int foo(int k){
    int result = 0;
    //Please fill in commands here
    int x=1;
    if(k/x*x!=k){
      result++;
    }
    return result;
  }
}
```

ภาพที่ 4.16 รหัสต้นฉบับของผู้ทดสอบคนที่ 10 ของแบบทดสอบที่ 4 ณ ลำดับที่ 11

```
import java.util.Scanner;
public class Lab4CheckPrecision {
  public static void main(String[] args) {
    System.out.println("Please select k value :");
    Scanner sc = new Scanner(System.in);
    int k = sc.nextInt();
    System.out.println("Nearest positive integer which give wrong
precision value is " + foo(k));
  }
  public static int foo(int k){
    int result = 0;
    //Please fill in commands here
    for(int i = 1; true; i++){
      if(((double)k/i)*i != k)
        return i;
    }
    return result;
  }
}
```

ภาพที่ 4.17 รหัสต้นฉบับของชุดเฉลยชุดที่ 6 ของแบบทดสอบที่ 4

จากภาพที่ 4.16 และ 4.17 รหัสต้นฉบับของโปรแกรมเริ่มต้นสำหรับผู้ทดสอบจะมีคำสั่งในส่วนของเมทอด main มาให้ ผู้ทดสอบเพียงแต่พัฒนาในส่วนของเมทอด foo ซึ่งมีวัตถุประสงค์เพื่อรับค่าของตัวเลขจำนวนเต็มเข้ามา และหาว่าจำนวนเต็มที่น้อยที่สุดที่นำไปหารและคูณกลับแล้วได้ค่าไม่เท่าเดิม ซึ่งรหัสต้นฉบับของโปรแกรม ณ ลำดับที่ 11 ดังภาพที่ 4.16 แสดงให้เห็นว่า ผู้ทดสอบคนที่ 10 เขียนคำสั่งในการคำนวนโดยการนำตัวเลขจำนวนเต็มไปหารและคูณกลับ และนำมาเปรียบเทียบกับค่าตัวเลขจำนวนเต็มที่รับเข้ามาก่อน ซึ่งเมื่อเทียบกับผลเฉลยจะมีรายละเอียดคำสั่งที่มากกว่า คือ มีการเขียนวนซ้ำของอาเรย์โดยเริ่มต้นจาก 1 เพื่อทดสอบหาค่าที่น้อยที่สุดที่ทำให้ผลการคำนวนเป็นไปตามที่โจทย์กำหนด จากความแตกต่างของรหัสต้นฉบับของโปรแกรมทั้งสอง ได้ค่าเปอร์เซ็นต์คะแนนเป็น 66 % ซึ่งยอมรับได้เมื่อดูจากผลของการทำงานที่ต้องทำเพิ่มจนกระทั่งสิ้นสุดโปรแกรมตามชุดเฉลยที่ 6

จากการวิเคราะห์ผลของคะแนนที่ได้ทั้งหมดเมื่อเทียบกับรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายแสดงให้เห็นว่า วิธีที่ใช้ในการเปรียบเทียบความละม้ายเพื่อหาคู่ของลำดับของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายที่สุด เพื่อนำมาคำนวนค่าความพยายามจากมาตรวัดของฮอลสตีด และเทียบเป็นสัดส่วนให้คะแนนได้ผลตามแนวคิดที่ออกแบบไว้ ไม่มีค่าผลคะแนนที่ได้เกินร้อยเปอร์เซ็นต์ ซึ่งแสดงว่ามีคำสั่งที่เป็นส่วนเกินเข้ามาปนในลำดับรหัสต้นฉบับของโปรแกรม ทำให้ในการประเมินค่าความพยายามจะได้ค่าที่ไม่ถูกต้องเมื่อนำไปเทียบกับค่าความพยายามของชุดเฉลย จากผลการคำนวน ค่าเปอร์เซ็นต์ของคะแนนที่ได้มีความเหมาะสม เมื่อนำคู่ของรหัสต้นฉบับของโปรแกรมมาเปรียบเทียบเพื่อยืนยันผลของคะแนน ดังตัวอย่างที่ได้แสดงในภาพที่ 4.14 – 4.17

4.2 **สรุปผลการวิเคราะห์**

ผลที่ได้จากการวิเคราะห์ข้อมูลสรุปได้ว่า วิธีที่เหมาะสมในการเก็บบันทึกการเปลี่ยนแปลงของโปรแกรมควรใช้วิธีเก็บบันทึกรายคำสั่ง ซึ่งจะได้ผลการบันทึกที่สื่อความหมายถึงสิ่งที่ผู้ทดสอบตั้งใจจะเขียนลงไปได้ครบถ้วน จำนวนลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม มีจำนวนไม่มากเมื่อเทียบกับวิธีอื่น และเหมาะสมที่จะนำไปใช้เพื่อเปรียบเทียบความละม้ายในขั้นต่อไป

ในการวิเคราะห์เปรียบเทียบความละม้าย ผลที่ได้แสดงให้เห็นว่าการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรมให้ผลที่ดีกว่าการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย ซึ่งจากข้อมูลแสดงให้เห็นว่ามีลำดับที่อยู่ก่อนลำดับสุดท้ายของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบที่แสดงว่าผู้ทดสอบมีความเข้าใจมากกว่ารหัสต้นฉบับของโปรแกรมที่ได้ ณ ลำดับสุดท้าย หรือถูกต้อง

มากกว่า และการเปรียบเทียบความละม้ายโดยนำโปรแกรมการตรวจการลอกกัน (Plagiarism detection) เช่น Plaggie มาใช้ สามารถเปรียบเทียบเพื่อหาคู่ของรหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่ดีที่สุดได้ผลดี ผลลัพธ์จากการจับคู่ได้ผลลัพธ์ที่ถูกต้อง สามารถใช้วิเคราะห์จับคู่หาแนวทางการพัฒนาของผู้ทดสอบได้ว่าผู้ทดสอบดำเนินการพัฒนาโปรแกรมในแนวทางของเฉลยชุดไหน ซึ่งผลของคู่ของรหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่ละม้ายที่สุดที่ได้จะมีสำคัญต่อความถูกต้องในการคำนวนให้คะแนนตามค่าความพยายามจากมาตรวัดของฮอลสตีดในขั้นสุดท้าย

ในการคำนวนให้คะแนนตามค่าความพยายามจากมาตรวัดของฮอลสตีด ผลที่ได้แสดงให้เห็นว่า วิธีในการนับจำนวนตัวดำเนินการและจำนวนตัวถูกดำเนินการ ซึ่งเป็นพารามิเตอร์พื้นฐานของสูตรการคำนวนความซับซ้อน และค่าความพยายามตามมาตรวัดของฮอลสตีด ให้ผลคะแนนที่สอดคล้องกับเนื้อหาของรหัสต้นฉบับของโปรแกรมกับเนื้อหาของชุดเฉลยที่ละม้ายที่สุด แบบทดสอบที่มีในวิชาการทำโปรแกรมคอมพิวเตอร์ มีจำนวนบรรทัดในการพัฒนาโปรแกรมไม่มาก เช่น จากการทดสอบ จำนวนบรรทัดของรหัสต้นฉบับของโปรแกรมที่ได้จากการพัฒนาอยู่ระหว่าง 10 ถึง 30 บรรทัด การทำงานของโปรแกรมส่วนใหญ่เป็นการคำนวน การใช้คำสั่งควบคุมต่าง ๆ ทำให้การนำตัวดำเนินการและตัวถูกดำเนินการมาใช้ในการคำนวนค่าความพยายามเพื่อให้คะแนน ได้ผลที่ดี

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

การวิจัยระบบการตรวจให้คะแนนอัตโนมัติ โดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรม มีวัตถุประสงค์เพื่อศึกษาการให้คะแนนโดยดูตามลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมที่ทำให้ได้คุณภาพในการตรวจและประเมินผลงานของผู้เรียน ได้เทียบเท่าหรือดีกว่าการตรวจโดยดูผลลัพธ์สุดท้าย และนำมาพัฒนาเป็นระบบการให้คะแนนอัตโนมัติ เพื่อลดภาระงานของผู้สอนและผู้ช่วยสอน

การวิจัยอาศัยหลักทฤษฎีของตัววัดซอฟต์แวร์ (Software metric) เช่น ค่าความพยายามของเฮลสเตด (Halstead) มาเปรียบเทียบให้คะแนน ผลที่ได้ปรากฏว่าในการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรมให้ผลลัพธ์ที่ดีกว่า หรือเทียบเท่าการตรวจเฉพาะผลลัพธ์สุดท้าย เมื่อวิเคราะห์ลำดับการเปลี่ยนแปลงในบางลำดับ ผู้เรียนมีความรู้ความเข้าใจในการทำแบบทดสอบที่มากกว่าเมื่อดูเฉพาะผลลัพธ์สุดท้าย ผู้สอนสามารถประเมินความรู้ความเข้าใจของผู้เรียนได้ละเอียดมากขึ้น นอกจากนี้ลำดับการเปลี่ยนแปลงของโปรแกรมยังบอกถึงพฤติกรรมในการพัฒนางานของผู้เรียน ทำให้ผู้สอนมีข้อมูลจากการประเมินผู้เรียนมากกว่าการตรวจเฉพาะผลลัพธ์สุดท้าย สามารถย้อนกลับไปดูเพื่อหาสาเหตุที่ผู้เรียนไม่เข้าใจ ทำให้ได้ข้อมูลการประเมินผู้เรียนได้ถูกต้องครบถ้วนมากขึ้น ผู้สอนสามารถให้คำแนะนำในการแก้ปัญหาของผู้เรียนได้ตรงประเด็นและมีประสิทธิภาพมากขึ้น

การเก็บข้อมูลตามลำดับการเปลี่ยนแปลงของโปรแกรมมาตรวจสอบ จำเป็นต้องมีระบบอัตโนมัติเข้ามาช่วยในการวิเคราะห์ข้อมูลของผู้เรียนที่บันทึกไว้ เพราะว่าข้อมูลที่จัดเก็บจะเพิ่มขึ้นจากการเก็บเฉพาะผลลัพธ์สุดท้ายมาก ซึ่งระบบวิเคราะห์อัตโนมัตินี้จะช่วยให้วิธีการตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรมนี้ไม่กลายเป็นภาระเพิ่มแก่ผู้สอน การใช้ระบบการตรวจการลอกกัน (Plagiarism protection) มาช่วยในการวิเคราะห์ลำดับของรหัสต้นฉบับของผู้เรียนเพื่อหาแนวทางการแก้ไขปัญหาของโจทย์ที่ผู้เรียนคิด เป็นวิธีหนึ่งที่ได้ผลดี สามารถค้นพบได้ว่าผู้เรียนใช้แนวทางไหนในการแก้ไขปัญหา ซึ่งทำให้รู้ปลายทางของงานที่ผู้เรียนทำ เมื่อสามารถรู้ปลายทางที่ผู้เรียนจะพัฒนาไปถึงได้ ทำให้สามารถใช้การประเมินรหัสต้นฉบับของโปรแกรมด้วยมาตรวัดทางวิศวกรรมซอฟต์แวร์มาคำนวนเพื่อเทียบเป็นสัดส่วนที่บอกว่าผู้เรียนพัฒนางานมาไกลแค่ไหนแล้ว ทำให้สามารถพัฒนาเป็นระบบให้คะแนนอัตโนมัติเพื่อช่วยผู้สอนให้คะแนนแก่ผู้เรียนได้ ซึ่งระบบที่พัฒนาขึ้นทั้งหมดนี้ทำให้ผู้สอนมีข้อมูลจากการตรวจผลงานของผู้เรียนที่ละเอียดขึ้น

สามารถประเมินการเรียนรู้ของผู้เรียนได้ถูกต้องแม่นยำขึ้น และไม่เพิ่มภาระงานแก่ผู้สอนมากขึ้นกว่าการตรวจโดยดูเฉพาะผลลัพธ์สุดท้าย ผู้สอนจะสามารถมุ่งเน้นไปยังการเรียนการสอน และการให้แบบฝึกหัดแก่นักเรียนได้สะดวกขึ้น โดยไม่กลับมาเป็นภาระในการตรวจงานแก่ผู้สอน ส่งผลให้การเรียนการสอนด้านการทำโปรแกรมคอมพิวเตอร์มีประสิทธิผลดีขึ้น

5.2 ข้อเสนอแนะ

1) การปรับปรุงการจัดเก็บข้อมูล

ในการจัดเก็บข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม อาจนำระบบ revision control system เช่น cvs หรือ svn เข้ามาช่วยในการจัดเก็บการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมได้ เพื่ออำนวยความสะดวกในการรวมศูนย์การจัดเก็บและเรียกใช้ข้อมูล และลดขนาดของพื้นที่ในการจัดเก็บข้อมูล กรณีที่กลุ่มผู้เรียนมีจำนวนมาก

2) การทดสอบโปรแกรมในการเปรียบเทียบความละม้ายอื่น ๆ เพิ่มเติม

ในส่วนของการเปรียบเทียบความละม้าย งานวิจัยนี้ได้ใช้โปรแกรม Plaggie มาช่วยในการเปรียบเทียบเพื่อหาคู่ของรหัสต้นฉบับของโปรแกรมและชุดเฉลยที่ละม้ายที่สุดเพื่อนำไปใช้ประเมินให้คะแนน จากแนวคิดในการออกแบบระบบซึ่งออกแบบไว้ให้สามารถปรับเปลี่ยนโปรแกรมที่ใช้ในการเปรียบเทียบความละม้ายได้ ผู้ที่สนใจสามารถที่จะทดลองปรับเปลี่ยนโปรแกรมที่ช่วยในการเปรียบเทียบความละม้ายได้ เพื่อให้ได้ประสิทธิภาพทางด้านความเร็วในการเปรียบเทียบที่ดีขึ้น หรือได้ผลของการเปรียบเทียบที่ถูกต้องแม่นยำ หรือครอบคลุม หรือมีคุณสมบัติอื่น ๆ ที่มากกว่าโปรแกรม Plaggie

3) การพัฒนาปรับปรุงอัลกอริทึมในการเปรียบเทียบความละม้าย

ในการเปรียบเทียบความละม้าย งานวิจัยนี้ได้นำวิธีในการตรวจสอบการลอกกัน (Plagiarism detection) มาใช้ในการเปรียบเทียบ ซึ่งมีข้อดีคือ งานทางด้านการตรวจสอบการลอกกัน มีการพัฒนาและมีงานวิจัยออกมาเป็นจำนวนมาก อย่างไรก็ตามจากการวิจัยพบว่า การเปรียบเทียบลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย จะมีข้อมูลที่ต้องเปรียบเทียบจำนวนมาก การปรับปรุงอัลกอริทึมในการเปรียบเทียบความละม้ายให้สามารถวิเคราะห์รหัสต้นฉบับของโปรแกรมในการเปรียบเทียบแบบเพิ่ม (incremental) ได้ เป็นอีกวิธีที่น่าสนใจที่จะทำให้การเปรียบเทียบความละม้ายมีประสิทธิภาพในการทำงานที่สูงขึ้น ทำให้ไม่ต้อง

วิเคราะห์รหัสต้นฉบับของโปรแกรมซ้ำ ๆ ทั้งๆ ที่ลำดับของโปรแกรมแต่ละลำดับมีความแตกต่างกันไม่มาก ซึ่งเป็นลักษณะเฉพาะของข้อมูลลำดับการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม

4) การทดสอบมาตรวัดทางวิศวกรรมซอฟต์แวร์อื่น ๆ เพื่อใช้คำนวนให้คะแนน

ในการนำมาตรวัดทางวิศวกรรมซอฟต์แวร์มาใช้ในการประเมินรหัสต้นฉบับของโปรแกรม เพื่อคำนวนให้คะแนน สามารถที่จะลองนำมาตรวัดตัวอื่น ๆ ที่มีการพัฒนาเพิ่มเติม หรือมีการนำเสนอในงานวิจัยใหม่ ๆ มาทดสอบ เพื่อปรับปรุงสูตรในการคำนวนให้คะแนน เพื่อให้รองรับการให้คะแนนหลายรูปแบบมากขึ้น หรือได้คะแนนที่เหมาะสมมากขึ้น

5) การทดสอบกับกลุ่มทดสอบขนาดใหญ่ หรือมีความสามารถในการเรียนรู้ที่แตกต่างกัน

ในการทดสอบระบบกับกลุ่มทดสอบขนาดใหญ่ หรือผู้เรียนที่ต่างสถาบันกัน จะทำให้ได้กลุ่มทดสอบที่มีความสามารถในการเรียนรู้ที่แตกต่างกันมาก มีพื้นฐานความรู้ความเข้าใจและความสามารถที่หลากหลาย เพื่อนำผลการทดสอบที่ได้มาปรับปรุงระบบให้สมบูรณ์ขึ้น

6) การนำไปประยุกต์ใช้กับการเรียนการสอนการทำโปรแกรมภาษาอื่น ๆ

แนวคิดในงานวิจัยนี้สามารถนำไปประยุกต์ใช้ในการพัฒนาระบบตรวจข้อสอบอัตโนมัติ โดยตรวจตามลำดับการเปลี่ยนแปลงของโปรแกรมในภาษาอื่น ๆ เช่น C C++ C# เป็นต้น

7) การนำไปประยุกต์ใช้กับการเรียนการสอนในสาขาวิชาอื่น

แนวคิดในงานวิจัยนี้สามารถนำไปประยุกต์ใช้ในการพัฒนาระบบตรวจข้อสอบอัตโนมัติสำหรับใช้ในสาขาวิชาอื่น เช่น ในการเรียนการสอนการเขียนเรียงความภาษาอังกฤษ (Essay) ซึ่งระบบการเขียนภาษาอังกฤษมีการหยุดประโยคที่แน่นอนโดยใช้ “.” (Full stop) ทำให้สามารถนับจำนวนประโยคที่เพิ่มขึ้น ลดลง หรือมีการแก้ไขประโยคได้ ทำให้สามารถประเมินความรู้ความเข้าใจของผู้เรียนได้ว่ามีความเข้าใจและสามารถเขียนเรียงความภาษาอังกฤษได้ดีหรือไม่ ถ้ามีความเข้าใจดีการเพิ่มขึ้นของจำนวนประโยคน่าจะเพิ่มขึ้นต่อเนื่อง มีการแก้ไขน้อย หรือมีการลดลงของประโยคไม่มาก เป็นต้น

8) การนำไปพัฒนาเป็นระบบในการวิเคราะห์พฤติกรรมของผู้เรียน

ในการพัฒนาระบบการวิเคราะห์พฤติกรรมของผู้เรียน จากลักษณะของค่าความละม้ายที่ได้จากการวิเคราะห์ลำดับของรหัสต้นฉบับของโปรแกรม สามารถบอกได้ว่าผู้เรียนมีความเข้าใจมากน้อยเพียงไร และระยะเวลาที่ใช้ในการเขียนชุดคำสั่งแต่ละคำสั่งใช้เวลาเท่าไร มีการหยุดเว้นช่วงสั้นหรือยาว ช่วงที่หยุดเว้นยาวติดอยู่ที่คำสั่งไหน เป็นต้น ข้อมูลพื้นฐานเหล่านี้สามารถนำมาวิเคราะห์พฤติกรรมและความรู้ความเข้าใจของผู้เรียนได้

9) การนำไปใช้เป็นระบบช่วยเหลืออัตโนมัติแก่ผู้เรียน

แนวคิดในการเปรียบเทียบหาคู่ของรหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่ละม้าย สามารถนำไปใช้เป็นระบบช่วยเหลืออัตโนมัติแก่ผู้เรียนได้ โดยวิเคราะห์ว่าผู้เรียนกำลังพัฒนาโปรแกรมไปในแนวทางไหน ถ้าผู้เรียนติดหรือทำต่อไม่ได้กลางทาง ระบบสามารถวิเคราะห์ได้ว่าคำสั่งต่อไปควรจะใช้คำสั่งอะไร เมื่อผู้เรียนกดขอความช่วยเหลือจากระบบ ระบบสามารถแนะนำผู้เรียนได้อัตโนมัติ

10) การนำไปใช้ในระบบ E-Learning

ในการพัฒนาระบบ E-Learning ทางด้านการทำโปรแกรมคอมพิวเตอร์ กราฟที่ได้จากการเปรียบเทียบความละม้ายของลำดับของรหัสต้นฉบับของโปรแกรมกับชุดเฉลย สามารถวิเคราะห์พฤติกรรมของผู้เรียนได้ว่าแบบทดสอบที่ผู้เรียนทำเสร็จไป ผู้เรียนมีความเข้าใจดีหรือยัง ถ้ากราฟแสดงถึงการเพิ่มขึ้นของค่าความละม้ายอย่างต่อเนื่องจนจบถึงร้อยเปอร์เซ็นต์ แสดงว่าผู้เรียนมีความเข้าใจดีในการทำโจทย์ข้อนี้ ผู้เรียนสามารถข้ามไปทำข้อต่อไปได้ ในทางกลับกันถ้าแสดงว่าผู้เรียนยังไม่ค่อยเข้าใจดี ระบบอาจให้ผู้เรียนทำจนกว่าจะมีความเข้าใจถึงจะสามารถทำข้อต่อไปได้ เป็นต้น

รายการอ้างอิง

[1] สมชาย ประสิทธิ์จูตระกูล. การทำโปรแกรมคอมพิวเตอร์ (2110101). [ออนไลน์]. 2551. แหล่งที่มา: http://www.cp.eng.chula.ac.th/~somchai/ 2110101 [2553,พฤษภาคม 1]

[2] Michael J. Wise. String similarity via greedy string tiling and running Karp-Rabin matching [online]. 1993. Available from : http://www.pam1.bcs.uwa.edu.au/~michaelw/ftp/doc/RKR_GST.ps [2010, May 1].

[3] Prechelt, L., Malpohl G., and Philippsen, M. Finding plagiarisms among a set of programs with JPlag. Journal of Universal Computer Science. 8(11) : 1016-1038. 2002.

[4] บุญธรรม กิจปรีดาบริสุทธิ์. การวัดและประเมินผลการสอน. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : บี แอนด์ บี พับลิชชิ่ง, 2535.

[5] ปราโมทย์ ลือนาม. การพัฒนาโปรแกรมเพื่อใช้วัดความซับซ้อนของซอฟต์แวร์. วิทยาศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2536.

[6] David Binkley. Source Code Analysis: A Road Map. Future of Software Engineering (FOSE'07). pp. 104-119. IEEE, 2007.

[7] James W. Howatt. On criteria for grading student programs. SIGCSE, ACM. 1994.

[8] Andrew Violette. Generating syntactical deltas from java source code [online]. 1999. Available from : http://facweb.cs.depaul.edu/ctiphd/ctirs99/online/violette.html [2010,May 1]

[9] Jose Paulo Leal,and Nelma Moriera. Automatic grading of programming exercises, University of Porta. 1998.

[10] Rachel Edita Roxas, Nathalie Rose Lim,and Natasja Bautista. Automatic Generation of Plagiarism Detection Among Student Programs. Information Techonology Based Higher Education and Training, 2006. pp. 226-235. IEEE. 2006.

[11] Tobias Sager, Abraham Bernstein, Martin Pinzger,and Christoph Keifer. Detecting similar java classes using tree algorithms. Proceedings of the 2006 international workshop on Mining software repositories. pp. 65-71. ACM. 2006.

[12] Chao Liu, Chen, Jiawei Han,and Philip S. Yu. GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis. Proceedings of the 12th

ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 872-881. ACM. 2006.

[13] J.Ferrante, K.J. Ottenstein,and J.D. Warren. The program dependence graph and its use in optimization. ACM Trans. Program. Lang Syst. 9(3) (1987) : 319-349.

[14] Ahtiainen, A., Surakka, S.,and Rahikainen, M. Plaggie: GNU-licensed Source Code Plagiarism Detection Engine for Java Exercises. Proceedings, Koli Calling. 2006.

[15] Schleimer, S., Wilkerson, D.,and Aiken, A. Winnowing: Local Algorithms for Document Fingerprinting. SIGMOD. San diego, CA, 9 – 12 June 2003.

[16] Thompson, E., Luxton-Reilly, A., Whalley, J., Hu, M.,and Robbins, P. Bloom's Taxonomy for CS Assessment. Tenth Australasian Computing Education Conference (ACE2008). Wollongong, Australia. 2008.

ภาคผนวก

ภาคผนวก ก

แบบทดสอบและชุดเฉลยที่ใช้ในการทดสอบ

<u>แบบทดสอบ</u>

1. จงเขียนชุดคำสั่งสำหรับตรวจสอบ ความอ้วนหรือผอม โดยใช้สูตรคำนวนของ BMI

   ระยะเวลา/คะแนน : 5 นาที / 10คะแนน

   คำอธิบาย : ถ้าเราต้องการอยากรู้ว่าอ้วนหรือผอม สามารถตรวจสอบได้จากค่า BMI (body mass index) จากสูตร

$$BMI = \frac{w}{h^2}$$

   โดย w คือน้ำหนักมีหน่วยเป็น กิโลกรัม และ h คือ ความสูงมีหน่วยเป็นเมตร

   วิธีการตรวจสอบ : คำนวนค่า BMI ตาม น้ำหนัก และส่วนสูงที่รับค่าเข้ามา

   - ค่า BMI น้อยกว่า 18.5 แสดงว่าผอมไป
   - ค่า BMI มากกว่าหรือเท่ากับ 25 แสดงว่าอ้วนไป
   - ค่า BMI อยู่ในช่วง 18.5 ถึง 25 แสดงว่าปรกติ

   ผลลัพธ์ที่ต้องการ : ให้เขียนการทำงานของคลาส Lab1BMI โดยมีการทำงานภายใน method main ดังนี้

   - แสดงคำว่า weight (kg.) : แล้วรอรับให้ผู้ใช้ใส่ค่าน้ำหนักเป็นกิโลกรัม
   - แสดงคำว่า height (m) : แล้วรอรับให้ผู้ใช้ใส่ส่วนสูงเป็นเมตร
   - คำนวนค่าตามสูตร BMI
   - แสดงค่า BMI ของทางหน้าจอ (ไม่ต้องแสดงข้อความใด ๆ กำกับ)

   ไฟล์งาน : Lab1BMI.java

   ตัวอย่าง :

```
JLab>java BMI
weight (kg.) : 65
height (cm.) : 173
21.71806608974573
JLab>
```

2. จงเขียนชุดคำสั่งสำหรับตรวจสอบหมายเลข ISBN-10 ว่ามีค่าถูกต้องตามข้อกำหนดหรือไม่

   ระยะเวลา/คะแนน : 10 นาที / 10 คะแนน

   คำอธิบาย : หนังสือที่พิมพ์กันทั้งหลายในโลกนี้จะมีหมายเลข ISBN ขนาด 10 หลัก กำกับ เช่น “Compilers: Principle, Techniques, and Tools” มีหมายเลข ISBN คือ 0201100886

วิธีการตรวจสอบ : หมายเลข 10 หลัก จะใช้มีเลขหลักที่ 10 เป็นตัวตรวจสอบความถูกต้องของข้อมูลด้วยเลขอีก 9 ตัวทางด้านซาย โดยให้ d_k แทนตัวเลขหลักที่ k จะได้ว่า check digit ต้องมีค่าเท่ากับ

$$\left(\sum_{k=1}^{9}(k \times d_k)\right) \bmod 11$$

ผลลัพธ์ที่ต้องการ : เขียนคลาส Lab2ISBN โดยมีการทำงานใน method main ให้รอรับหมายเลข ISBN จากผู้ใช้ทางแป้นพิมพ์เพื่อตรวจสอบว่า ISBN ที่ได้รับถูกต้องหรือไม่ โดยมีการตรวจสอบความถูกต้อง 4 แบบ คือ

- แบบที่ 1 หมายเลข ISBN ไม่ครบหรือเกิน 10 หลัก ให้แสดงคำว่า INCORRECT 1
- แบบที่ 2 หมายเลข ISBN 9 หลักทางซ้ายบางตัวไม่ใช่ตัวเลข หรือตัวที่ 10 ไม่ใช่ตัวเลขหรือตัวอักษร x ให้แสดงคำว่า INCORRECT 2
- แบบที่ 3 ค่า check digit มีค่าไม่ถูกต้องตามกฎ ให้แสดงคำว่า INCORRECT 3
- แบบที่ 4 ค่า ISBN ที่รับเข้ามาถูกต้องตามกฎเกณฑ์ ให้แสดงคำว่า CORRECT

ไฟล์งาน : Lab2ISBN.java

ตัวอย่าง : ถ้า 9 หลักซ้ายของ ISBN คือ 020110088 จะคำนวณ check digit ได้

1x0+ 2x2 + 3x0 + 4x1 + 5x1 + 6x0 + 7x0 + 8x8 + 9x8 = 177

check digit คือ 149 mod 11 = 6

(ในกรณีที่ mod 11 แล้วได้ค่าเป็น 10 จะใช้ตัวอักษร x แทนเลข 10)

```
JLab>java ISBN
ISBN : 9742290261
CORRECT
JLab>
```

```
JLab>java ISBN
ISBN : 12345
INCORRECT 1
JLab>
```

```
JLab>java ISBN
ISBN : 12345X7890
INCORRECT 2
JLab>
```

```
JLab>java ISBN
ISBN : 987654321W
INCORRECT 2
JLab>
```

```
JLab>java ISBN
ISBN : 987654321X
INCORRECT 3
JLab>
```

3. จงเขียนโปรแกรมเพื่อหาว่ามีข้อมูลใน data array ที่เมื่อนำ ข้อมูลใน data ใน array ไปเรียงลำดับจากน้อยไปมากแล้วไม่มีการเปลี่ยนตำแหน่ง

ระยะเวลา/คะแนน : 5 นาที / 10 คะแนน

คำอธิบาย : การนำข้อมูล array ของ data ไปเรียงลำดับจากน้อยไปมาก เมื่อเรียงเสร็จเรียบร้อยแล้วต้องการทราบว่ามีข้อมูลที่ตำแหน่งใน array ที่ไม่มีการเปลี่ยนตำแหน่งหลังจากเรียงลำดับใหม่

สิ่งที่ต้องการ : ให้เขียนชุดคำสั่งการทำงานใน method getNumNoMoveData เพื่อคืนค่าตัวเลขจำนวนข้อมูลที่ไม่มีการเปลี่ยนตำแหน่ง ให้สมบูรณ์ตามโจทย์ที่กำหนด

ไฟล์งาน : Lab3CheckNoMoveData.java

ตัวอย่าง : กำหนดให้ data = (9,3,6,1,7) ถ้านำไปเรียงลำดับจะได้ (1,3,6,7,9) ซึ่งจะได้ว่าเลข 3 และ เลข 6 อยู่ในตำแหน่งเดิมไม่เปลี่ยนแปลง ดังนั้น method getNumNoMoveData จะคืนค่า 2 ออกไป

4. จงเขียนโปรแกรมเพื่อตรวจสอบความเที่ยงตรงของการคำนวน precision

ระยะเวลา/คะแนน : 5 นาที/10 คะแนน

คำอธิบาย : ในการคำนวนโดยปรกติแล้ว $\left(\frac{k}{x}\right)\cdot x = k$ แต่คอมพิวเตอร์มีข้อจำกัดในเรื่องความละเอียดในการคำนวนเลขทศนิยมซึ่งอาจจะทำให้ (k/x)*x != k ได้

สิ่งที่ต้องการ : ให้เขียนชุดคำสั่งใน public static int foo(int k) เพื่อนำค่า k ที่รับเข้ามาคำนวนหาและคืนค่าจำนวนเต็มบวก x ที่มีค่าน้อยที่สุดที่ทำให้ผลคำนวนของ ((double) k/x)*x != k

ไฟล์งาน : Lab4CheckPrecision.java

5. จงเขียนโปรแกรมเพื่อนับจำนวนคู่ใน array ที่มีค่าแตกต่างกันอยู่ k

ระยะเวลา/คะแนน : 10 นาที / 10 คะแนน

สิ่งที่ต้องการ : ให้เขียนชุดคำสั่งใน public static int diffByK(int[] array, int k) เพื่อคำนวนหาว่าข้อมูลใน array มีอยู่กี่คู่ที่มีค่าต่างกัน k

ไฟล์งาน : Lab5DiffByK.java

ตัวอย่าง : ถ้า array มีข้อมูลเป็น (10,9,12,1,8,2) เมื่อ k = 1 จะมีคู่ข้อมูลที่มีค่าแตกต่างกันอยู่ 1 คือ (10,9) , (9,8) , (1,2) ซึ่งนับรวมได้ 3 คู่

เฉลยแบบทดสอบ

- ชุดเฉลยสำหรับแบบทดสอบที่ 1

ชุดที่	เนื้อหาของรหัสต้นฉบับของเฉลย
1	`public class Lab1BMI {` `  public static void main(String[] args){` `  // Please fill in commands here` `  Double w = Double.valueOf(System.console().readLine("weight (kg):"));` `  Double h = Double.valueOf(System.console().readLine("height (m):"));` `  System.out.println(w / (h * h));` `  }` `}`
2	`import java.io.*;` `public class Lab1BMI {` `  public static void main(String[] args) {` `  //Please fill in commands here` `  double BMI =0;` `  System.out.print("weight (kg.) : ");` `  String line = null;` `  int weight = 0;` `  double height = 0;` `  try{` `  BufferedReader is = new BufferedReader(new InputStreamReader(System.in));` `  line = is.readLine();` `  weight = Integer.parseInt(line);` `  }catch(NumberFormatException ex) {` `  System.err.println("Not a valid number: " + line);` `  }catch(IOException e) {` `  System.err.println("Unexpected IO ERROR: " + e);` `  }` `  System.out.print("height (m) :");` `  line = null;` `  try{` `  BufferedReader is = new BufferedReader(` `  new InputStreamReader(System.in));` `  line = is.readLine();` `  height = Double.parseDouble(line);` `  }catch(NumberFormatException ex) {` `  System.err.println("Not a valid number: " + line);` `  }catch(IOException e) {` `  System.err.println("Unexpected IO ERROR: " + e);` `  }` `  BMI = weight /(height*height);` `  System.out.println(BMI);` `  }` `}`
3	`import java.util.*;` `public class Lab1BMI {` `  public static void main(String[] args) {` `  //Please fill in commands here` `  Scanner sc = new Scanner(System.in);` `  System.out.print("Weight : ");` `  String weight = sc.next();` `  System.out.print("Height : ");` `  String height = sc.next();` `  float w = Float.parseFloat(weight);` `  float h = Float.parseFloat(height);` `  h = h/100;` `  float r = w/(h*h);` `  System.out.print("Result : ");` `  System.out.print(r);` `  }` `}`
4	`import java.util.Scanner;` `public class Lab1BMI {` `  public static double getBMI(double height, double weight) {` `  if ((height > 0) && (weight > 0))`

	`` return (weight/(height*height)); return 0; } public static double scanKeyboardInput(String InputDescString) { Scanner sc = new Scanner(System.in); if (sc.hasNext()) return sc.nextDouble(); return 0; } public static void main(String[] args) { //Please fill in commands here double height = 1.6; double weight = 71; height = scanKeyboardInput("Height = "); weight = scanKeyboardInput("Weight = "); System.out.println(String.valueOf(getBMI(height,weight))); } } ```
5	``` import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here Scanner sc = new Scanner(System.in); System.out.print("weight (kg.) :"); double weight = sc.nextDouble(); System.out.print("height (m) :"); double height = sc.nextDouble(); System.out.println(weight/(height*height)); } } ```
6	``` import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here Scanner sc = new Scanner(System.in); System.out.println("Weight (kg.) :"); double weight = sc.nextDouble(); System.out.println("Height (m.) :"); double height = sc.nextDouble(); System.out.println(weight/(height*height)); } } ```
7	``` import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here int w, h; Scanner snData = new Scanner(System.in); System.out.print("weight (kg.) : "); w = snData.nextInt(); System.out.print("height (cm.) : "); h = snData.nextInt(); double EMI = ((double)w) / ((h / 100) ^ 2); System.out.println(EMI); } } ```
8	``` import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { int weight,height; float fheight; //Please fill in commands here Scanner sd = new Scanner(System.in); System.out.print("weightb(kg.) : "); weight = sd.nextInt(); System.out.print("height (cm.) :"); height = sd.nextInt(); fheight = (float)height / 100; System.out.println((float)weight/(fheight*fheight)); } } ```

9	import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here double w,h; System.out.println("weight (kg.) :"); Scanner snData = new Scanner(System.in); w = snData.nextDouble(); System.out.println("height (cm.) :"); h = snData.nextDouble(); double emi = (w/(h*h/10000)); System.out.println(emi); } }
10	import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here Scanner snData = new Scanner(System.in); System.out.println("weight (kg.) : "); double weight = snData.nextDouble(); System.out.println("height (cm.) : "); double height = snData.nextDouble(); double bmi = (weight / height/ height); System.out.println(bmi); } }
11	import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here Scanner scn = new Scanner(System.in); float weight,height; System.out.print("weight (kg.) : "); weight = scn.nextFloat(); System.out.print("height (m) : "); height = scn.nextFloat(); System.out.println(weight / height / height * 10000); } }
12	import java.util.Scanner; public class Lab1BMI { public static void main(String[] args) { //Please fill in commands here double bmi = 0.0; Scanner sc = new Scanner(System.in); System.out.println("weight(kg.) :"); float w = sc.nextFloat(); System.out.println("height(m) :"); float h =sc.nextFloat(); bmi = w/(h*h); System.out.println(bmi); } }
13	import java.io.*; public class Lab1BMI { public static void main(String[] args)throws IOException { BufferedReader in = new BufferedReader(new InputStreamReader (System.in)); System.out.println("input weight = :"); int h = Integer.parseInt(in.readLine()); System.out.println("input height = :"); int w = Integer.parseInt(in.readLine()); double total = w/Math.pow(h, 2); System.out.println(total); } }

- ชุดเฉลยสำหรับแบบทดสอบที่ 2

ชุดที่	เนื้อหาของรหัสต้นฉบับของเฉลย
1	`public class Lab2ISBN {` `  public static void main(String[] args) {` `  //Please fill in commands here` `  String isbn = System.console().readLine("ISBN:");` `  try {` `  int checksum=0;` `  String lastc="";` `  if (isbn.length() != 10)` `  throw new Exception("1");` `  for (int i=0;i<isbn.length();i++) {` `  String c = isbn.substring(i,i+1);` `  int di=0;` `  if (!(i==9 && "X".equals(c.toUpperCase()))) {` `  di = Integer.valueOf(c);` `  }` `  if (i<9)` `  checksum = checksum + (i+1)*di;` `  else {` `  lastc = c.toUpperCase();` `  }` `  }` `  checksum = checksum % 11;` `  if (checksum==10 && lastc.equals("X"))` `  System.out.println("CORRECT");` `  else if (lastc.equals(String.valueOf(checksum).trim()))` `  System.out.println("CORRECT");` `  else` `  System.out.println("INCORRECT 3");` `  } catch (Exception e) {` `  if ("1".equals(e.getMessage()))` `  System.out.println("INCORRECT 1");` `  else` `  System.out.println("INCORRECT 2");` `  }` `  }` `}`
2	`import java.util.Scanner;` `public class Lab2ISBN {` `  public static void main(String[] args) {` `  //Please fill in commands here` `  Scanner scn = new Scanner(System.in);` `  System.out.print("ISBN : "); String st = scn.next();` `  if (st.length() < 10 || st.length() > 10 ) {` `  System.out.println("INCORRECT 1");` `  return ;` `  }` `  int sum = 0;` `  int digit = 0;` `  for (int i  = 0;i < st.length();i++) {` `  try {` `  digit = Integer.parseInt(st.charAt(i)+"");` `  } catch (NumberFormatException e) {` `  if (i < st.length() - 1 || st.charAt(9) != 'X') {` `  System.out.println("INCORRECT 2");` `  return;` `  }` `  }` `  if (i < st.length() -1) {` `  sum = sum + (i+1) * digit;` `  }` `  }` `  int checkDigit =  (st.charAt(9) == 'X') ? 10 :` `  Integer.parseInt(st.charAt(9)+"");` `  if (sum % 11 != checkDigit) {` `  System.out.println("INCORRECT 3");` `  return ;` `  }`

```
    System.out.println("CORRECT");
  }
}
```

3

```
import java.util.Scanner;
public class Lab2ISBN {
  public static void main(String[] args) {
    // Please fill in commands here
    Scanner scanner = new Scanner(System.in);
    System.out.print("ISBN : ");
    String ISBN = scanner.nextLine();
    if (ISBN.length() != 10) {
      System.out.println("INCORRECT 1");
    }else{
      int i;
      char elem;
      for (i = 0; i < 9; i++) {
        elem = ISBN.charAt(i);
        if (!Character.isDigit(elem))
          break;
      }
      elem = ISBN.charAt(9);
      if (i == 9) {
        if (Character.isDigit(elem) || elem == 'X') {
          int sum = 0;
          for (i = 0; i < 9; i++) {
            int elem_int = Integer.parseInt(ISBN.charAt(i) + "");
            System.out.println("elem_digit = " + elem_int);
            sum += (i + 1) * elem_int;
          }
          sum = sum % 11;
          System.out.println("sum = " + sum);
          if ((sum == 10 && ISBN.charAt(9) == 'X')
              || sum == Integer.parseInt(ISBN.charAt(9) + "")) {
            System.out.println("CORRECT");
          }else{
            System.out.println("INCORRECT 3");
          }
        }else
          System.out.println("INCORRECT 2");
      }else{
        System.out.println("INCORRECT 2");
      }
    }
  }
}
```

4

```
import java.util.Scanner;
public class Lab2ISBN {
  public static int parityCheckISBN(int[] first9digits) {
    int checksum = 0;
    for (int i=1; i<10; i++) {
      checksum += i*first9digits[i-1];
    }
    return (checksum % 11);
  }
  public static void main(String[] args) {
    //Please fill in commands here
    Scanner sc = new Scanner(System.in);
    System.out.println("Fill in ISBN (without dash)");
    String datastr = sc.next();
    if (datastr.length() != 10) {
      System.out.println("INCORRECT1");
      return;
    }
    char[] char_array = datastr.toCharArray();
    int[] int_array = new int[10];
    for(int i=0; i<9; i++) {
      try {
        int_array[i] = Integer.parseInt(String.valueOf(char_array[i]));
      }catch(Exception e){
        System.out.println("INCORRECT2");
        return;
      }
    }
```

```
    int x;
    try {
     x = Integer.parseInt(String.valueOf(char_array[9]));
    }catch(Exception e) {
     if (char_array[9] != 'x') {
      System.out.println("INCORRECT2");
      return;
     }else{
      x = 10;
     }
    }
    if (parityCheckISBN(int_array) == x) {
     System.out.println("CORRECT");
    }else{
     System.out.println("INCORRECT3");
    }
    return;
   }
}
```

5

```
import java.util.Scanner;
public class Lab2ISBN {
 public static void main(String[] args) {
  //Please fill in commands here
  Scanner s = new Scanner(System.in);
  System.out.print("ISBN : " );
  String isbn = s.nextLine();
  int length = isbn.length();
  if(length!=10)
   System.out.println("INCORRECT 1");
  else {
   int sum=0;
   for(int i=0;i<length-1;i++){
    if(isbn.charAt(i)-'0'>9||isbn.charAt(i)-'0'<0){
     System.out.println("INCORRECT 2");
     return;
    }
    sum += (isbn.charAt(i)-'0') * (i+1);
   }
   if(isbn.charAt(9)!='x'){
    if(isbn.charAt(9)-'0'>9||isbn.charAt(9)-'0'<0){
     System.out.println("INCORRECT 2");
     return;
    }
   }
   int chk = sum % 11;
   //System.out.println(chk);
   if(chk==10&&isbn.charAt(9)=='x'){
    System.out.println("CORRECT");
    return;
   }
   if(chk!=isbn.charAt(9)-'0'){
    System.out.println("INCORRECT 3");
   }else
    System.out.println("CORRECT");
  }
 }
}
```

6

```
import java.util.*;
public class Lab2ISBN {
 public static void main(String[] args) {
  //Please fill in commands here
  Scanner sc = new Scanner(System.in);
  System.out.print("ISBN : ");
  String isbn = sc.next();
  if(isbn.length()!=10 ){
   System.out.print("INCORRECT 1");
  }else if(isbn.length()==10 && ((!isbn.substring(9,10).equals("X") ||
            isbn.substring(9,10).matches("/^\\d{13}$/") ) ||
            isbn.substring(0,9).matches("/^\\d{13}$/") )){
   double isbnx1 = ((1*Double.valueOf(isbn.substring(0,1)).doubleValue())
                   +(2*Double.valueOf(isbn.substring(1,2)).doubleValue())
                   +(3*Double.valueOf(isbn.substring(2,3)).doubleValue())
                   +(4*Double.valueOf(isbn.substring(3,4)).doubleValue())
```

```
                      +(5*Double.valueOf(isbn.substring(4,5)).doubleValue())
                      +(6*Double.valueOf(isbn.substring(5,6)).doubleValue())
                      +(7*Double.valueOf(isbn.substring(6,7)).doubleValue())
                      +(8*Double.valueOf(isbn.substring(7,8)).doubleValue())
                      +(9*Double.valueOf(isbn.substring(8,9)).doubleValue()))%11;
    if(isbnx1==10.0 && isbn.substring(9,10).equals("X")){
    }
    System.out.print("INCORRECT 2");
   }else if(isbn.length()==10 && (isbn.substring(9,10).equals("X")
            || isbn.substring(9,10).matches("/^\\d{13}$/")) ){
    double isbnx2 = ((1*Double.valueOf(isbn.substring(0,1)).doubleValue())
                      +(2*Double.valueOf(isbn.substring(1,2)).doubleValue())
                      +(3*Double.valueOf(isbn.substring(2,3)).doubleValue())
                      +(4*Double.valueOf(isbn.substring(3,4)).doubleValue())
                      +(5*Double.valueOf(isbn.substring(4,5)).doubleValue())
                      +(6*Double.valueOf(isbn.substring(5,6)).doubleValue())
                      +(7*Double.valueOf(isbn.substring(6,7)).doubleValue())
                      +(8*Double.valueOf(isbn.substring(7,8)).doubleValue())
                      +(9*Double.valueOf(isbn.substring(8,9)).doubleValue()))%11;
    if(isbnx2==10.0 && isbn.substring(9,10).equals("X")){
     System.out.print("CORRECT");
    }else{
     System.out.print("INCORRECT 3");
    }
   }else{
    System.out.print("CORRECT");
   }
  }
}
```

7

```
import java.util.*;
public class Lab2ISBN {
 public static void main(String[] args) {
  Scanner s = new Scanner(System.in);
  System.out.print("ISBN : ");
  String isbn = s.next();
  if(isbn.length()!=10){
   System.out.print("INCORRECT 1");
  }else if(isbn.length()==10 && ((!isbn.substring(9,10).equals("X")
            || isbn.substring(9,10).matches("/^\\d{13}$/") )
            || isbn.substring(0,9).matches("/^\\d{13}$/") )){
   System.out.print("INCORRECT 2");
  }else if(isbn.length()==10 && (isbn.substring(9,10).equals("X")
          || isbn.substring(9,10).matches("/^\\d{13}$/")) ){
   double isbnx2 = ((1 * Double.valueOf(isbn.substring(0,1)).doubleValue())
                      +(2*Double.valueOf(isbn.substring(1,2)).doubleValue())
                      +(3*Double.valueOf(isbn.substring(2,3)).doubleValue())
                      +(4*Double.valueOf(isbn.substring(3,4)).doubleValue())
                      +(5*Double.valueOf(isbn.substring(4,5)).doubleValue())
                      +(6*Double.valueOf(isbn.substring(5,6)).doubleValue())
                      +(7*Double.valueOf(isbn.substring(6,7)).doubleValue())
                      +(8*Double.valueOf(isbn.substring(7,8)).doubleValue())
                      +(9*Double.valueOf(isbn.substring(8,9)).doubleValue()))%11;
   if(isbnx2==10.0 && isbn.substring(9,10).equals("X")){
    System.out.print("CORRECT");
   }else{
    System.out.print("INCORRECT 3");
   }
  }else{
   System.out.print("CORRECT");
  }
 }
}
```

8

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Lab2ISBN {
 public static void main(String[] args) throws Exception{
  //Please fill in commands here
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  String isbn=br.readLine();
  if(isbn.length()!=10)System.out.println("incorrect 1");
  try{
   int tmp=Integer.parseInt(isbn.substring(0,8));
  }catch(Exception e){
```

```
     System.out.println("incorrect 2");
    }
    int tmp2=0;
    for(int i=0;i<isbn.length();i++){
     tmp2=tmp2+i*Integer.parseInt(isbn.substring(i, i+1));
    }
    if(tmp2 % 11==Integer.parseInt(isbn.substring(8, 9))){
     System.out.println("correct");
    }else{
     System.out.println("incorrect");
    }
   }
}
```

9

```
import java.util.Scanner;
public class Lab2ISBN {
 public static void main(String[] args) {
  //Please fill in commands here
  Scanner sc = new Scanner(System.in);
  System.out.println("Please input ISBN : ");
  String isbn = sc.nextLine();
  int chk1 = 0;
  int chk = 0;
  int j=9;
  int sum=0;
  if(isbn.length()<10){
   System.out.println("INCORRECT CODE 1");
  }else{
   for (int i = 0; i < 10; i++){
    try{
     int inp2 =0;
     if(i==9&&(isbn.substring(i, i + 1).equals("x")
        ||isbn.substring(i, i + 1).equals("X"))){
      inp2=10;
     }else{
      inp2 = Integer.parseInt(isbn.substring(i, i + 1));
      if (i == 10)
       chk1 = inp2;
      else{
       System.out.println(inp2+"*"+(i+1));
       chk = inp2 * (i+1);
       sum = sum + chk;
      }
     }
    }catch(NumberFormatException ex){
     System.out.println("INCORRECTCODE 2 ");
     System.exit(0);
    }
   }
   int MD1 = sum % 11;
   if(MD1==chk1){
    System.out.println("CORRECT");
   }else{
    System.out.println("INCORRECTCODE 3");
   }
  }
 }
}
```

10

```
import java.util.Scanner;
public class Lab2ISBN {
 public static void main(String[] args) {
  //Please fill in commands here
  Scanner snData = new Scanner(System.in);
  System.out.println("ISBN : ");
  String isbn = snData.next();
  if( isbn.length() != 10){
   System.out.println("INCORRECT 1");
   return;
  }
  for(int i = 0; i < 10; i++){
   if(!(('0' <= isbn.charAt(i) && isbn.charAt(i) <= '9')
       || (i == 9 && isbn.charAt(i)=='X'))){
    System.out.println("INCORRECT 2");
    return;
```

```
    }
   }
   int sum = 0;
   for(int i = 0; i < 9; i++)
    sum += (i+1 ) * (isbn.charAt(i)-'0');

   sum %= 11;
   if((sum == 10 && isbn.charAt(9) != 'X')
       ||(sum != 10 && sum != isbn.charAt(9)-'0')){
    System.out.println("INCORRECT 3");
    return;
   }
   System.out.println("CORRECT");
  }
}
```

11

```
import java.util.Scanner;
public class Lab2ISBN {
 public static void main(String[] args) {
  //Please fill in commands here
  System.out.println("ISBN : ");
  Scanner snData = new Scanner(System.in);
  String allDigit = snData.nextLine();
  int sum = 0;
  if(allDigit.length()!=10){
   System.out.println("Incorrect 1");return;
  }
  for(int i=0;i>10;i++){
   try{
    int fag = Integer.parseInt(allDigit.substring(i, i+1));
    sum+=fag;
   }catch(Exception e){
    System.out.println("Incorrect 2");
    return;
   }
  }
  sum = sum%11;
  try{
   if(sum == Integer.parseInt(allDigit.substring(9))){
    System.out.println("Correct");return;
   }
  }catch(Exception e){}
  if(sum == 10 && allDigit.substring(9).equals("x")){
   System.out.println("Correct");}
  else{
   System.out.println("Incorrect 3");
   return;
  }
 }
}
```

12

```
import java.io.*;
public class Lab2ISBN {
 public static void main(String[] args) throws IOException{
  BufferedReader in = new BufferedReader(new InputStreamReader (System.in));
  System.out.println("input ISBN  = :");
  String L = in.readLine();
  if(L.length()!= 10) {
   System.out.println("INCORRECT1");
  }
  int N;
  try{
   N = Integer.parseInt(L);
  }catch(NumberFormatException e){
   System.out.println("INCORRECT2");
  }
  int x = 0;
  int a = 0;
  for (int i = 1 ; i<11 ;i++){
   a = Integer.parseInt(L.substring(i-1, i));
   if(i ==10 ){
    break;
   }
   x = x + (a*i);
  }
```

	`` if((x%11) == a){ System.out.println("Correct"); }else{ System.out.println("INCORRECT3"); } } } ```
13	``` import java.util.Scanner; public class Lab2ISBN { public static void main(String[] args) { //Please fill in commands here Scanner sc = new Scanner(System.in); System.out.print("ISBN : "); String isbn = sc.next(); isbn = isbn.trim(); if(isbn.length() != 10) System.out.println("INCORRECT 1"); else{ boolean chkInput = true; for(int i=0;i<9;i++){ try{ Integer.parseInt(String.valueOf(isbn.charAt(i))); }catch(Exception e){ chkInput = false; } } String chkDigit = String.valueOf(isbn.charAt(9)); if(chkDigit.toLowerCase().equals("x")) chkDigit = "10"; try{ Integer.parseInt(chkDigit); }catch(Exception e){ chkInput = false; } if(chkInput == false) System.out.println("INCORRECT 2"); else{ int sum = 0; for(int i=0;i<9;i++){ sum += (i+1)*Integer.parseInt(String.valueOf(isbn.charAt(i))); } if(Integer.parseInt(chkDigit).equals(sum%11)) System.out.println("CORRECT"); else System.out.println("INCORRECT 3"); } } } } ```
14	``` import java.util.Scanner; import java.util.regex.Matcher; import java.util.regex.Pattern; public class Lab2ISBN { public static void main(String[] args) { //Please fill in commands here Scanner sc = new Scanner(System.in); String isbn = ""; boolean checkNumber = false; System.out.print("isbn : "); isbn = sc.next(); Pattern p = Pattern.compile("[a-zA-Z]"); Matcher m = p.matcher(isbn); checkNumber = m.find(); if (isbn.length() > 10 || isbn.length() < 10){ System.out.println("INCORRECT 1 "); }else if (checkNumber){ System.out.println("INCORRECT 2 "); }else if (checkISBN(isbn)){ System.out.println("INCORRECT 3 "); }else{ System.out.println("CORRECT"); } ```

```
  }
  public static boolean checkISBN(String isbn){
   int isbnArray [] = new int[10];
   int isbnSum = 0;
   int j = 10;
   for (int i = 0 ; i<isbnArray.length; i++){
    isbnArray[i]= Integer.parseInt(isbn.substring(i,i+1));
    isbnSum += isbnArray[i]*j;
    j--;
   }
   System.out.println("sum : "+isbnSum);
   if (isbnSum % 11 != 0 ){
    return true;
   }
   return false;
  }
}
```

- ชุดเฉลยสำหรับแบบทดสอบที่ 3

ชุดที่	เนื้อหาของรหัสต้นฉบับของเฉลย
1	`import java.util.Arrays;` `import java.util.Scanner;` `public class Lab3CheckNoMoveData {` `  public static void main(String[] args) {` `   Scanner sc = new Scanner(System.in);` `   System.out.println("Please fill in data array(seperate value by ,)");` `   System.out.println("Example : 10,3,9,7,4,0");` `   System.out.println("Fill in data :");` `   String datastr = sc.next();` `   if(datastr.trim().length()>0){` `    String[] data = datastr.split(",");` `    int[] array = new int[data.length];` `    for(int i=0;i<data.length;i++)` `     array[i] = Integer.parseInt(data[i]);` `    System.out.println("No move count = " + getNumNoMoveData(array));` `   }else` `    System.out.println("Wrong fill in data");` `  }` `  public static int getNumNoMoveData(int[] data){` `   int count = 0;` `   //Please fill in commands here` `   int[] data2 = data.clone();` `   Arrays.sort(data2);` `   for (int i=0;i< data2.length;i++){` `    if (data[i]==data2[i])` `    count++;` `   }` `   return count;` `  }` `}`
2	`import java.lang.reflect.Array;` `import java.util.Arrays;` `import java.util.Scanner;` `public class Lab3CheckNoMoveData {` `  public static void main(String[] args) {` `   Scanner sc = new Scanner(System.in);` `   System.out.println("Please fill in data array(seperate value by ,)");` `   System.out.println("Example : 10,3,9,7,4,0");` `   System.out.println("Fill in data :");` `   String datastr = sc.next();` `   if(datastr.trim().length()>0){` `    String[] data = datastr.split(",");` `    int[] array = new int[data.length];` `    for(int i=0;i<data.length;i++)` `     array[i] = Integer.parseInt(data[i]);` `    System.out.println("No move count = " + getNumNoMoveData(array));` `  }else` `   System.out.println("Wrong fill in data");`

	} public static int getNumNoMoveData(int[] data){ int count = 0; //Please fill in commands here int[] destArr = new int[data.length]; for(int i=0;i<data.length;i++) destArr[i] = data[i]; Arrays.sort(destArr); for(int i=0;i<data.length;i++) if(destArr[i] != data[i]) count++; return data.length - count; } }
3	import java.util.Scanner; public class Lab3CheckNoMoveData { public static void main(String[] args) { Scanner sc = new Scanner(System.in); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("No move count = " + getNumNoMoveData(array)); }else System.out.println("Wrong fill in data"); } public static int getNumNoMoveData(int[] data){ int count = 0; //Please fill in commands here int sort[] = new int[data.length]; for(int i=0;i<data.length;i++){ sort[i]=data[i]; } for(int i=0;i<data.length;i++) for(int j=i+1;j<data.length;j++) if(sort[i]>sort[j]){ int t = sort[i]; sort[i]=sort[j]; sort[j]=t; for(int i=0;i<data.length;i++) if(sort[i]==data[i]) count++; return count; } }
4	import java.util.Scanner; import java.util.Arrays; public class Lab3CheckNoMoveData { public static void main(String[] args) { Scanner sc = new Scanner(System.in); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("No move count = " + getNumNoMoveData(array)); }else System.out.println("Wrong fill in data"); } public static int getNumNoMoveData(int[] data){ int count = 0; int[] data2 = data; int[] data3 = data;

```
  int minvalue=0;
  int position=0;
  String outputdata2="";
  Arrays.sort(data2);
  for(int j=0;j<data2.length;j++){
   outputdata2 = outputdata2+data2[j]+",";
  }
  System.out.println("DATA2 : "+outputdata2);
  outputdata2="";
  for(int j=0;j<data3.length;j++){
   outputdata2 = outputdata2+data3[j]+",";
  }
  System.out.println("DATA : "+outputdata2);
  for(int i=0;i<data2.length;i++){
   if(data2[i]==data[i]){
    count++;
   }
  }
  return count;
 }
}
```

5

```
import java.util.Arrays;
import java.util.Scanner;
public class Lab3CheckNoMoveData {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Please fill in data array(seperate value by ,)");
  System.out.println("Example : 10,3,9,7,4,0");
  System.out.println("Fill in data :");
  String datastr = sc.next();
  if(datastr.trim().length()>0){
   String[] data = datastr.split(",");
   int[] array = new int[data.length];
   for(int i=0;i<data.length;i++)
    array[i] = Integer.parseInt(data[i]);

   System.out.println("No move count = " + getNumNoMoveData(array));
  }else
   System.out.println("Wrong fill in data");
 }
 public static int getNumNoMoveData(int[] data){
  int count = 0;
  int a[] = data.clone();
  int b[]= data.clone();

  Arrays.sort(b);
  for(int i = 0 ; i<a.length ; i++){
   if (a[i]==b[i]){
    count++;
   }
  }
  return count;
 }
}
```

6

```
import java.util.Arrays;
import java.util.Scanner;
public class Lab3CheckNoMoveData {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Please fill in data array(seperate value by ,)");
  System.out.println("Example : 10,3,9,7,4,0");
  System.out.println("Fill in data :");
  String datastr = sc.next();
  if(datastr.trim().length()>0){
   String[] data = datastr.split(",");
   int[] array = new int[data.length];
   for(int i=0;i<data.length;i++)
    array[i] = Integer.parseInt(data[i]);

   System.out.println("No move count = " + getNumNoMoveData(array));
  }else
   System.out.println("Wrong fill in data");
 }
```

	public static int getNumNoMoveData(int[] data){ int count = 0; //Please fill in commands here int[] dataOrg = new int[data.length]; for (int i=0; i<data.length; i++) { dataOrg[i] = data[i]; } // Sort the data array Arrays.sort(data); for (int i=0; i<data.length; i++) { if (data[i] == dataOrg[i]) count++; } return count; } }
7	import java.util.Scanner; public class Lab3CheckNoMoveData { public static void main(String[] args) { Scanner sc = new Scanner(System.in); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("No move count = " + getNumNoMoveData(array)); }else System.out.println("Wrong fill in data"); } public static int getNumNoMoveData(int[] data){ int count = 0; int[] data2 = data.clone(); for(int i = 0; i < data.length; i++){ for(int j = 1; j < data.length; j++){ if(data[j-1] > data[j]){ int tmp = data[j-1]; data[j-1] = data[j]; data[j] = tmp; } } } for(int i = 0; i < data.length; i++) if(data[i] == data2[i]) count++; return count; } }

- ชุดเฉลยสำหรับแบบทดสอบที่ 4

ชุดที่	เนื้อหาของรหัสต้นฉบับของเฉลย
1	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 0; //Please fill in commands here boolean found=false; while (!found) { result++; double k2 = (k/result)*result;

	found = !((k2-k)==0); } return result; } }
2	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 0; //Please fill in commands here int x = 1; while (true) { if (((double) k/x)*x != k) break; x++; } result = x; return result; } }
3	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 1; //Please fill in commands here\ while(((double)k/result)*result == k){ result++; } return result; } }
4	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 0; double x = 1; //Please fill in commands here while(Double.compare((((double)k/x)*x),Double.valueOf(k)) == 0){ x+=1; } result = (int)x; return result; } }
5	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k));

	} public static int foo(int k){ int result = 0; //Please fill in commands here for(int i=1;i<=k;i++){ double newk = ((double)k/i)*i; //System.out.println(i + ":" + newk + " "); if(newk-k!=0.0) return i; } return k; } }
6	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 0; //Please fill in commands here for(int i = 1; true; i++){ if(((double)k/i)*i != k) return i; } //return result; } }
7	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 0; //Please fill in commands here for(int i = 1 ;i<=k;i++){ if(((double)k/i)*i!=k){ result = i; break; } } return result; } }
8	import java.util.Scanner; public class Lab4CheckPrecision { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Nearest positive integer which give wrong precision value is " + foo(k)); } public static int foo(int k){ int result = 0; //Please fill in commands here for(int i=1;i<=k;i++) if(((double)k/i)*i != k) return i; return result; } }

- ชุดเฉลยสำหรับแบบทดสอบที่ 5

ชุดที่	เนื้อหาของรหัสต้นฉบับของเฉลย
1	import java.util.Scanner; public class Lab5DiffByK { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("No move count = "+diffByK(array,k)); }else System.out.println("Wrong fill in data"); } public static int diffByK(int[] array,int k){ int result = 0; //Please fill in commands here for (int i=0;i<array.length-1;i++) { for (int j=i+1;j<array.length;j++) { int dif = array[i]-array[j]; if (dif==k || dif == (k*-1)) result++; } } return result; } }
2	import java.util.Scanner; public class Lab5DiffByK { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("DiffByK Count = "+diffByK(array,k)); }else System.out.println("Wrong fill in data"); } public static int diffByK(int[] array,int k){ int result = 0; //Please fill in commands here for(int i = 0;i < array.length;i++){ for(int j = i + 1;j < array.length;j++){ if(Math.abs(array[i] - array[j]) == k){ result++; } } } return result; } }
3	import java.util.Scanner; public class Lab5DiffByK {

	public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length() > 0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for (int i = 0; i < data.length; i++) array[i] = Integer.parseInt(data[i]); System.out.println("DiffByK Count = " + diffByK(array, k)); }else System.out.println("Wrong fill in data"); } public static int diffByK(int[] array, int k) { int result = 0; // Please fill in commands here for(int i = 0; i < array.length - 1; i++){ for(int j = i + 1; j < array.length; j++) { if (array[i] - array[j] == k || array[i] - array[j] == -k) result++; } } return result; } }
4	import java.util.Scanner; import java.lang.Math; import java.lang.Math; public class Lab5DiffByK { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("No move count = "+diffByK(array,k)); }else System.out.println("Wrong fill in data"); } public static int diffByK(int[] array,int k){ int result = 0; //Please fill in commands here for (int i=0; i<(array.length-1); i++){ for (int j=i+1; j<array.length; j++){ if (Math.abs(array[i]-array[j])==k) { result++; } } } return result; } }
5	import java.lang.reflect.Array; import java.util.Arrays; public class Lab5DiffByK { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :");

	String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("DiffByK Count = "+diffByK(array,k)); }else System.out.println("Wrong fill in data"); } public static int diffByK(int[] array,int k){ int result = 0; //Please fill in commands here Arrays.sort(array); for(int i=0;i<array.length;i++){ for(int j=i;j<array.length;j++){ if(array[j]+k==array[j]){result++;} } } return result; } }
6	import java.util.Scanner; public class Lab5DiffByK { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("DiffByK Count = "+diffByK(array,k)); }else System.out.println("Wrong fill in data"); } public static int diffByK(int[] array,int k){ int result = 0; //Please fill in commands here for(int i = 0;i<array.length;i++){ for(int j = i;j<array.length;j++){ if(Math.abs(array[i]-array[j]) == k){ result++; } } } return result; } }
7	import java.util.Scanner; public class Lab5DiffByK { public static void main(String[] args) { System.out.println("Please select k value :"); Scanner sc = new Scanner(System.in); int k = sc.nextInt(); System.out.println("Please fill in data array(seperate value by ,)"); System.out.println("Example : 10,3,9,7,4,0"); System.out.println("Fill in data :"); String datastr = sc.next(); if(datastr.trim().length()>0){ String[] data = datastr.split(","); int[] array = new int[data.length]; for(int i=0;i<data.length;i++) array[i] = Integer.parseInt(data[i]); System.out.println("DiffByK Count = "+diffByK(array,k)); }else System.out.println("Wrong fill in data");

```
 }
 public static int diffByK(int[] array,int k){
  int result = 0;
  int a[] = array.clone();
  for (int i = 0 ; i<a.length ; i++){
   for (int x = i+1 ; x<a.length ; x++){
    if (Math.abs(a[i]-a[x]) == k ){
     result++;
    }
   }
  }
  return result;
 }
}
```

ภาคผนวก ข

รายละเอียดของโปรแกรมเสริมที่ใช้เก็บบันทึกข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรม

การพัฒนาโปรแกรมเสริมที่ใช้เก็บข้อมูลการเปลี่ยนแปลงของรหัสต้นฉบับของโปรแกรมระหว่างการทำปฏิบัติการ มีรายละเอียดของคลาสที่พัฒนาขึ้น ดังต่อไปนี้

Package : javasnapshot

ใน package javasnapshot ประกอบด้วยคลาส 2 คลาส คือ

- คลาส Activator ทำหน้าที่เป็นคลาสเริ่มต้นที่โปรแกรม Eclipse จะเรียกใช้งานโปรแกรมเสริม โดยดูจากจุดเชื่อมต่อที่ประกาศเรียกใช้งานไว้

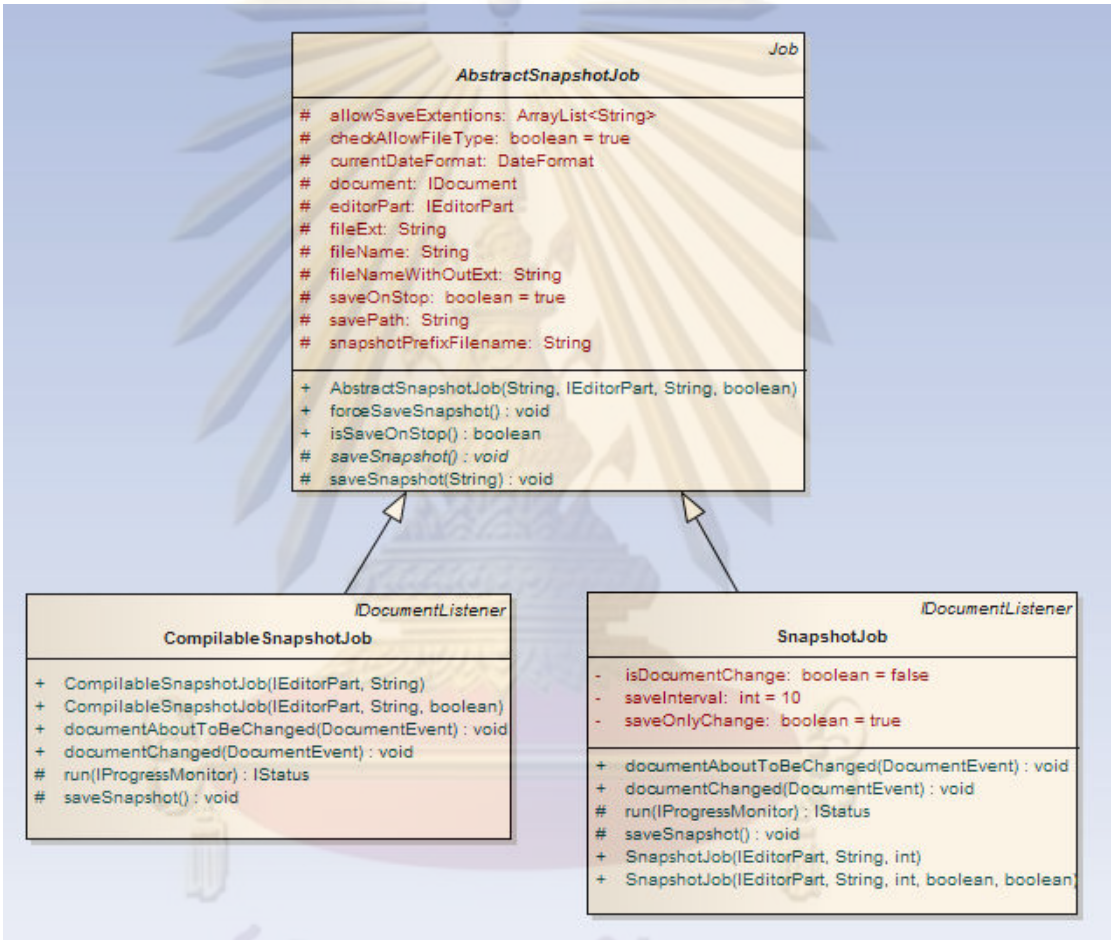
- คลาส JavasnpshotLog ทำหน้าที่เป็นคลาสที่ใช้เก็บบันทึกการทำงาน (log) ของโปรแกรมเสริม

Package : javasnapshot.handlers

ใน package javasnapshot.handlers ประกอบด้วยคลาส 2 คลาส คือ

- คลาส RecordHandler ทำหน้าที่รับเหตุการณ์ที่เกิดขึ้นเมื่อมีการกดปุ่มบันทึกของโปรแกรมเสริมเพื่อให้โปรแกรมเสริมเริ่มบันทึกการทำงาน หรือหยุดบันทึกการทำงาน
- คลาส SettingHandler ทำหน้าที่รับเหตุการณ์ทีเกิดขึ้นเมื่อมีการกดปุ่มตั้งค่าการทำงานของโปรแกรมเสริม

Package : javasnapshot.jobs

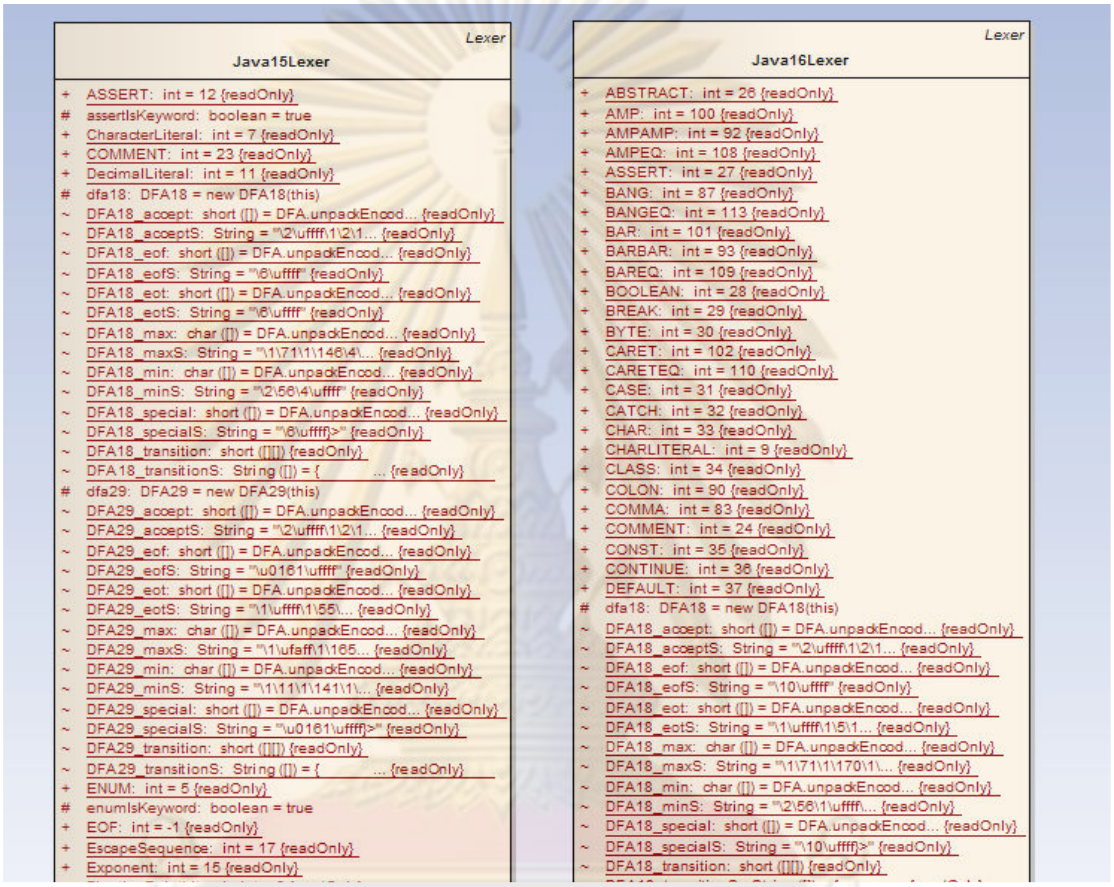


ใน package javasnapshot.jobs ประกอบด้วยคลาส 3 คลาส คือ

- คลาส AbstractSnapshotJob ทำหน้าที่เป็นคลาสต้นแบบของงานในการตรวจสอบรหัสต้นฉบับของโปรแกรมและบันทึกข้อมูลลงแหล่งจัดเก็บ
- คลาส CompilableSnapshotJob ทำหน้าที่เป็นคลาสที่ใช้ในการรับรหัสต้นฉบับของโปรแกรมเมื่อเกิดเหตุการณ์การพิมพ์เพิ่ม แก้ไข หรือลบ ภายในโปรแกรม Eclipse มาตรวจสอบความถูกต้องของรหัสต้นฉบับ ถ้าถูกต้องถึงจะบันทึกข้อมูลลงแหล่งจัดเก็บ

- คลาส SnapshotJob ทำหน้าที่เป็นคลาสที่ใช้ในการรับรหัสต้นฉบับของโปรแกรม เมื่อเกิดเหตุการณ์การพิมพ์เพิ่ม แก้ไข หรือลบ ภายในโปรแกรม Eclipse และบันทึกข้อมูลลงแหล่งจัดเก็บ

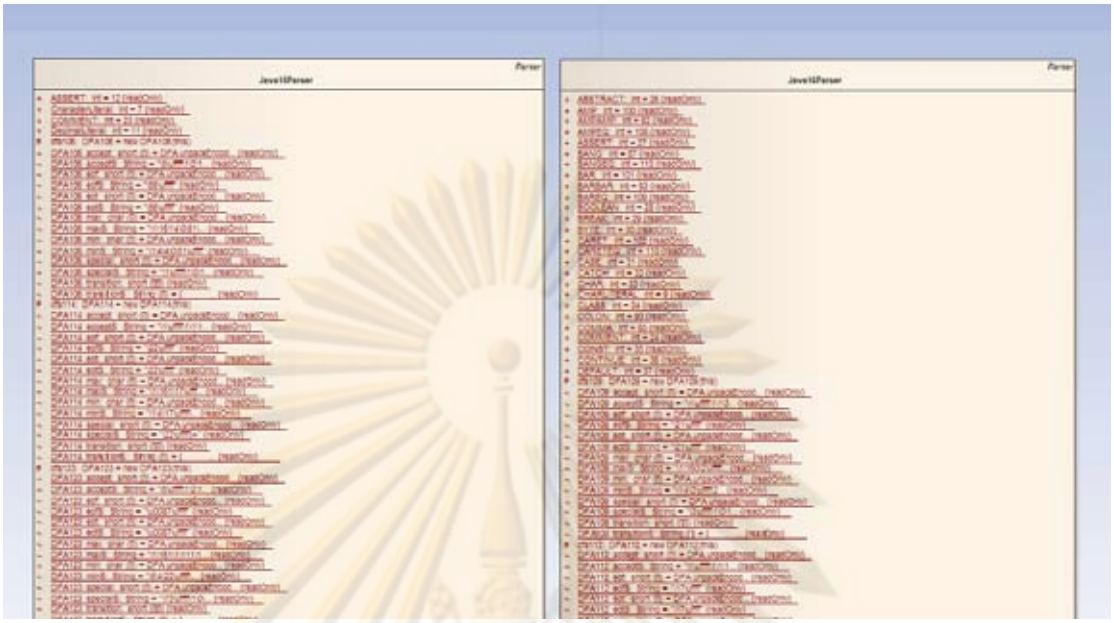
Package : javasnapshot.lexer



ใน package javasnapshot.lexer ประกอบด้วยคลาส 2 คลาส คือ

- คลาส Java15Lexer ทำหน้าที่เป็นคลาสที่ใช้ในการทำ Lexical Analysis ตามข้อกำหนดของโปรแกรมภาษาจาวา รุ่น 1.5
- คลาส Java16Lexer ทำหน้าที่เป็นคลาสที่ใช้ในการทำ Lexical Analysis ตามข้อกำหนดของโปรแกรมภาษาจาวา รุ่น 1.6

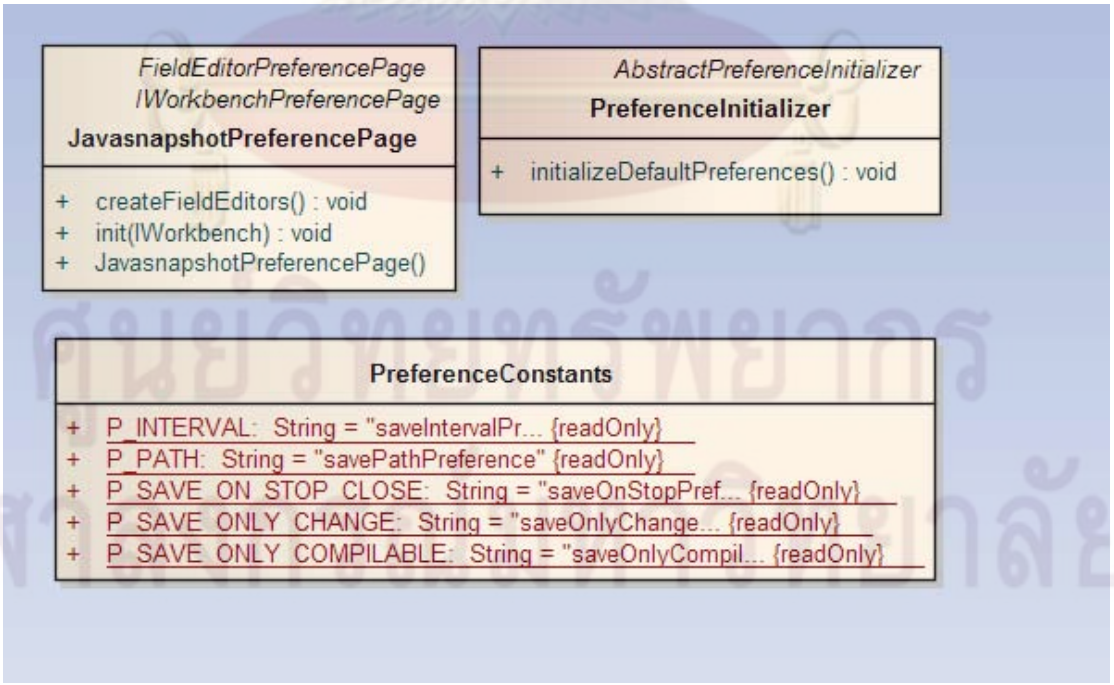
Package : javasnapshot.parser



ใน package javasnapshot.parser ประกอบด้วยคลาส 2 คลาส คือ

- คลาส Java15Parser ทำหน้าที่เป็นคลาสที่ใช้ในการทำตรวจสอบความถูกต้องของรหัสต้นฉบับของโปรแกรมตามข้อกำหนดของโปรแกรมภาษาจาวา รุ่น 1.5
- คลาส Java16Parser ทำหน้าที่เป็นคลาสที่ใช้ในการทำตรวจสอบความถูกต้องของรหัสต้นฉบับของโปรแกรมตามข้อกำหนดของโปรแกรมภาษาจาวา รุ่น 1.6

Package : javasnapshot.preferences



ใน package javasnapshot.preferences ประกอบด้วยคลาส 3 คลาส คือ

- คลาส PreferenceInitializer ทำหน้าที่เป็นคลาสที่ใช้ในการอ่านข้อมูลการตั้งค่าที่บันทึกไว้ ซึ่งค่าต่าง ๆ ที่ตั้งค่าไว้ในหน้าตั้งการตั้งค่าของโปรแกรมเสริม จะถูกเก็บบันทึกไว้ในส่วนเก็บข้อมูลภายในโปรแกรม Eclipse
- คลาส JavasnapshotPreferencePage ทำหน้าที่เป็นคลาสที่ใช้ในการสร้างหน้าต่างการกำหนดค่าต่าง ๆ ของโปรแกรมเสริม และรับเหตุการณ์ที่เกิดขึ้นในหน้าต่างนี้จากการใช้งานของผู้ทดสอบ
- คลาส PreferenceConstants ทำหน้าเป็นคลาสอัตถประโยชน์ที่ใช้เก็บค่าคงที่ที่ใช้อ้างอิงค่าต่าง ๆ ที่สามารถกำหนดได้ของโปรแกรมเสริม

ภาคผนวก ค

การทดสอบผลการเปรียบเทียบความละม้ายของโปรแกรม Plaggie

การทดสอบผลเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรม โดยเพิ่มคำสั่งทีละคำสั่งจนจบโปรแกรม ในส่วนพื้นที่สีเทาเป็นส่วนโครงของรหัสต้นฉบับของโปรแกรมเริ่มต้น และจะเพิ่มคำสั่งเข้าไปทีละคำสั่งตามลำดับที่ระบุ จนสิ้นสุด โดยเปรียบเทียบความละม้ายทุกครั้งที่เพิ่มคำสั่งเข้าไปเทียบกับรหัสต้นฉบับของโปรแกรม ณ จุดสิ้นสุด ได้ค่า % ความละม้าย ดังนี้

ผลการทดสอบชุดที่ 1

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
2	import java.util.Scanner;	58.33
	public class Area {	
	public static void main(String[] args) {	33.33
1	Scanner sc = new Scanner(System.in);	50.00
3	System.out.println("รัศมี = ");	66.67
4	double r = sc.nextDouble();	83.33
5	double area = Math.PI * r * r;	91.67
6	System.out.println("พื้นที่ = "+area);	100.00
	}	
	}	

ผลการทดสอบชุดที่ 2

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
2	import java.io.File;	25.81
4	import java.io.FileNotFoundException;	32.26
3	import java.util.Scanner;	29.03
	public class AverageScore {	
	public static void main(String[] args) throws FileNotFoundException {	12.90
1	Scanner sc = new Scanner(new File("scores.txt"));	22.58
5	double sum = 0;	35.48
6	int n = 0;	38.71
7	while(true){	51.61
8	if(!sc.hasNext()) break;	64.52
9	String t = sc.nextLine();	70.97
10	String t1 = t.substring(10, t.length());	80.65

11	double p = Double.parseDouble(t1.trim());	90.32
12	sum = sum + p;	93.55
13	n++;	96.77
7	}	
14	System.out.println("คะแนนเฉลี่ย = " + (sum/n));	100.00
	}	
	}	

ผลการทดสอบชุดที่ 3

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
	public class BankAccount {	8.00
1	private double balance = 0;	12.00
2	public void deposit(double amt){	20.00
3	if(amt>0) this.balance = this.balance +amt;	32.00
2	}	
4	public void withdraw(double amt){	40.00
5	if(amt>0 && this.balance>amt)	52.00
5	this.balance = this.balance - amt;	
4	}	
6	public double getBalance(){	60.00
7	return this.balance;	64.00
6	}	
8	public void transfer(BankAccount a, double amt){	72.00
9	if(amt>0 && a.getBalance()>=amt){	92.00
	a.withdraw(amt);	96.00
	this.deposit(amt);	100.00
9	}	
8	}	
	}	

ผลการทดสอบชุดที่ 4

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
	public class Circle {	
	public static void main(String[] args) {	30.77
1	String s1 = "เส้นรอบวง = ";	38.46
2	String s2 = "พื้นที่ = ";	46.15

3	double radius = 5;	53.85
4	double circumference, area;	69.23
5	circumference = 2 * 3.14159 * radius;	76.92
6	area = 3.14159 * radius * radius;	84.62
7	System.out.println(s1 + circumference);	92.31
8	System.out.println(s2 + area);	100.00
	}	
	}	

ผลการทดสอบชุดที่ 5

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
2	import java.util.Scanner;	20.59
	public class EAN13 {	
	public static void main(String[] args) {	11.76
1	Scanner kb = new Scanner(System.in);	17.65
3	System.out.print("ตัวเลข 12 หลัก = ");	23.53
4	String d = kb.nextLine();	29.41
5	int s = 0, k = 0;	35.29
6	while(true){	47.06
7	int v = Integer.parseInt(d.substring(k, k+1));	55.88
8	if(k%2 ==0){	67.65
9	s = s+3*v;	70.59
8,10	}else{	82.35
11	s = s+v;	85.29
10	}	
12	if(++k>11) break;	97.06
6	}	
13	System.out.println("Check Digit คือ "+ (10 -(s%10))%10);	100.00
	}	
	}	

ผลการทดสอบชุดที่ 6

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
	public class Factorial {	
	public static void main(String[] args) {	28.57
5	System.out.println(fac(3));	100.00

	}	
1	public static int fac(int n){	43.86
2	if(n==0) return 1;	64.29
3	int f = fac(n-1);	78.57
4	return n*f;	85.71
1	}	
	}	

ผลการทดสอบชุดที่ 7

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
	public class HelloWorld {	
	public static void main(String[] args) {	57.14
1	System.out.print("Hello ");	71.43
2	System.out.print("World ");	85.71
3	System.out.print("A");	100.00
	}	
	}	

ผลการทดสอบชุดที่ 8

ลำดับ	รหัสต้นฉบับของโปรแกรม	% ที่ได้
2	import java.util.Scanner;	43.75
	public class Keyboard {	
	public static void main(String[] args) {	
1	Scanner kb = new Scanner(System.in);	37.50
3	System.out.print("a = ");	50.00
4	double a = kb.nextDouble();	62.50
5	System.out.print("b = ");	68.75
6	double b = kb.nextDouble();	81.25
7	double max = Math.max(a, b);	93.75
8	System.out.println("ตัวมาก = "+max);	100.00
	}	
	}	

ภาคผนวก ง

ผลการเปรียบเทียบความละม้ายของรหัสต้นฉบับของโปรแกรมของผู้ทดสอบกับชุดเฉลย

การเก็บข้อมูลการทำแบบทดสอบ ได้ผลการทำแบบทดสอบทั้งหมด 70 ชุดข้อมูล ดังแสดงรายละเอียดในตารางต่อไปนี้

รหัสต้นฉบับของโปรแกรมที่บันทึกได้ในแต่ละแบบทดสอบที่ผู้ทดสอบทำการทดสอบ

ผู้ทดสอบ	แบบทดสอบ ชุดที่ 1	แบบทดสอบ ชุดที่ 2	แบบทดสอบ ชุดที่ 3	แบบทดสอบ ชุดที่ 4	แบบทดสอบ ชุดที่ 5
คนที่ 1	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 2	บันทึกข้อมูล	ไม่บันทึก	ไม่บันทึก	ไม่บันทึก	ไม่บันทึก
คนที่ 3	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 4	บันทึกข้อมูล	ไม่บันทึก	ไม่บันทึก	ไม่บันทึก	ไม่บันทึก
คนที่ 5	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 6	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 7	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 8	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 9	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 10	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 11	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	ไม่บันทึก	ไม่บันทึก
คนที่ 12	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 13	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 14	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 15	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล
คนที่ 16	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล	บันทึกข้อมูล

ในการทดสอบที่ไม่มีการบันทึกข้อมูลเนื่องจาก ผู้ทดสอบทำไม่ได้ จึงเว้นไว้ไม่ได้ทำในบางข้อ ทำให้ไม่มีข้อมูลของแบบทดสอบเก็บบันทึกไว้ หลังการทดสอบข้อมูลที่บันทึกไว้จะถูกนำไปเปรียบเทียบกับชุดเฉลยสำหรับแต่ละแบบทดสอบ ระยะเวลาที่ใช้ในการเปรียบเทียบรหัสต้นฉบับของโปรแกรมกับชุดเฉลยที่มีสำหรับแต่ละแบบทดสอบ เป็นดังนี้

ระยะเวลาที่ใช้ในการเปรียบเทียบรหัสต้นฉบับของโปรแกรมที่บันทึกได้กับชุดเฉลยในแต่ละแบบทดสอบที่ผู้ทดสอบทำการทดสอบ

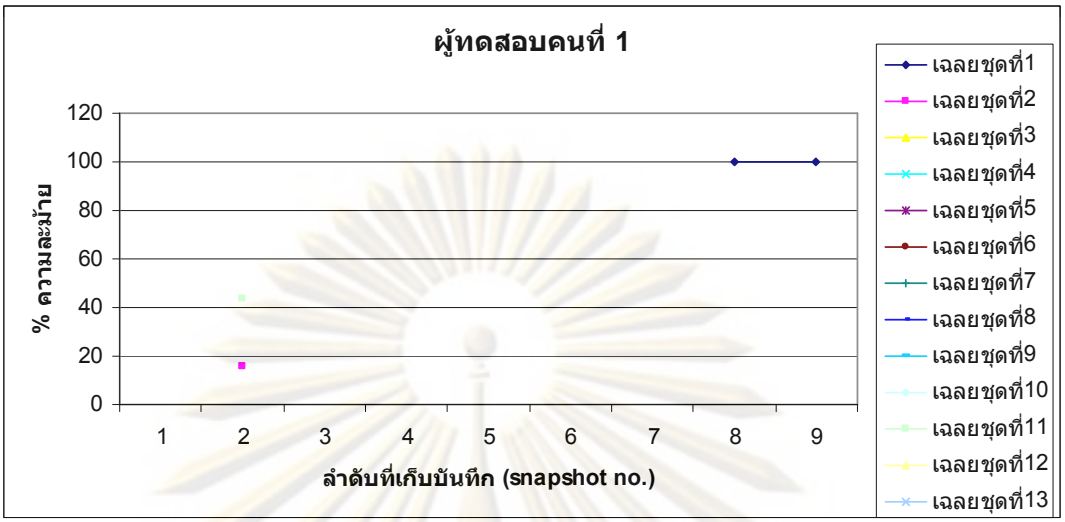
ผู้ทดสอบ	แบบทดสอบ ชุดที่ 1 (วินาที)	แบบทดสอบ ชุดที่ 2 (วินาที)	แบบทดสอบ ชุดที่ 3 (วินาที)	แบบทดสอบ ชุดที่ 4 (วินาที)	แบบทดสอบ ชุดที่ 5 (วินาที)
คนที่ 1	4.08	9.38	2.36	2.09	2.42
คนที่ 2	6.17	0	0	0	0
คนที่ 3	3.92	8.17	2.14	2.38	2.11
คนที่ 4	4.03	0	0	0	0
คนที่ 5	4.81	10.55	2.77	2.03	2.24
คนที่ 6	6.1	6.75	2.39	2.38	2.55
คนที่ 7	4.8	7.5	3.21	2.39	2.2
คนที่ 8	4.15	8.29	2.09	2.42	2.19
คนที่ 9	4.08	5.58	2.68	1.93	1.86
คนที่ 10	6.95	7.12	2.17	2.36	2.52
คนที่ 11	4.06	9.85	7.2	0	0
คนที่ 12	4.59	6.58	2.66	2.54	2.11
คนที่ 13	4.67	11.56	2.79	2.4	2.52
คนที่ 14	5.27	9.45	2.45	1.78	2.99
คนที่ 15	4.37	7.25	2.71	2.08	2.57
คนที่ 16	4.44	7.78	2.14	1.98	2.34
ค่าเฉลี่ย	4.78	7.24	2.49	1.80	1.91
ค่ามากสุด	3.92	5.58	2.09	1.78	1.86
ค่าน้อยสุด	6.95	11.56	7.2	2.54	2.99
เวลาที่ใช้ทั้งหมด	76.49	115.81	39.76	28.76	30.62

ผลการเปรียบเทียบความละม้ายระหว่างรหัสต้นฉบับของโปรแกรมที่จัดเก็บได้กับของชุดเฉลยสำหรับแต่ละแบบทดสอบ มีรายละเอียดดังนี้

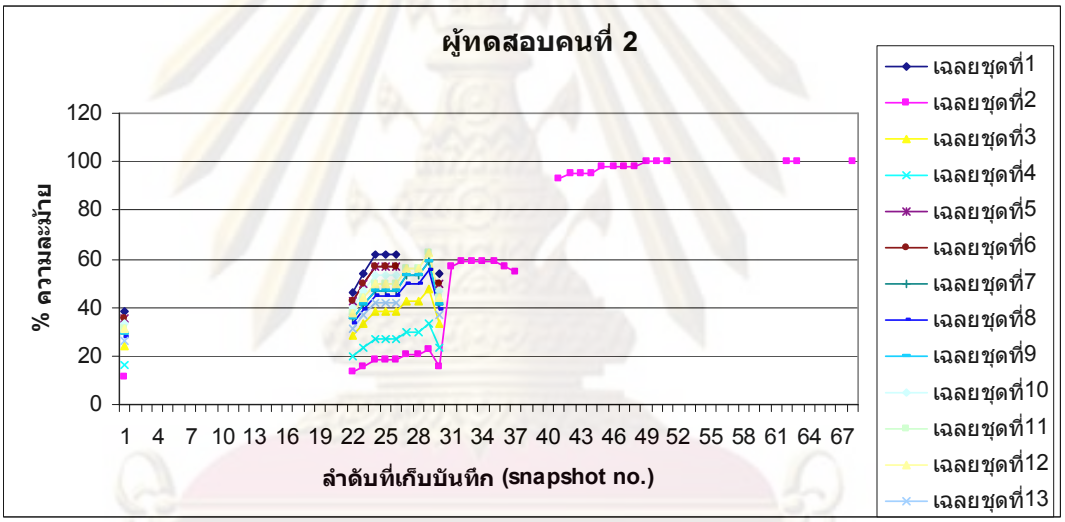
<u>แบบทดสอบที่ 1</u>

ผลการเปรียบเทียบความละม้ายของแบบทดสอบชุดที่ 1 กับของชุดเฉลย

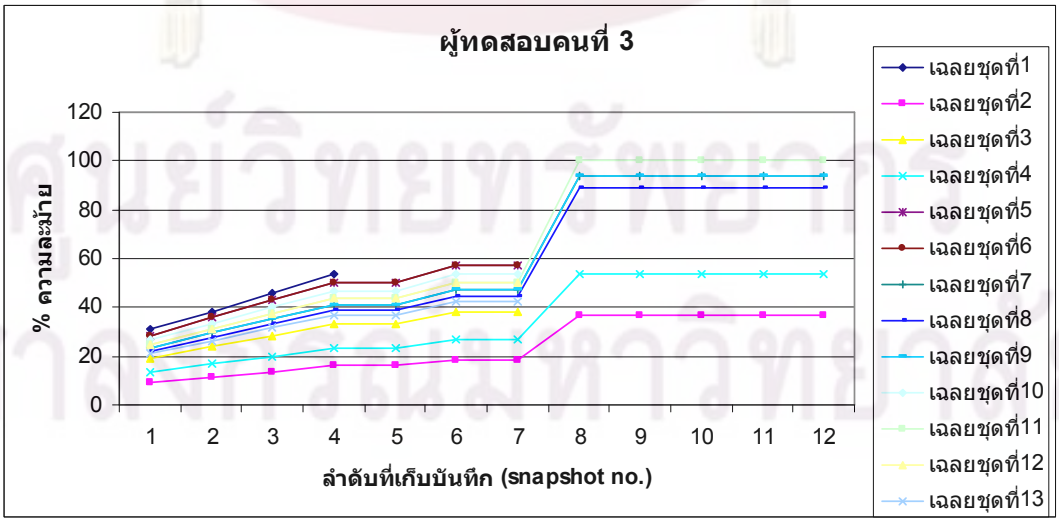
ผู้ทดสอบ	จำนวนที่บันทึกได้ (snapshots)	เวลาที่ใช้ (นาที)	ลำดับที่ละม้ายที่สุด	% ความละม้ายมากที่สุดที่ได้	ชุดเฉลยที่ละม้ายมากที่สุด
คนที่ 1	9	1.27	8,9	100	1
คนที่ 2	68	20.25	49,62,68	100	2
คนที่ 3	12	2.38	8 - 12	100	11
คนที่ 4	12	3.06	11	100	3
คนที่ 5	33	8.34	23	100	9
คนที่ 6	64	11.46	46,64	100	4
คนที่ 7	31	6.32	14 (เฉลยชุดที่ 11) 22 (เฉลยชุดที่ 8)	100	8,11
คนที่ 8	16	2.58	14	96.43	6
คนที่ 9	10	7.54	9	100	10
คนที่ 10	78	13.52	35	45.45	13
คนที่ 11	11	3.14	10	100	10
คนที่ 12	25	3.11	34	100	10
คนที่ 13	31	5.28	28	100	9
คนที่ 14	39	12.29	24	43.18	2
คนที่ 15	14	3.01	7	100	6
คนที่ 16	19	2.39	19	100	12



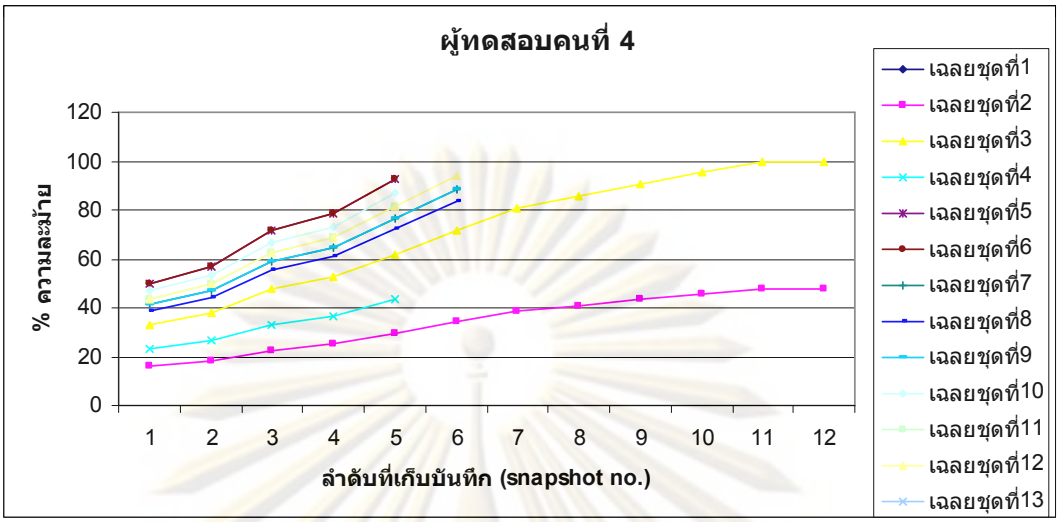
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 1 กับชุดเฉลย



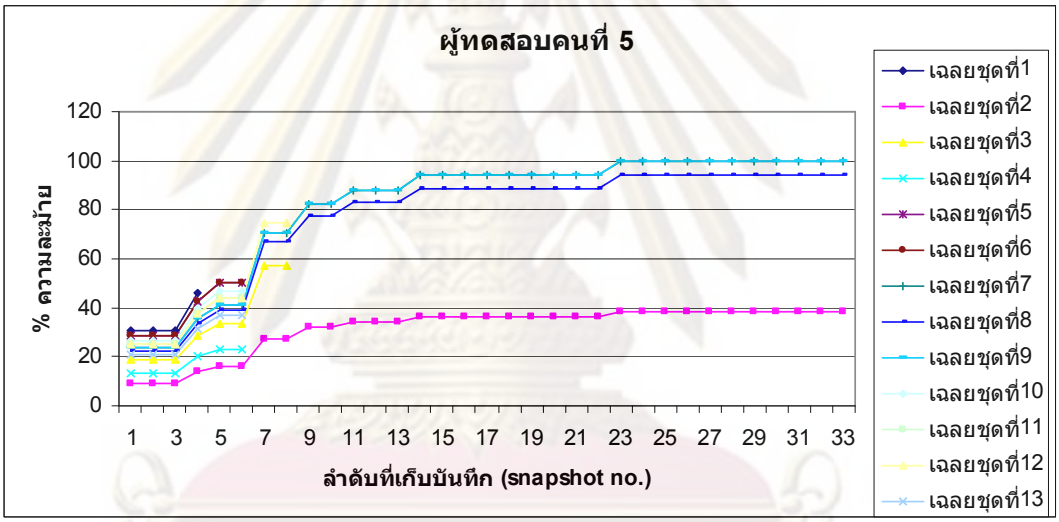
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 2 กับชุดเฉลย



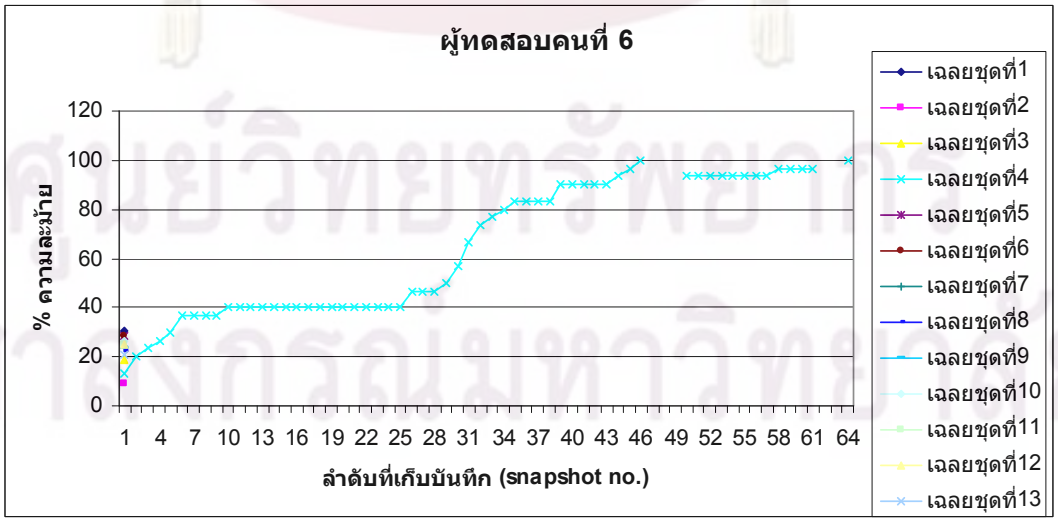
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 3 กับชุดเฉลย



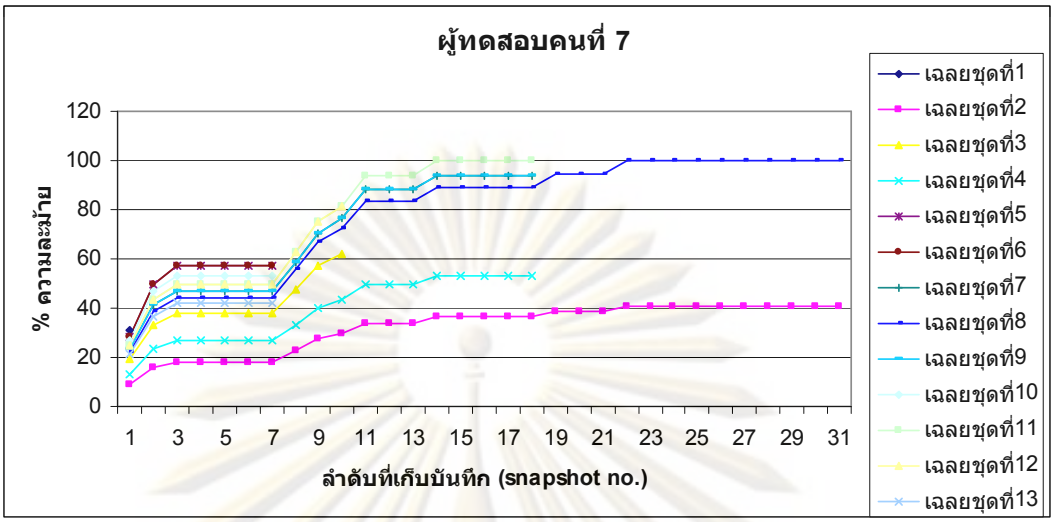
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 4 กับชุดเฉลย



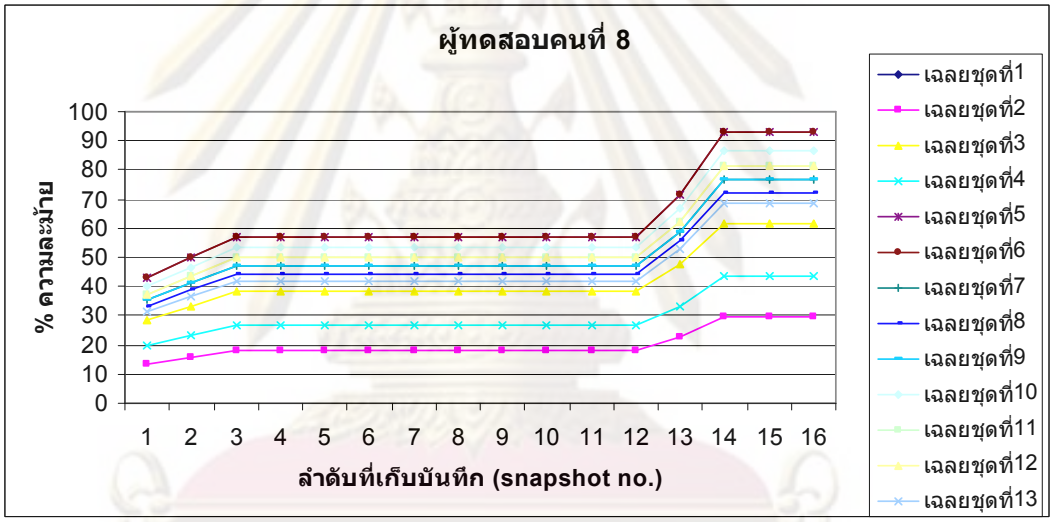
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 5 กับชุดเฉลย



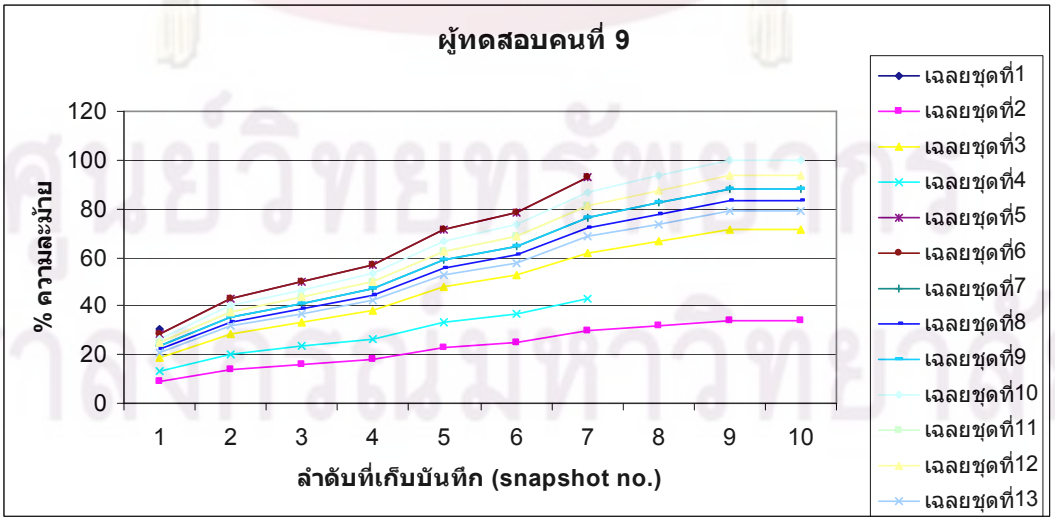
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 6 กับชุดเฉลย



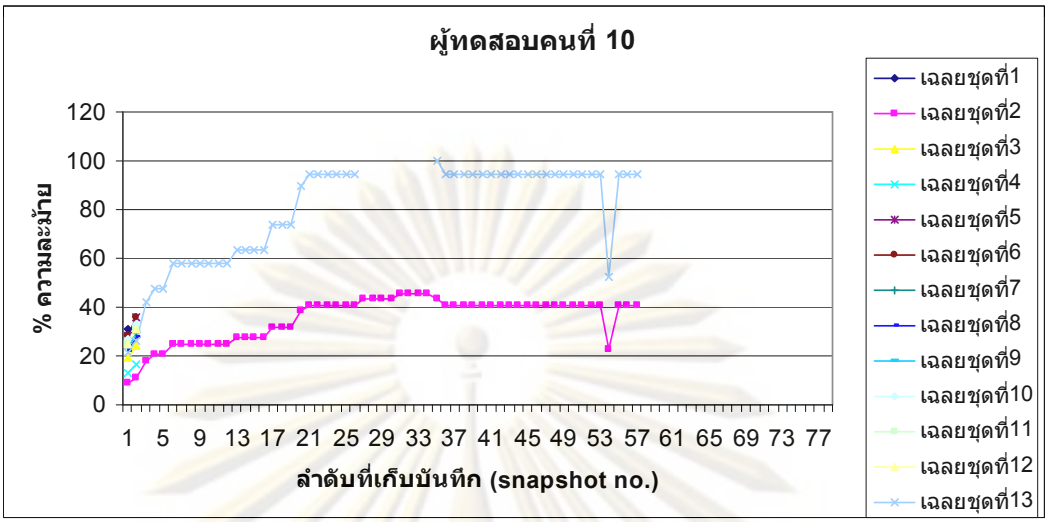
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 7 กับชุดเฉลย



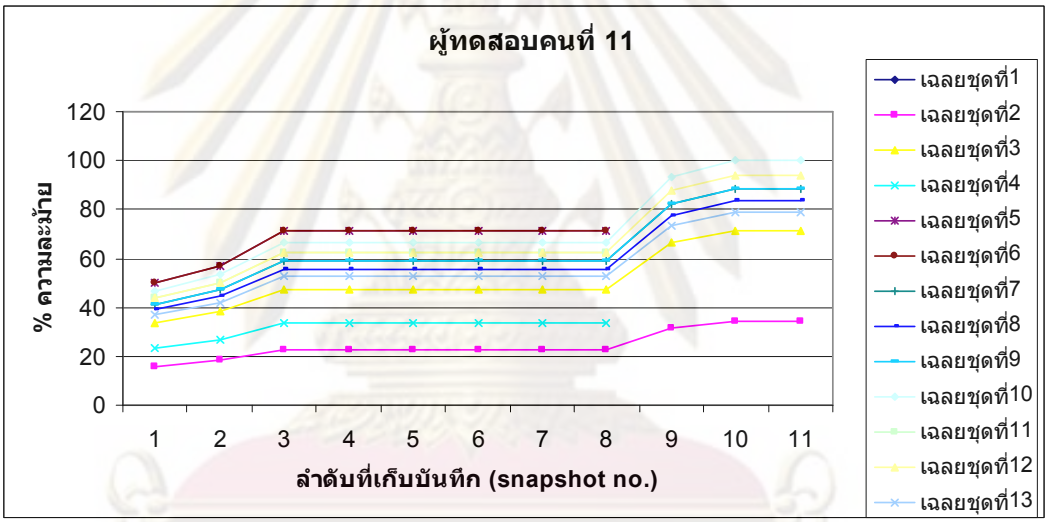
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 8 กับชุดเฉลย



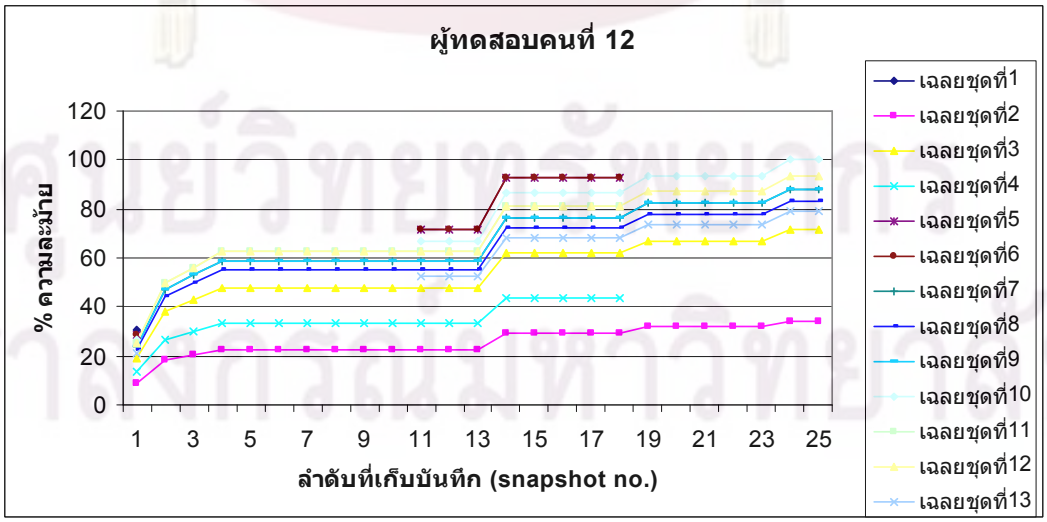
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 9 กับชุดเฉลย



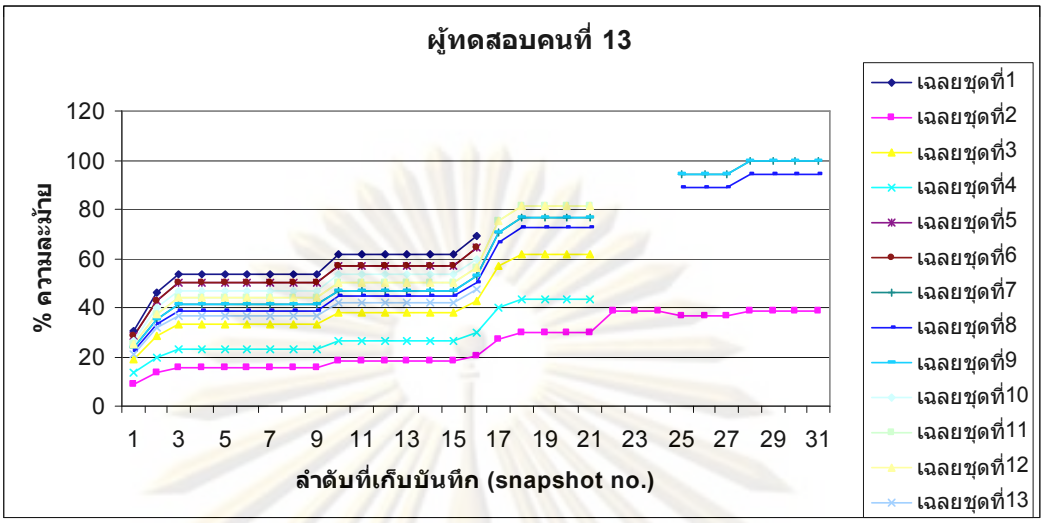
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 10 กับชุดเฉลย



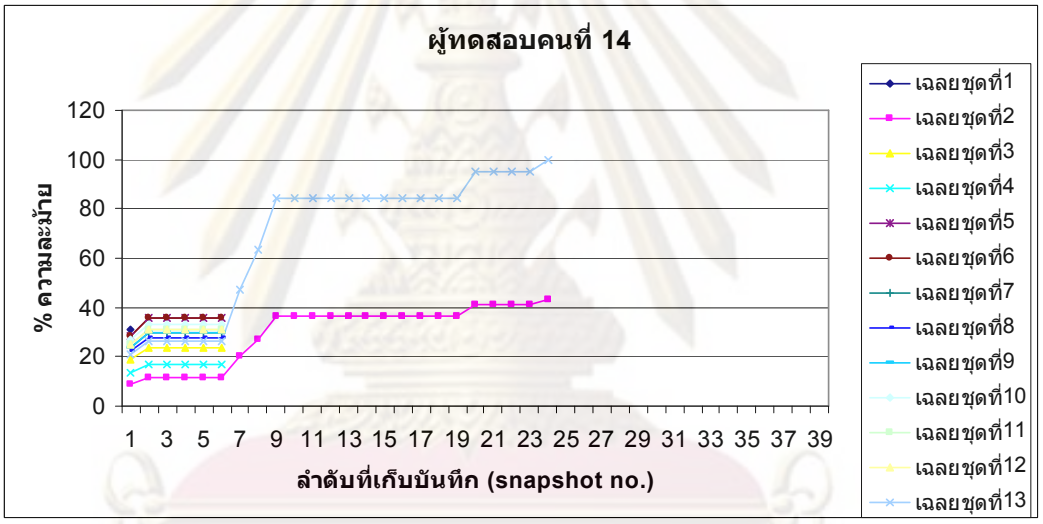
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 11 กับชุดเฉลย



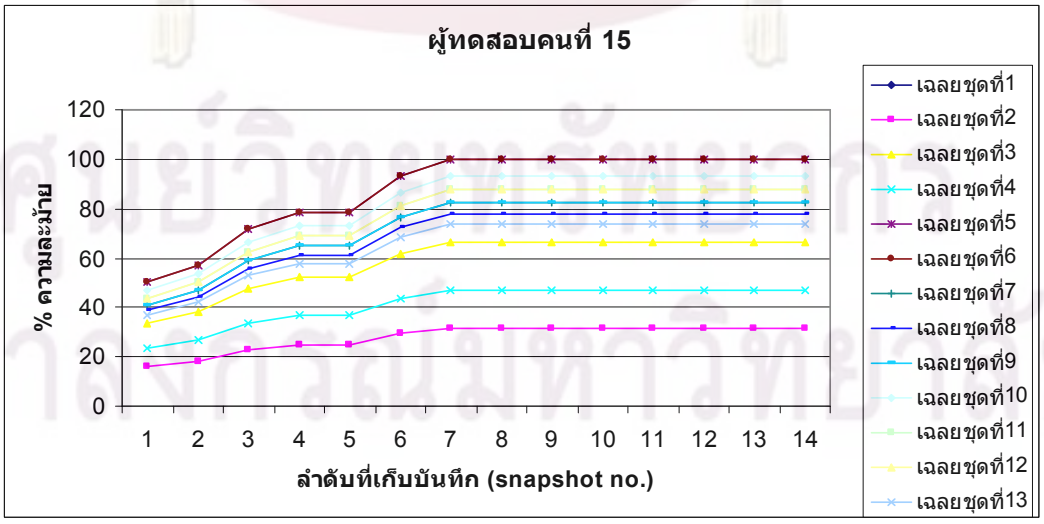
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 12 กับชุดเฉลย



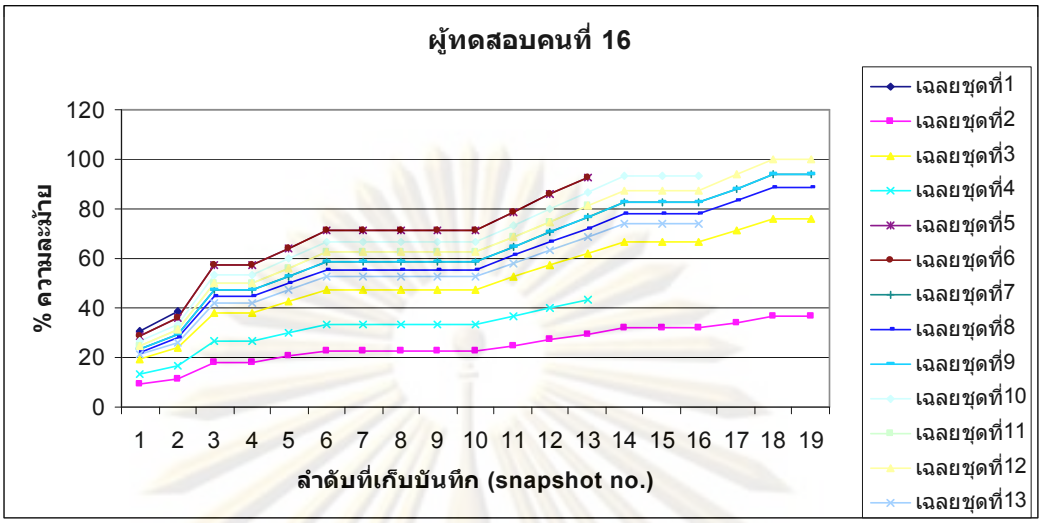
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 13 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 14 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 15 กับชุดเฉลย

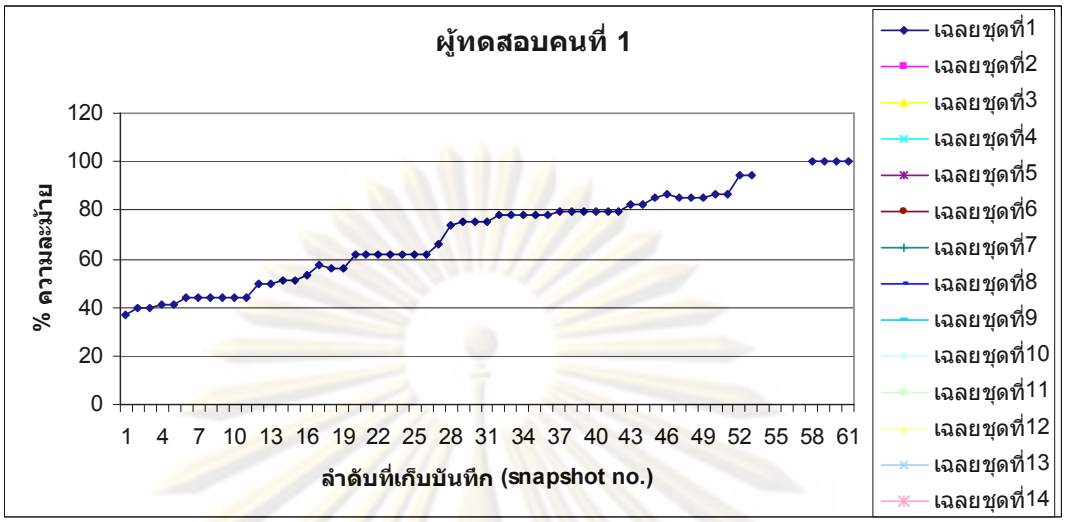


ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 1 ของผู้ทดสอบคนที่ 16 กับชุดเฉลย

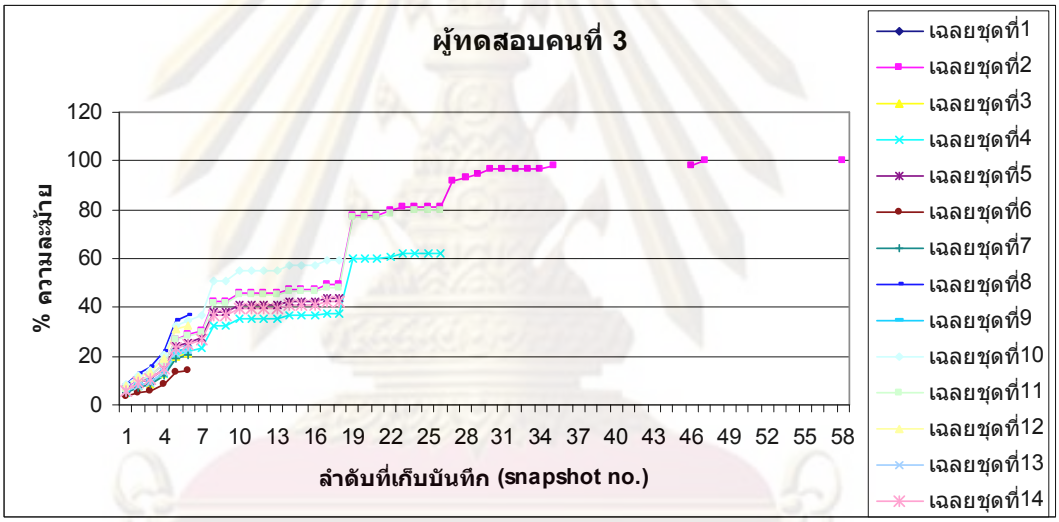
แบบทดสอบที่ 2

ผลการเปรียบเทียบความละม้ายของแบบทดสอบชุดที่ 2 กับของชุดเฉลย

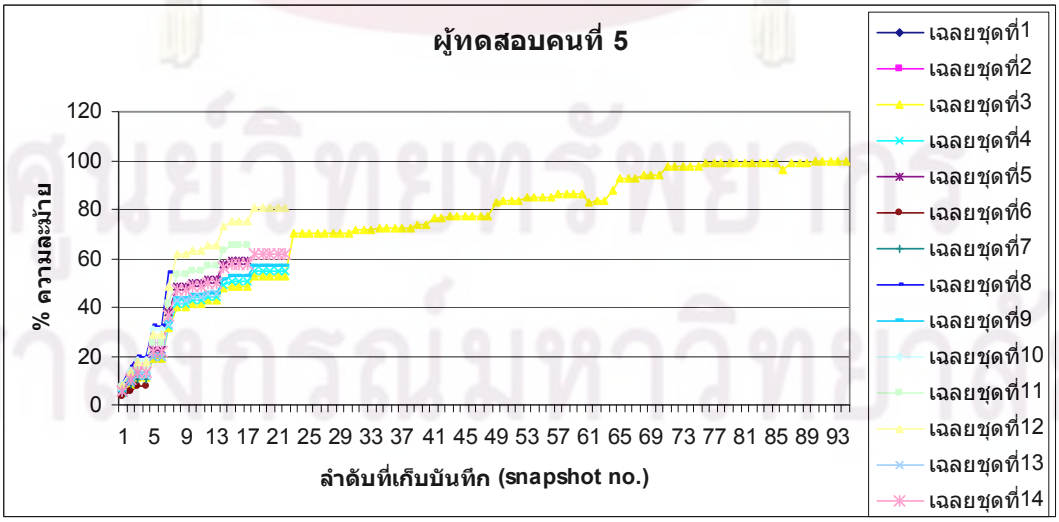
ผู้ทดสอบ	จำนวนที่บันทึกได้ (snapshots)	เวลาที่ใช้ (นาที)	ลำดับที่ละม้ายที่สุด	% ความละม้ายมากที่สุดที่ได้	ชุดเฉลยที่ละม้ายมากที่สุด
คนที่ 1	61	19.12	58	100	1
คนที่ 2	-	-	-	-	-
คนที่ 3	58	15.47	47	100	2
คนที่ 4	-	-	-	-	-
คนที่ 5	94	28.57	90	100	3
คนที่ 6	41	19.59	41	100	4
คนที่ 7	68	17.20	67	100	5
คนที่ 8	58	33.57	44,50 (เฉลยชุด 7) 58 (เฉลยชุด 6)	100	6,7
คนที่ 9	22	14.42	21	100	7
คนที่ 10	56	14.11	53	100	8
คนที่ 11	84	26.34	78	100	9
คนที่ 12	39	16.05	36	100	10
คนที่ 13	127	55.41	127	100	11
คนที่ 14	103	39.27	92	100	12
คนที่ 15	42	13.43	42	100	13
คนที่ 16	54	18.02	46	100	14



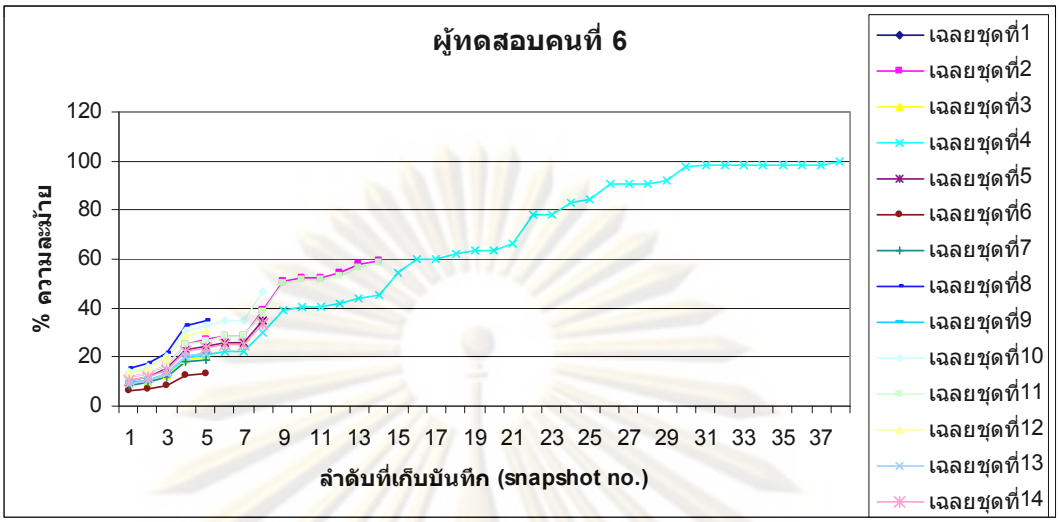
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 1 กับชุดเฉลย



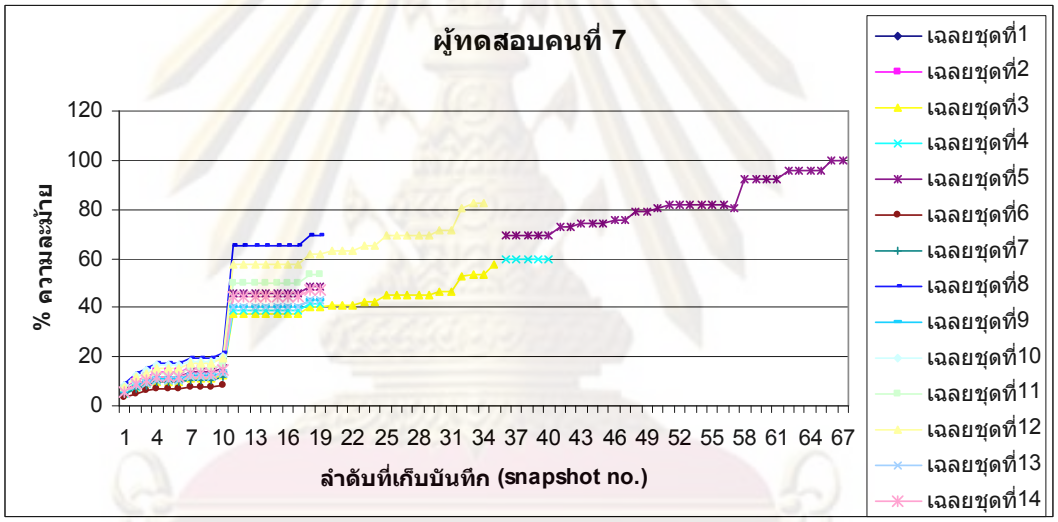
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 3 กับชุดเฉลย



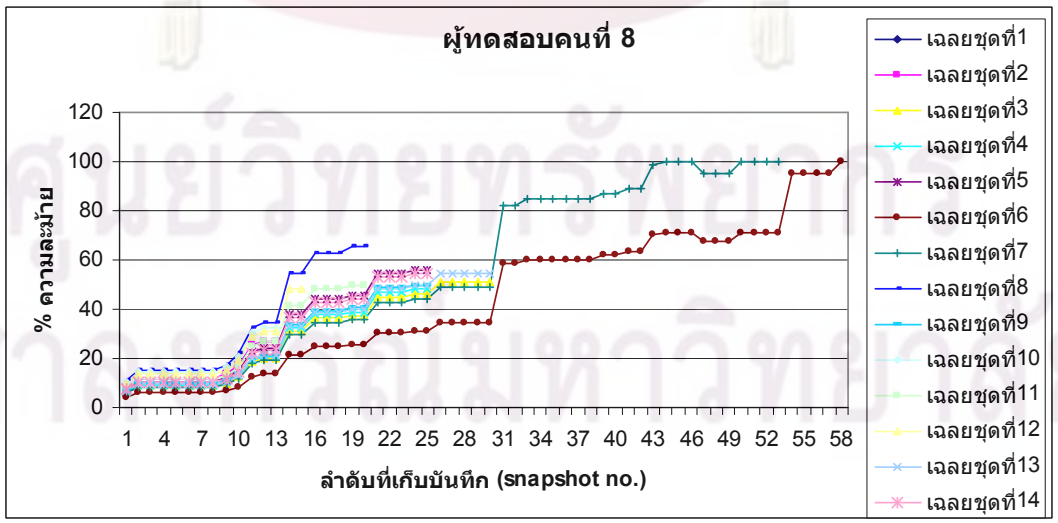
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 5 กับชุดเฉลย



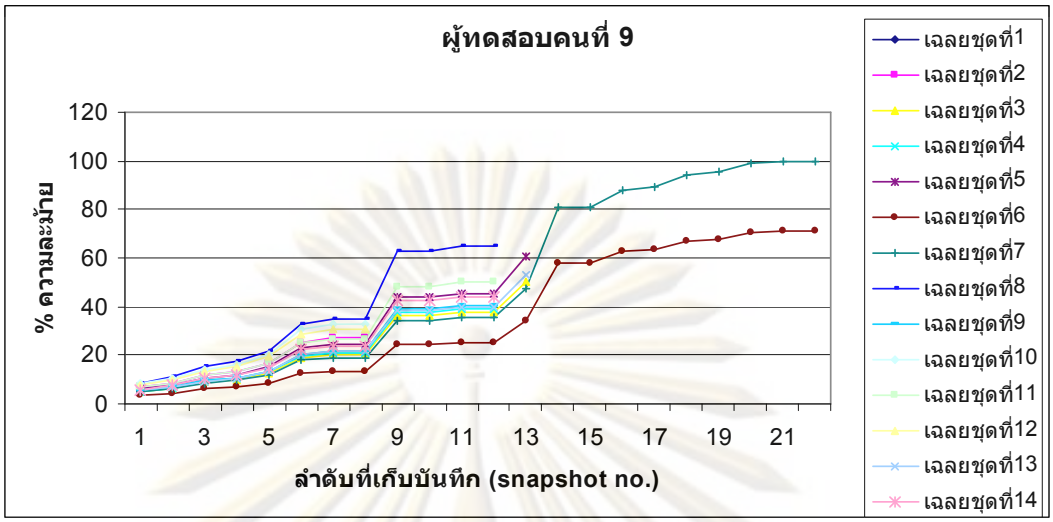
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 6 กับชุดเฉลย



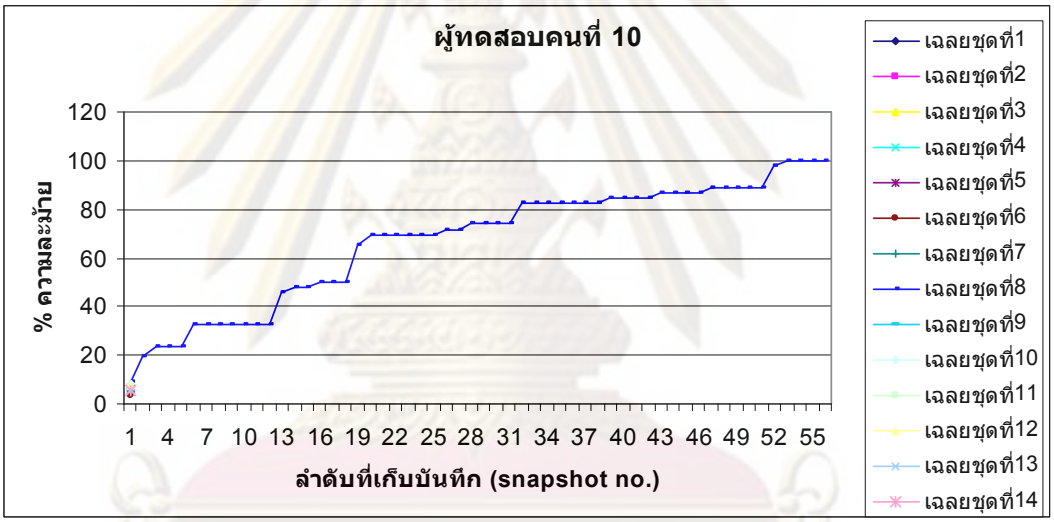
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 7 กับชุดเฉลย



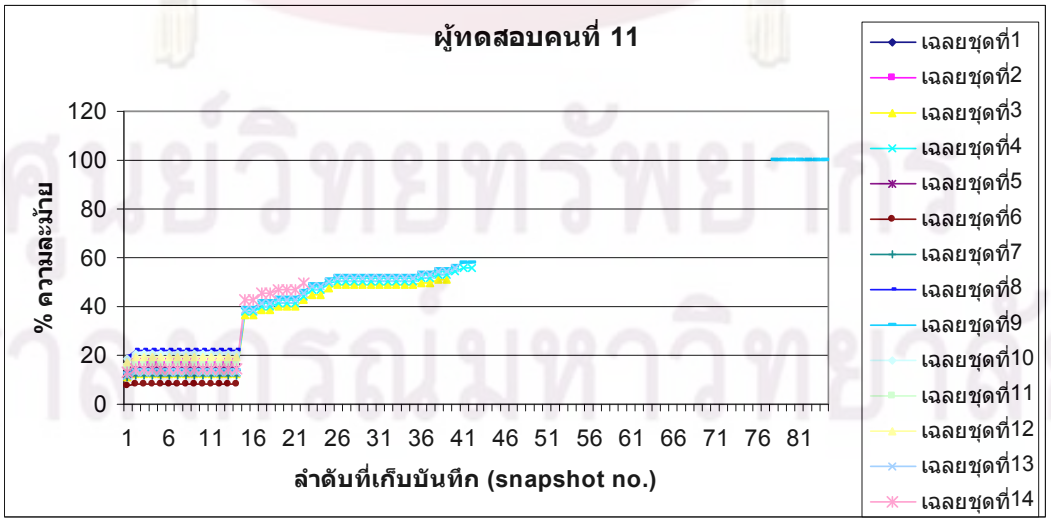
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 8 กับชุดเฉลย



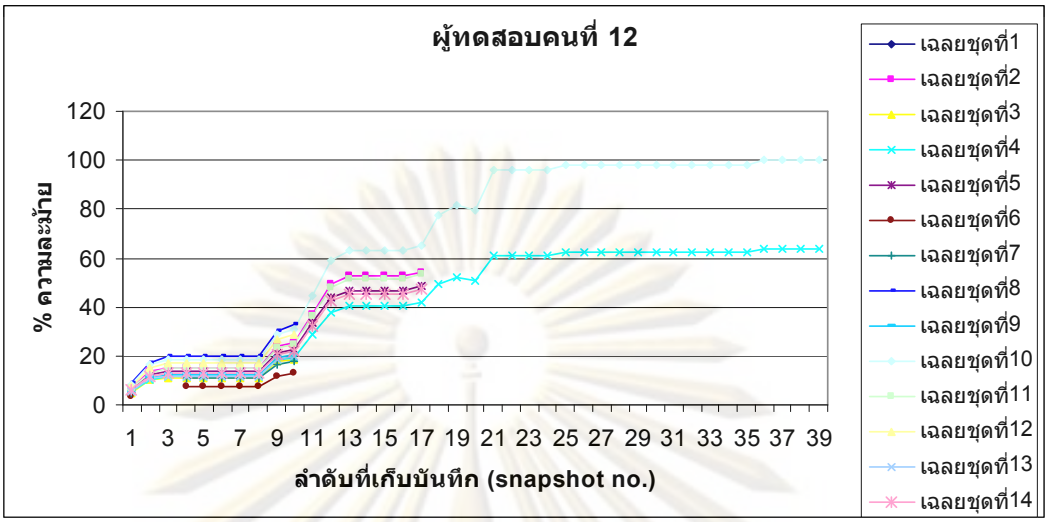
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 9 กับชุดเฉลย



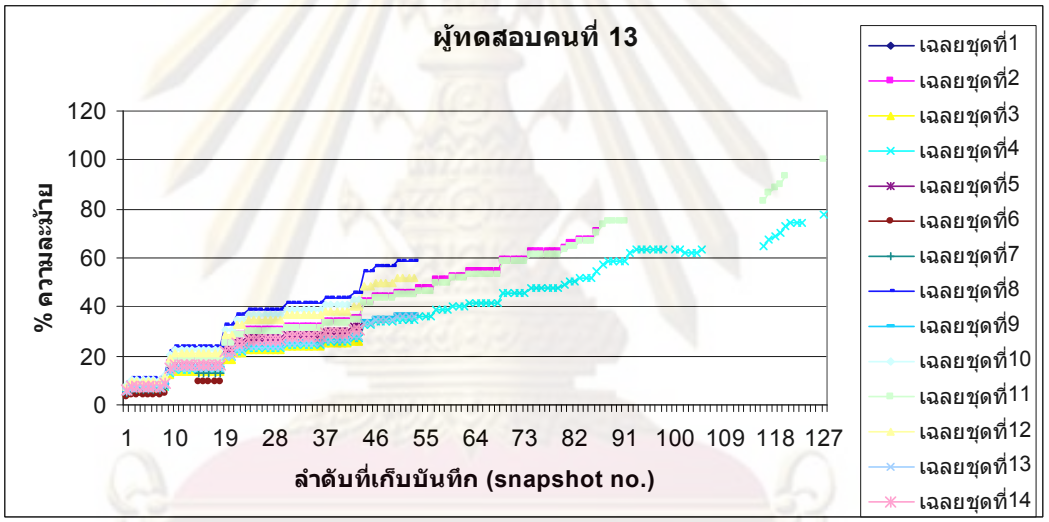
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 10 กับชุดเฉลย



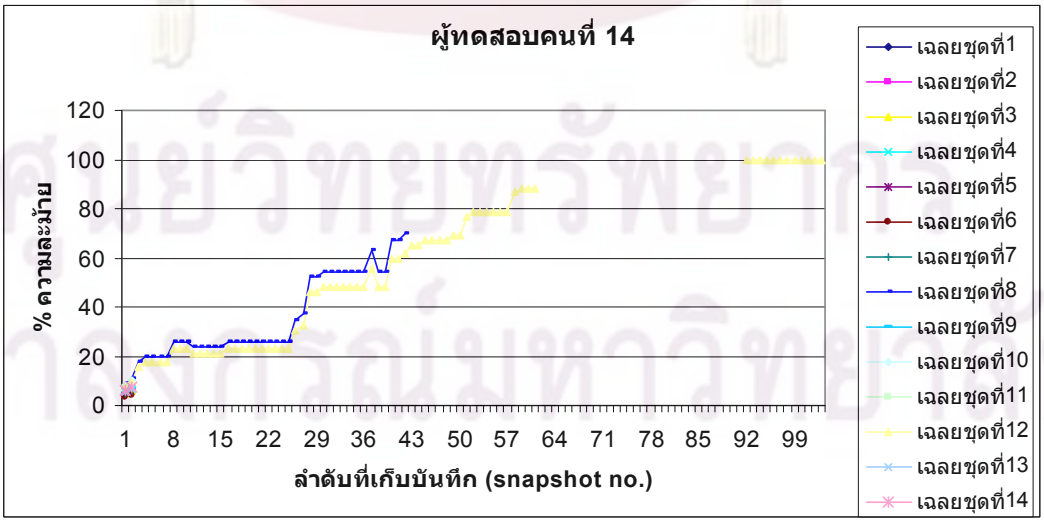
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 11 กับชุดเฉลย



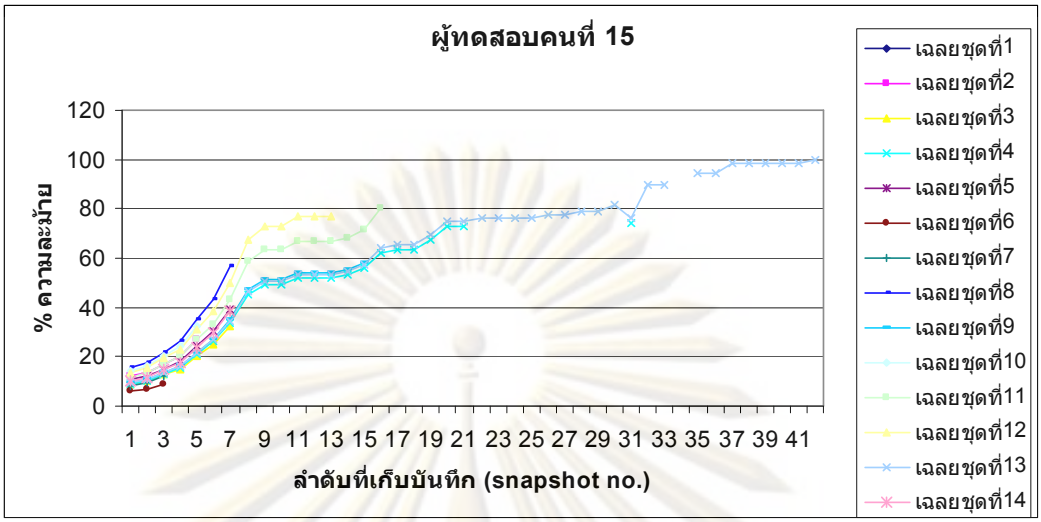
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 12 กับชุดเฉลย



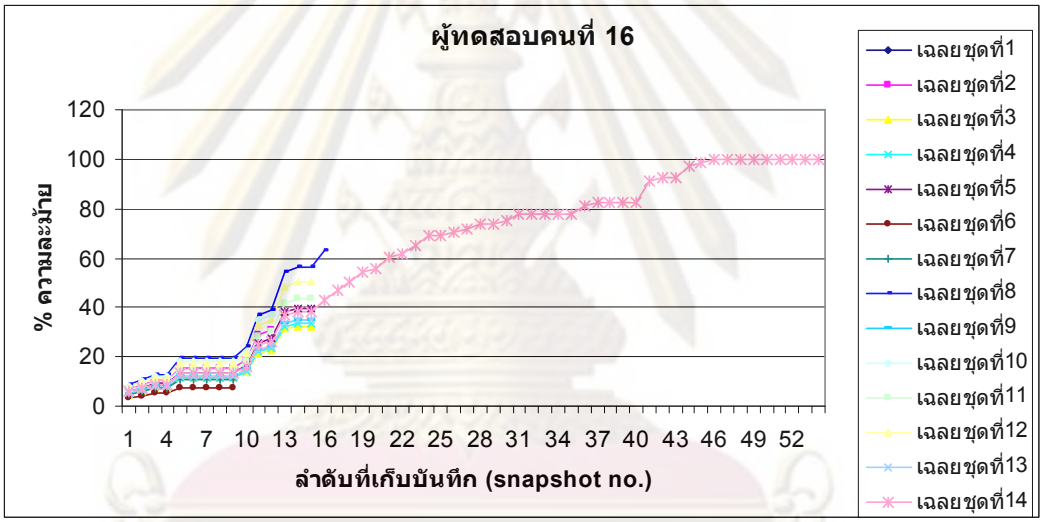
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 13 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 14 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 15 กับชุดเฉลย

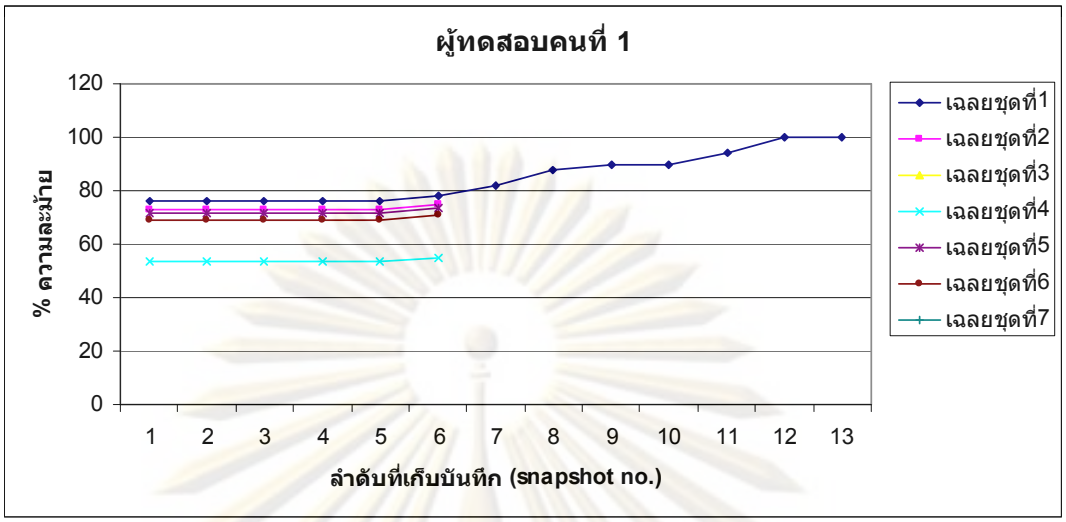


ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 2 ของผู้ทดสอบคนที่ 16 กับชุดเฉลย

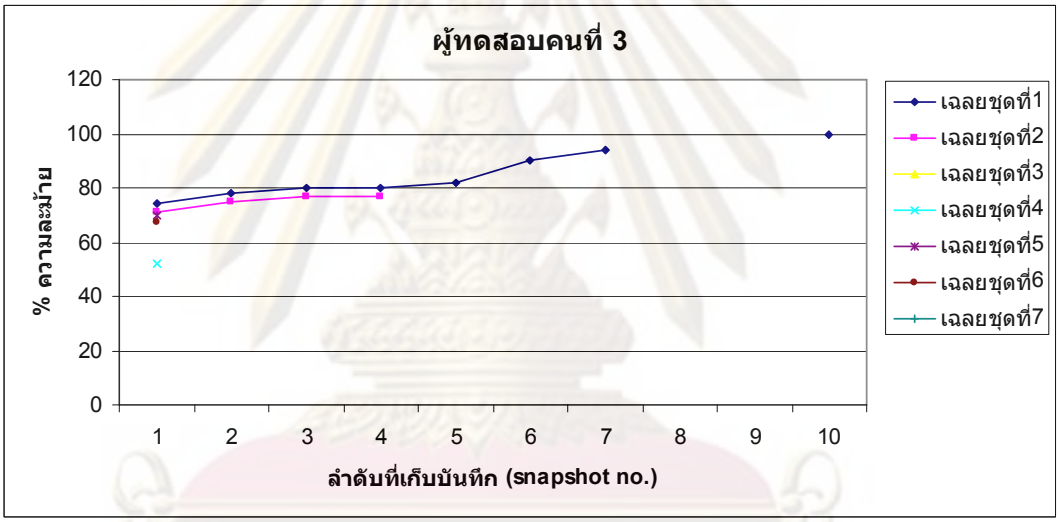
แบบทดสอบที่ 3

ผลการเปรียบเทียบความละม้ายของแบบทดสอบชุดที่ 3 กับของชุดเฉลย

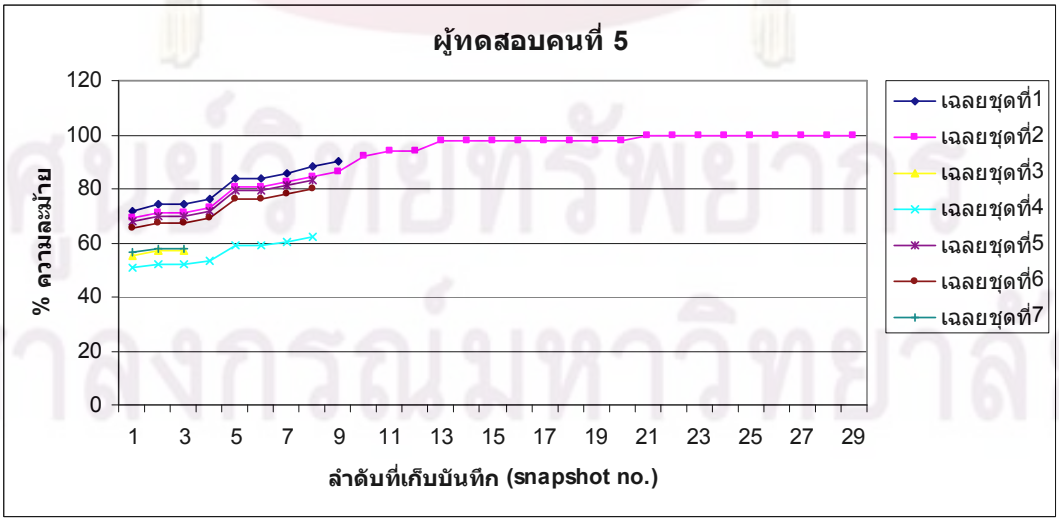
ผู้ทดสอบ	จำนวนที่บันทึกได้ (snapshots)	เวลาที่ใช้ (นาที)	ลำดับที่ละม้ายที่สุด	% ความละม้ายมากที่สุดที่ได้	ชุดเฉลยที่ละม้ายมากที่สุด
คนที่ 1	13	2.48	12	100	1
คนที่ 2	-	-	-	-	-
คนที่ 3	10	1.35	10	100	1
คนที่ 4	-	-	-	-	-
คนที่ 5	29	3.18	21	100	2
คนที่ 6	15	3.15	15	100	6
คนที่ 7	30	5.13	29	100	3
คนที่ 8	14	4.19	12	93	1
คนที่ 9	23	11.24	8,23	92.31	2
คนที่ 10	12	2.03	7	98.11	5
คนที่ 11	163	55.42	158	100	4
คนที่ 12	24	3.32	23	100	7
คนที่ 13	24	6.47	22	100	7
คนที่ 14	18	7.19	16	100	5
คนที่ 15	22	1.43	21	100	7
คนที่ 16	7	2.04	7	100	1



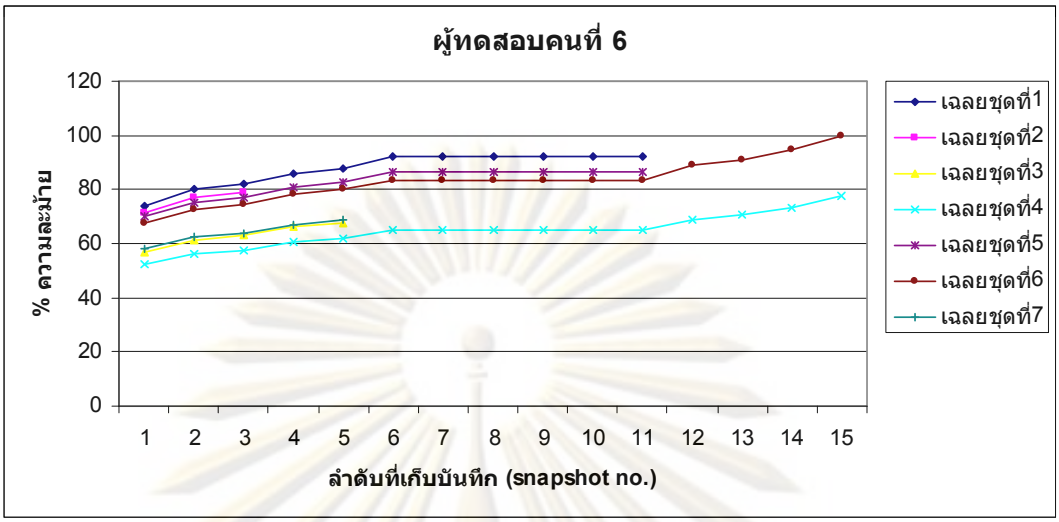
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 1 กับชุดเฉลย



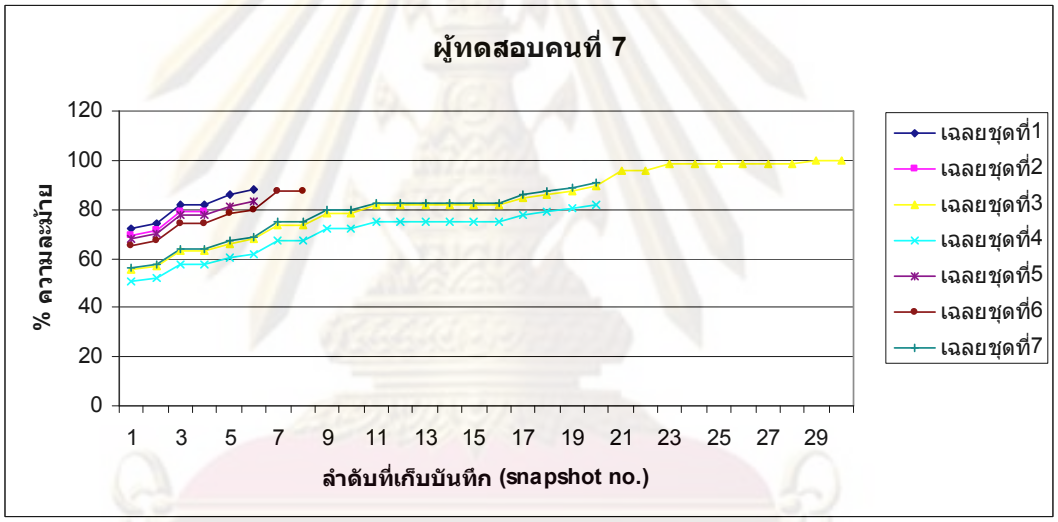
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 3 กับชุดเฉลย



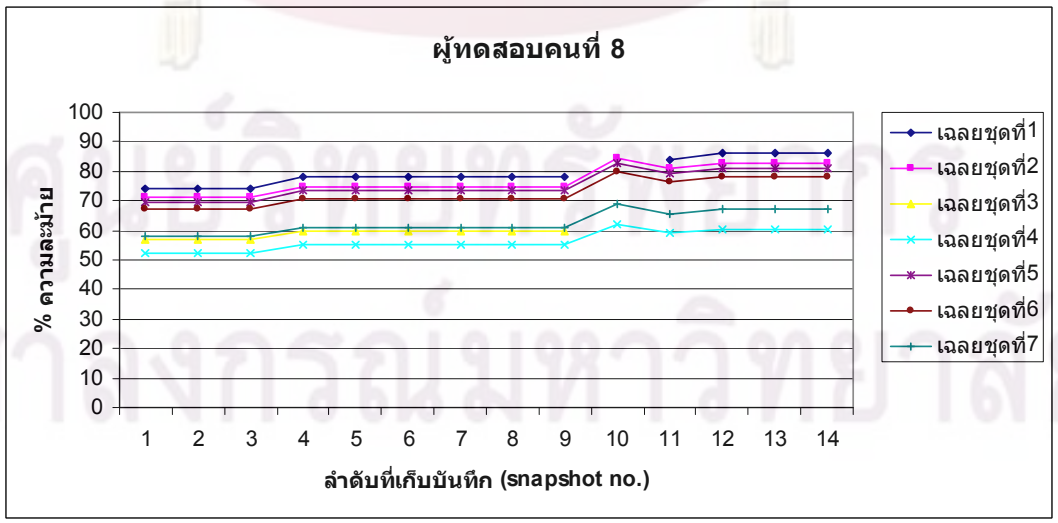
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 5 กับชุดเฉลย



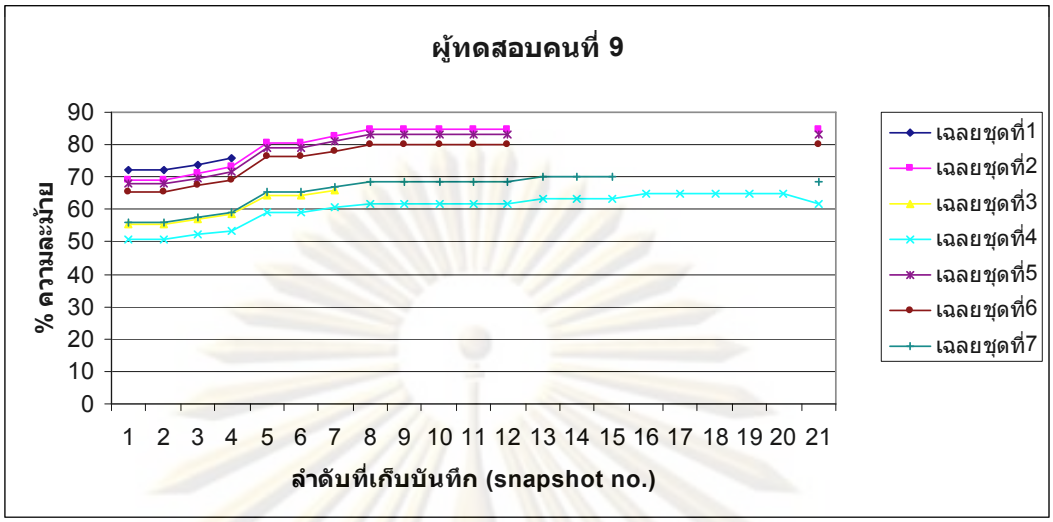
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 6 กับชุดเฉลย



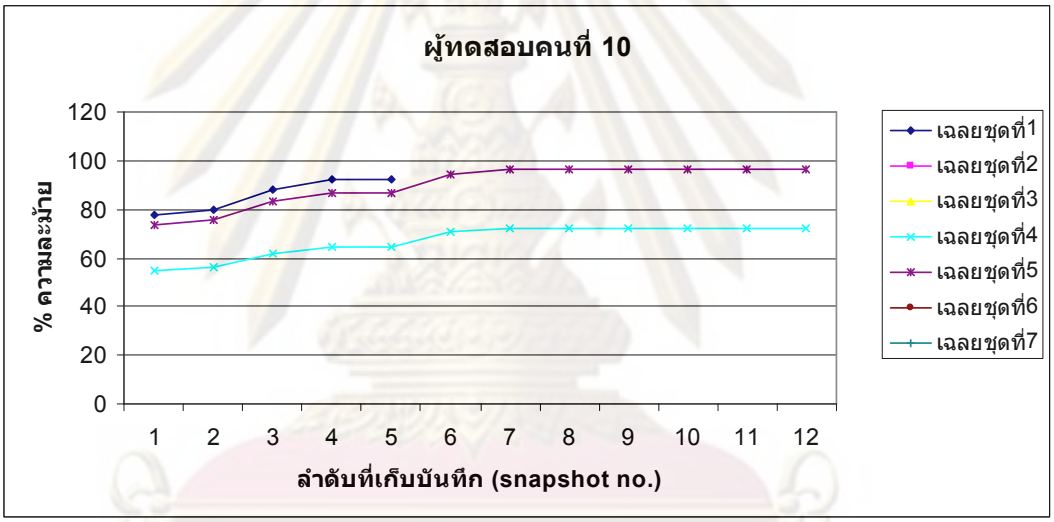
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 7 กับชุดเฉลย



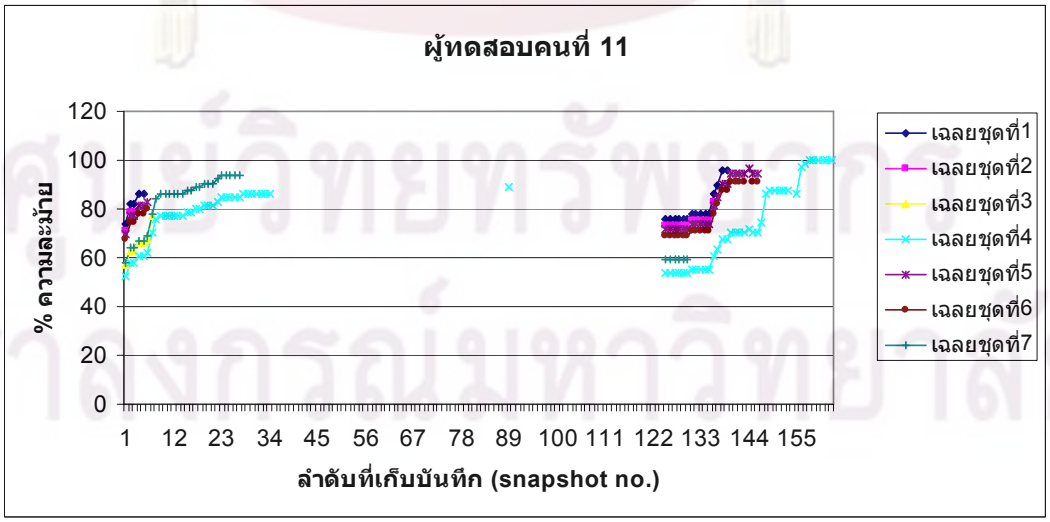
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 8 กับชุดเฉลย



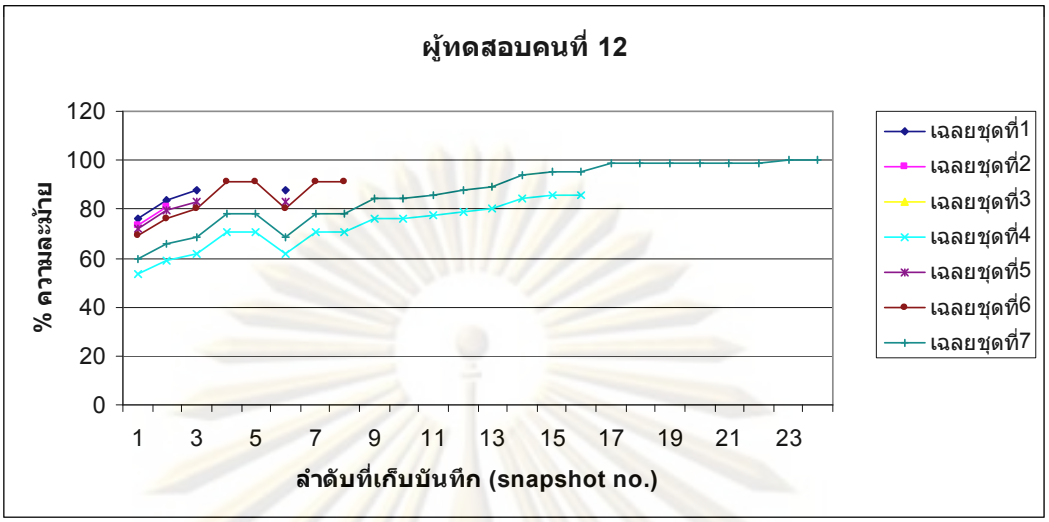
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 9 กับชุดเฉลย



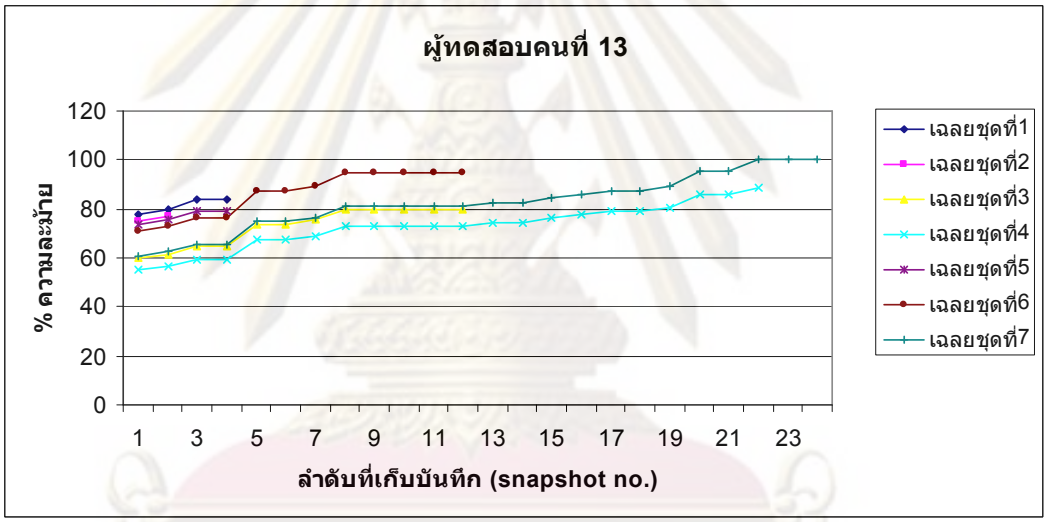
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 10 กับชุดเฉลย



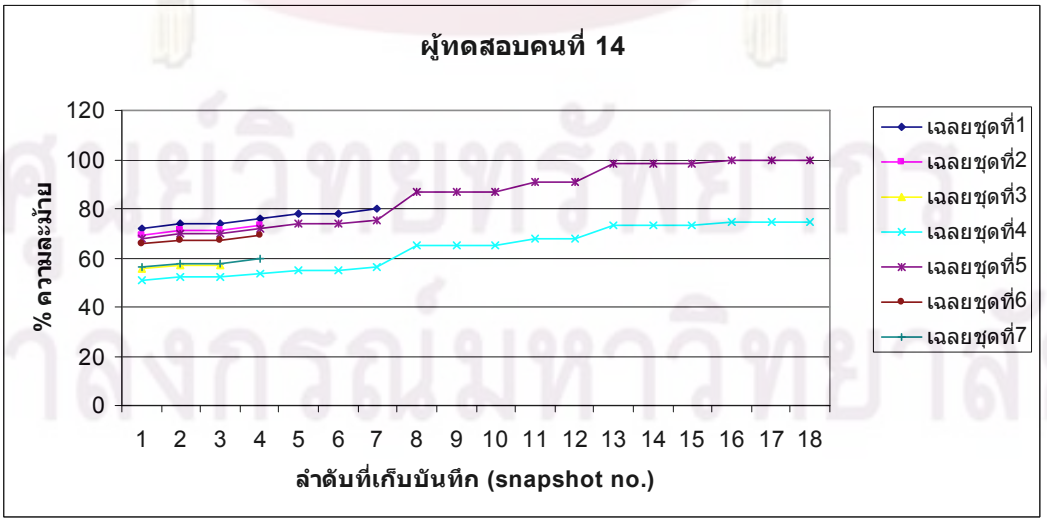
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 11 กับชุดเฉลย



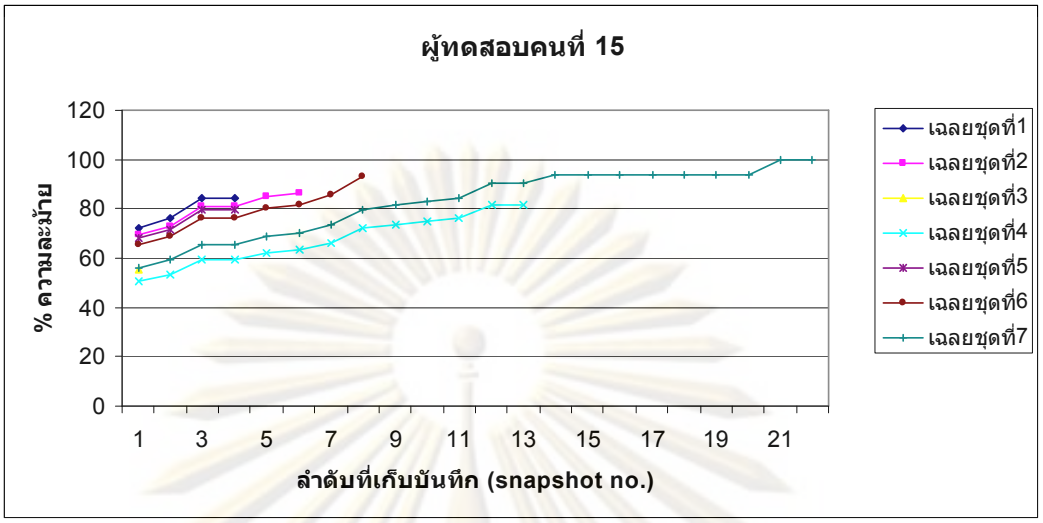
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 12 กับชุดเฉลย



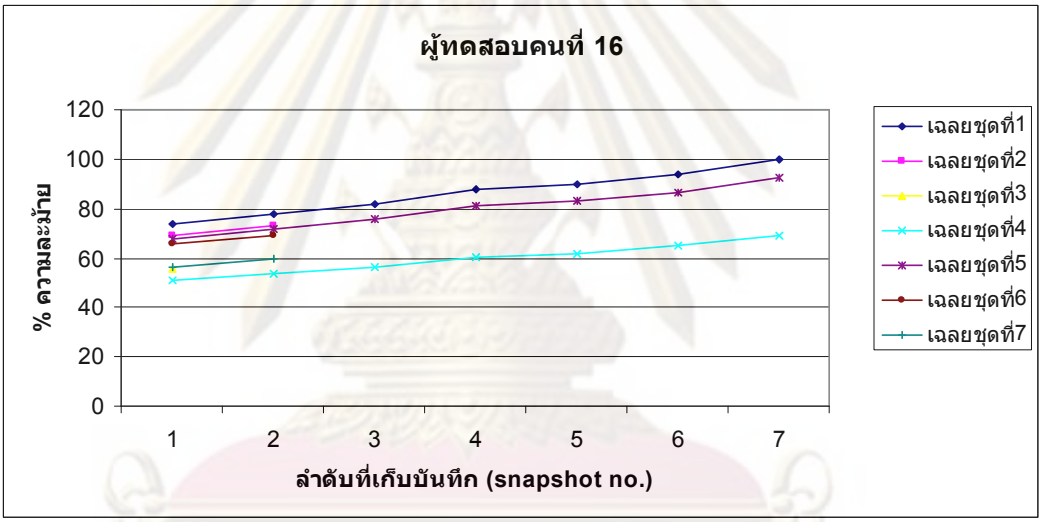
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 13 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 14 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 15 กับชุดเฉลย

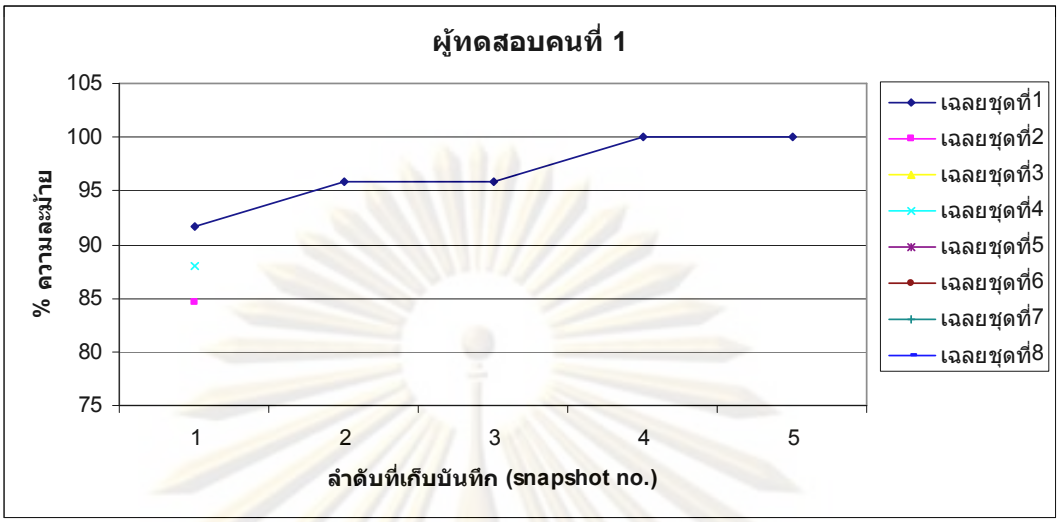


ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 3 ของผู้ทดสอบคนที่ 16 กับชุดเฉลย

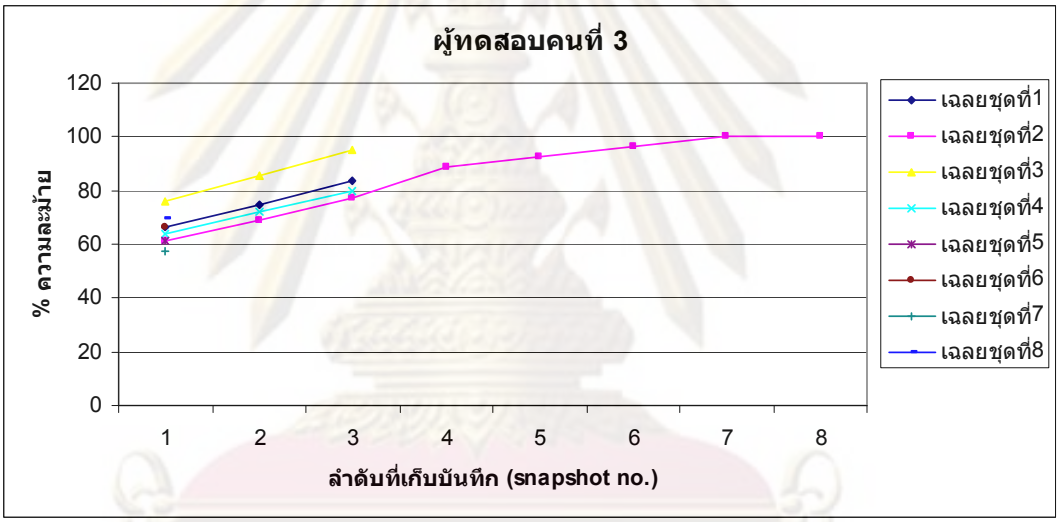
<u>แบบทดสอบที่ 4</u>

ผลการเปรียบเทียบความละม้ายของแบบทดสอบชุดที่ 4 กับของชุดเฉลย

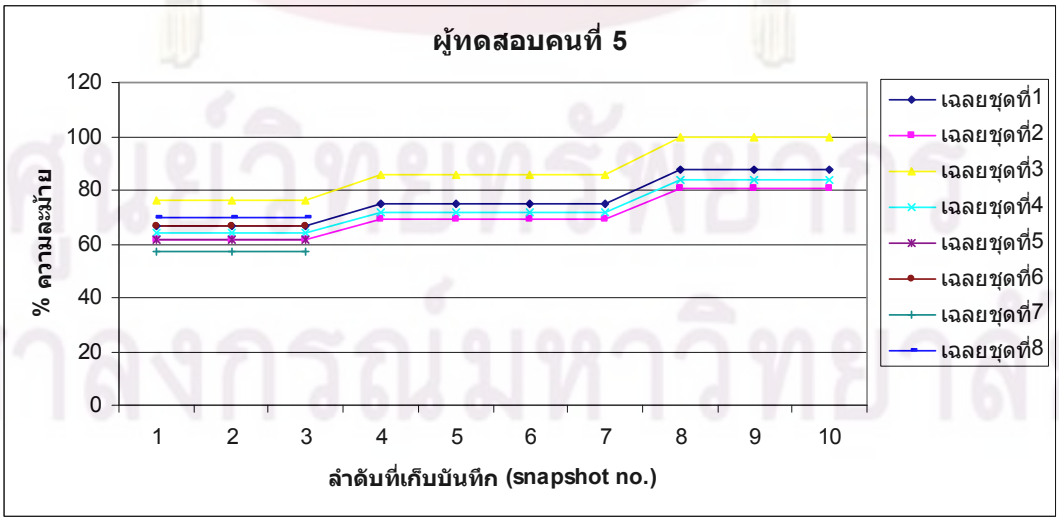
ผู้ทดสอบ	จำนวนที่บันทึกได้ (snapshots)	เวลาที่ใช้ (นาที)	ลำดับที่ละม้ายที่สุด	% ความละม้ายมากที่สุดที่ได้	ชุดเฉลยที่ละม้ายมากที่สุด
คนที่ 1	5	1.52	4	100	1
คนที่ 2	-	-	-	-	-
คนที่ 3	8	0.44	7	100	2
คนที่ 4	-	-	-	-	-
คนที่ 5	10	1.46	8	100	3
คนที่ 6	12	19.05	5,12	100	4
คนที่ 7	18	5.35	3,16	100	5
คนที่ 8	20	5.07	10	90	4
คนที่ 9	5	3.17	3	87.5	1
คนที่ 10	14	4.38	10	95.83	6
คนที่ 11	-	-	-	-	-
คนที่ 12	13	4.49	7,13	100	6
คนที่ 13	14	2.04	14	100	7
คนที่ 14	4	7.01	2	98.21	7
คนที่ 15	6	0.29	6	100	8
คนที่ 16	5	2.49	4	89.13	8



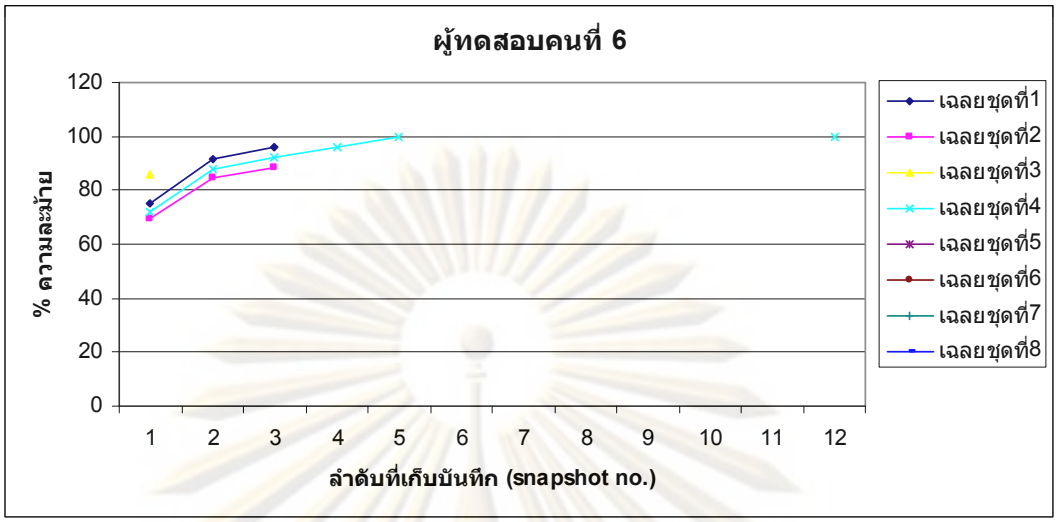
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 1 กับชุดเฉลย



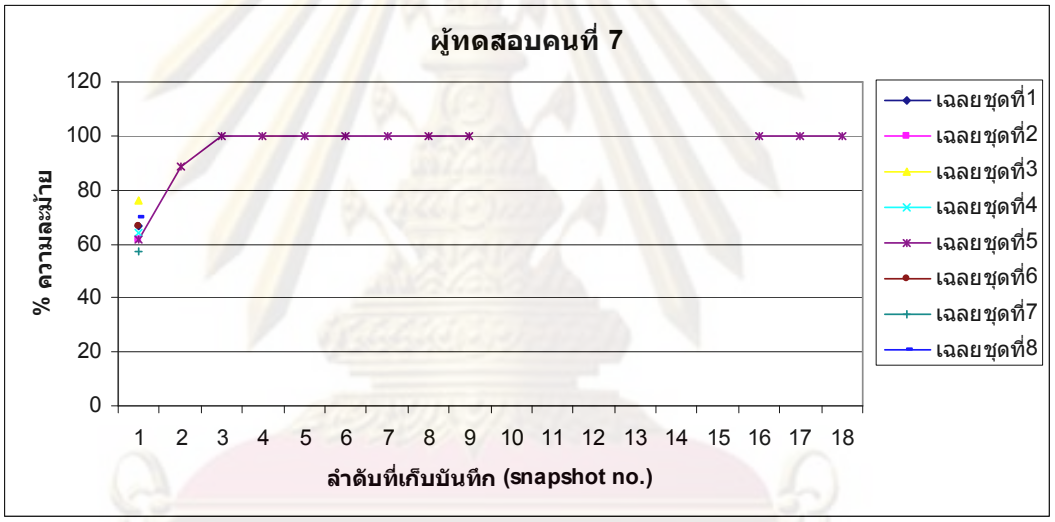
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 3 กับชุดเฉลย



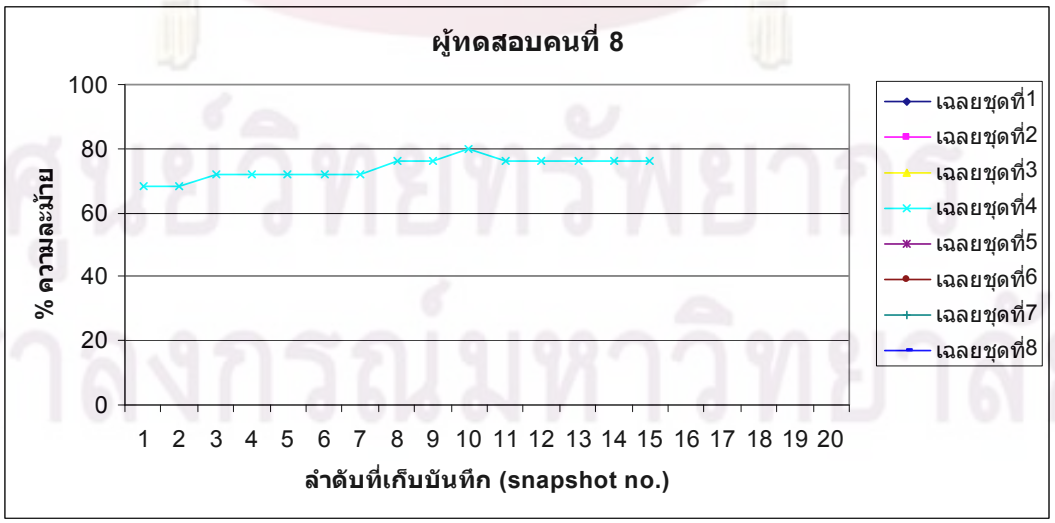
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 5 กับชุดเฉลย



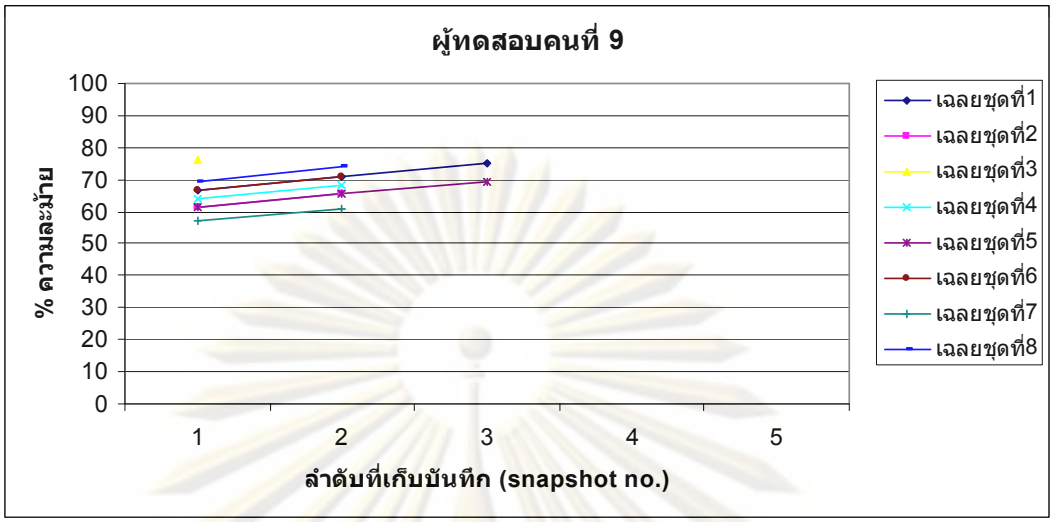
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 6 กับชุดเฉลย



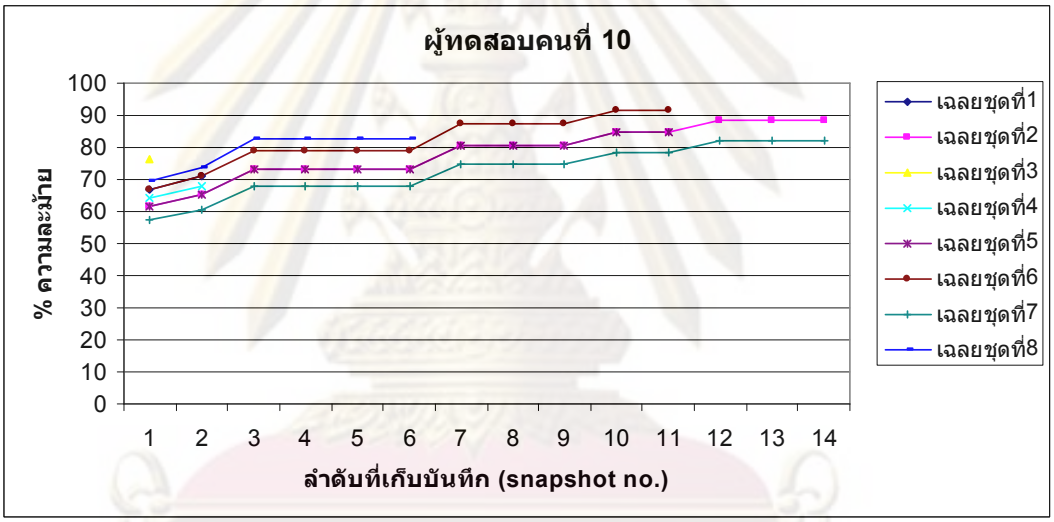
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 7 กับชุดเฉลย



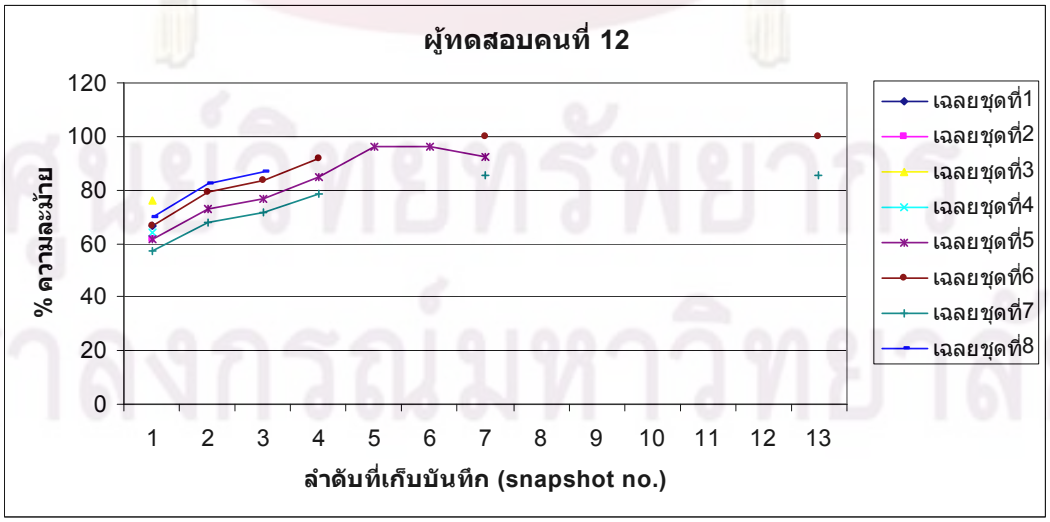
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 8 กับชุดเฉลย



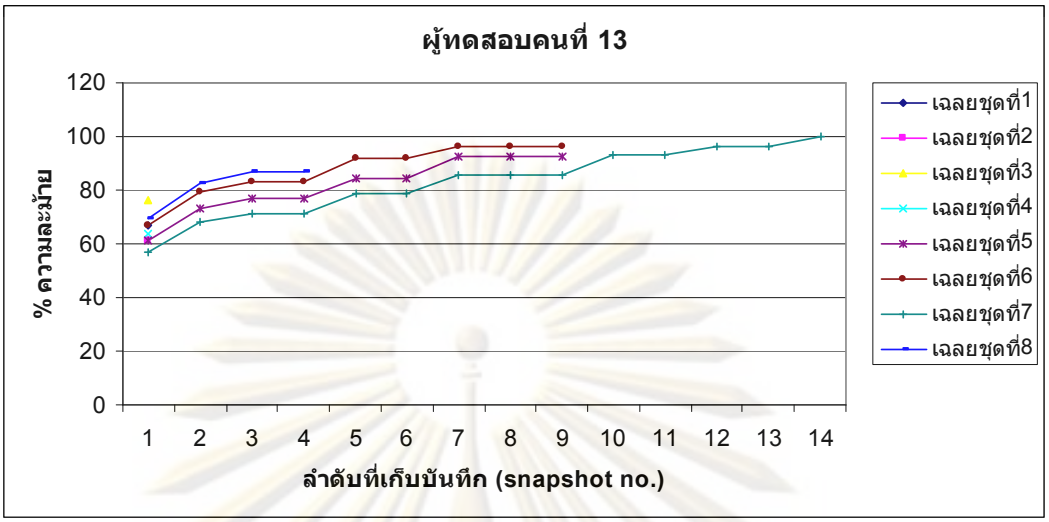
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 9 กับชุดเฉลย



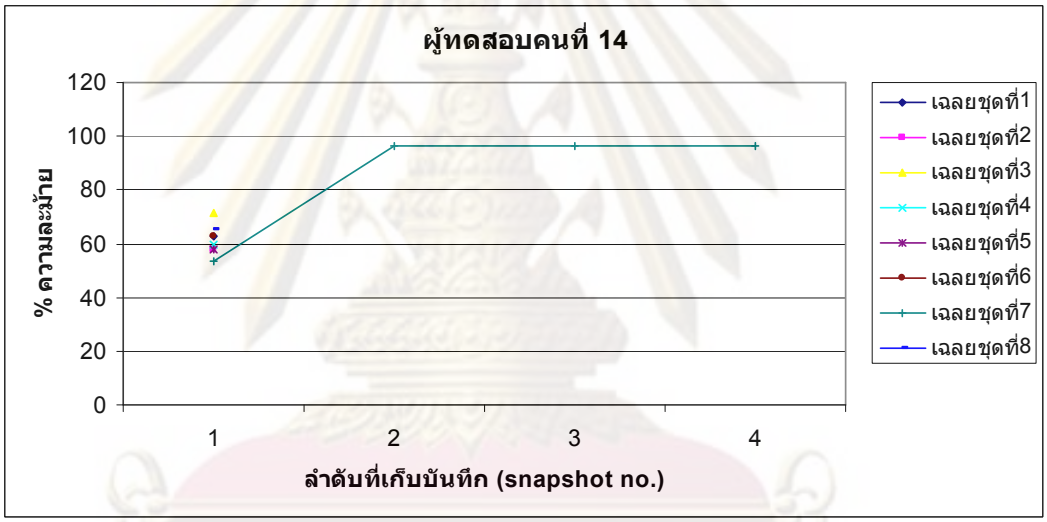
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 10 กับชุดเฉลย



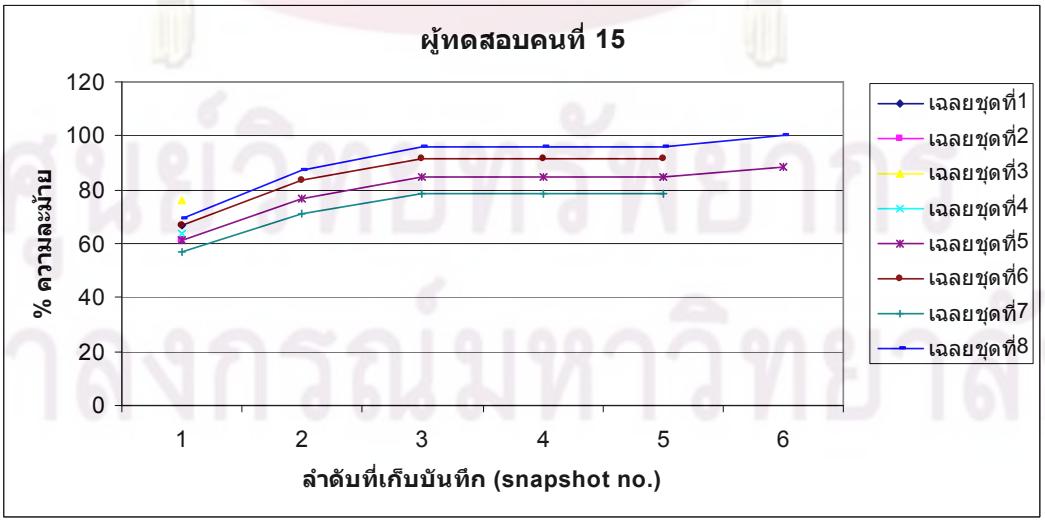
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 12 กับชุดเฉลย



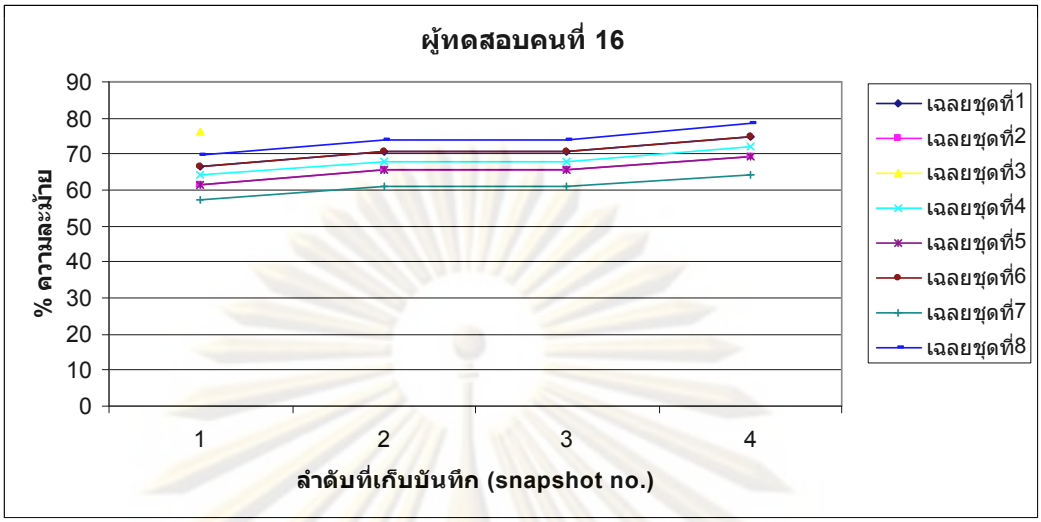
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 13 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 14 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 15 กับชุดเฉลย

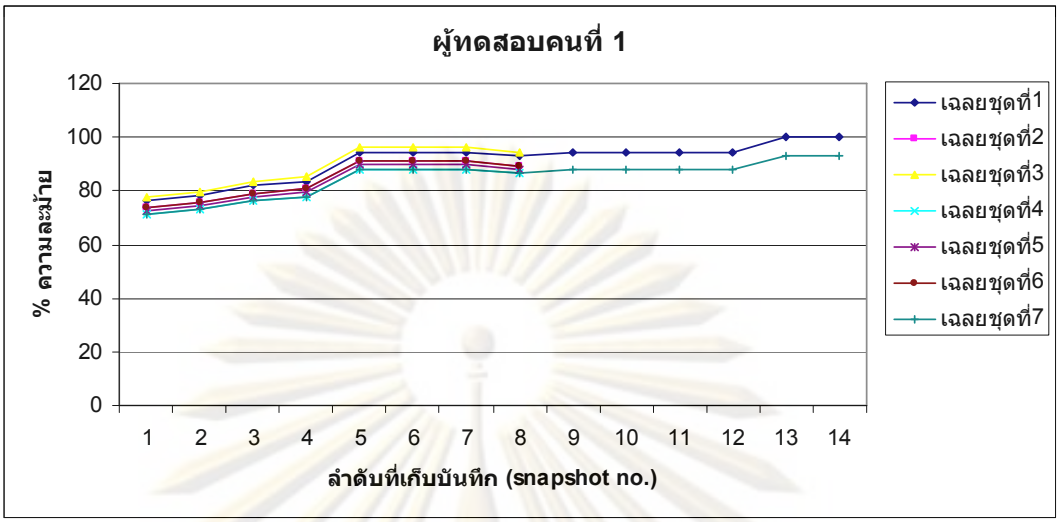


ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 4 ของผู้ทดสอบคนที่ 16 กับชุดเฉลย

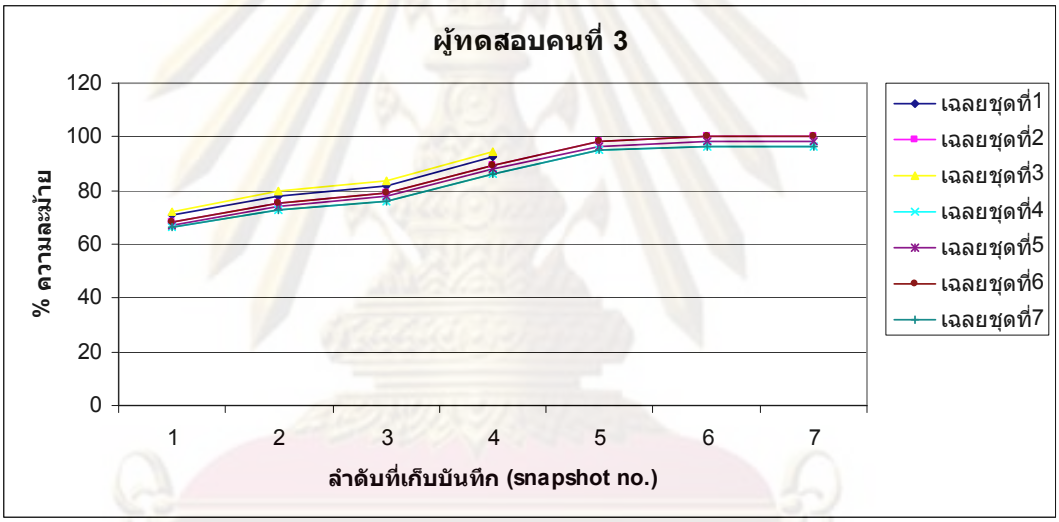
แบบทดสอบที่ 5

ผลการเปรียบเทียบความละม้ายของแบบทดสอบชุดที่ 5 กับของชุดเฉลย

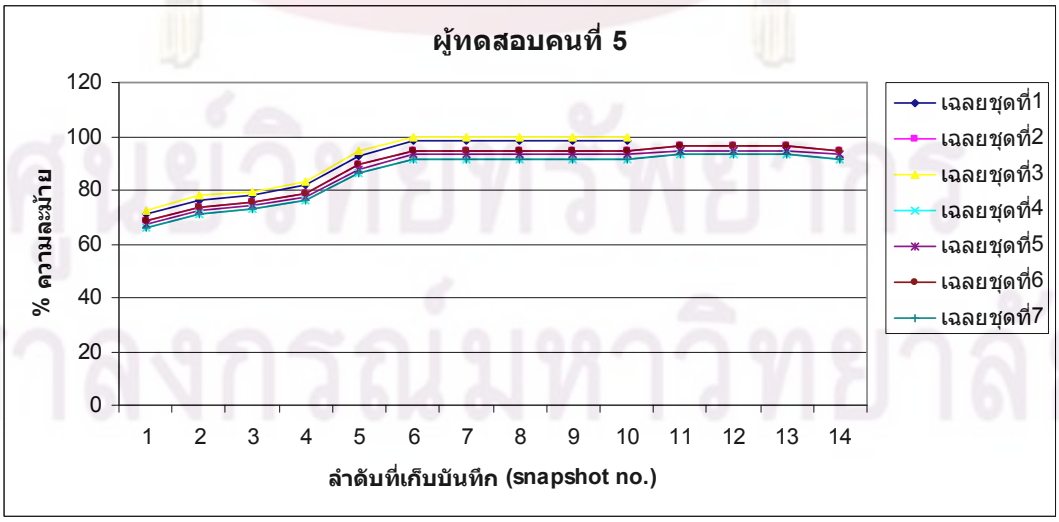
ผู้ทดสอบ	จำนวนที่บันทึกได้ (snapshots)	เวลาที่ใช้ (นาที)	ลำดับที่ละม้ายที่สุด	% ความละม้ายมากที่สุดที่ได้	ชุดเฉลยที่ละม้ายมากที่สุด
คนที่ 1	14	5.38	13	100	1
คนที่ 2	-	-	-	-	-
คนที่ 3	7	1.58	6	100	2,6
คนที่ 4	-	-	-	-	-
คนที่ 5	14	6.3	6,14	100	3
คนที่ 6	25	15.31	8,13,25	100	4
คนที่ 7	11	1.57	11	100	2,6
คนที่ 8	10	8.26	8	89.82	3
คนที่ 9	5	8.01	2	85.09	2,6
คนที่ 10	25	30.14	25	99.14	2,6
คนที่ 11	-	-	-	-	-
คนที่ 12	8	1.53	7	100	2,6
คนที่ 13	19	6.03	9	100	2,6
คนที่ 14	22	6.04	21	100	7
คนที่ 15	19	1.45	18	100	2,6
คนที่ 16	19	1.45	7	94.44	7



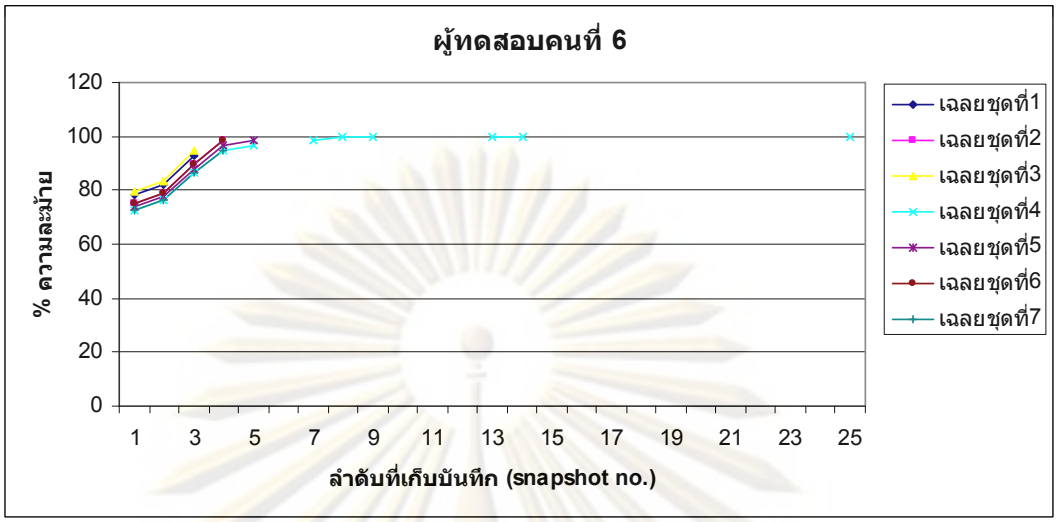
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 1 กับชุดเฉลย



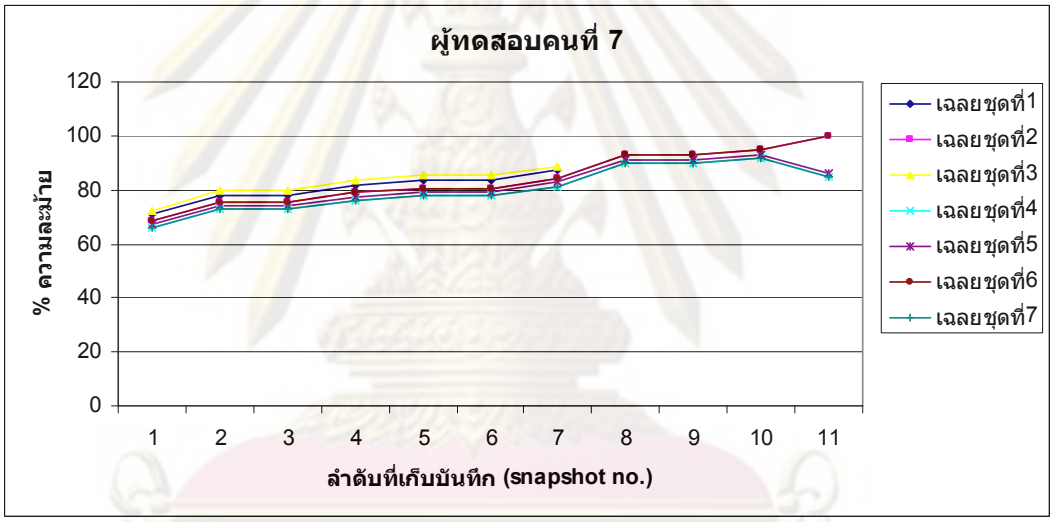
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 3 กับชุดเฉลย



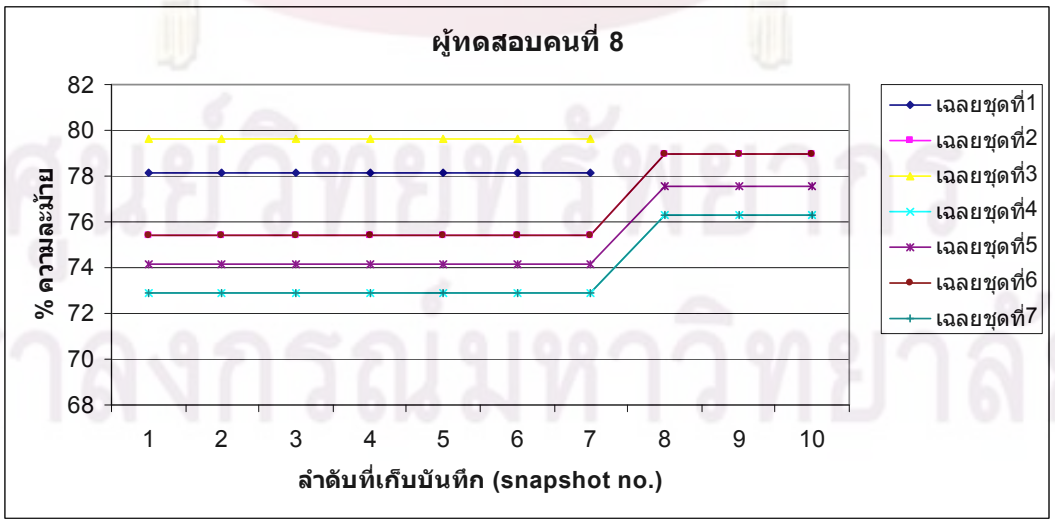
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 5 กับชุดเฉลย



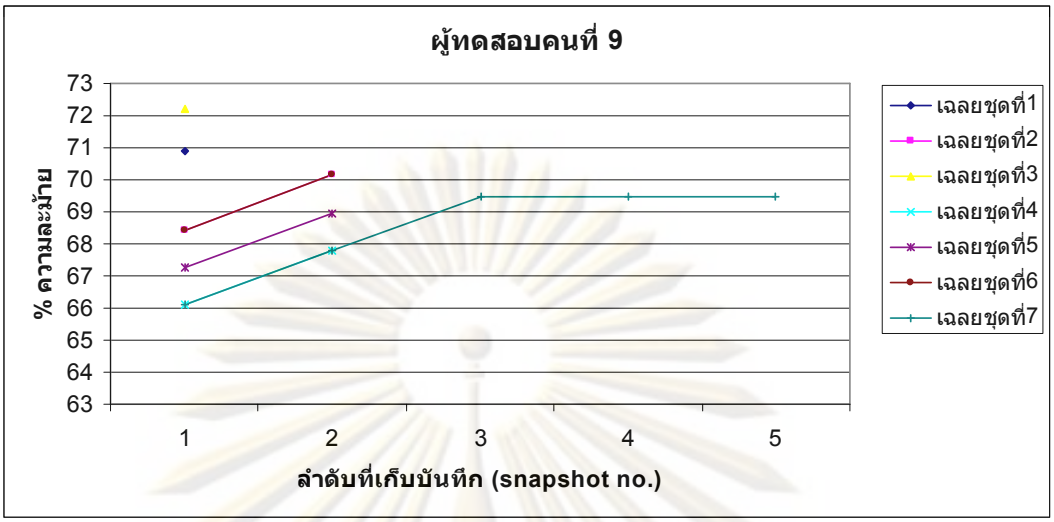
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 6 กับชุดเฉลย



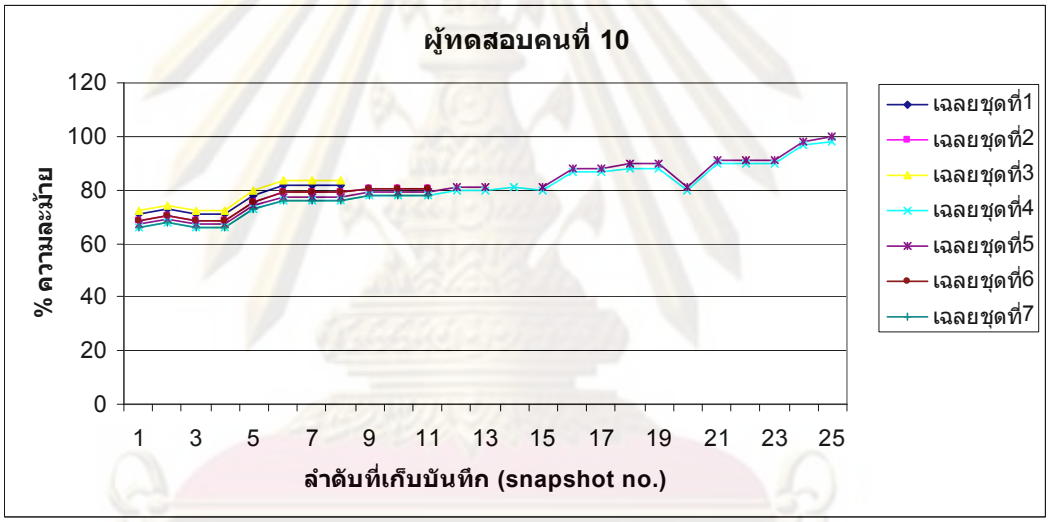
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 7 กับชุดเฉลย



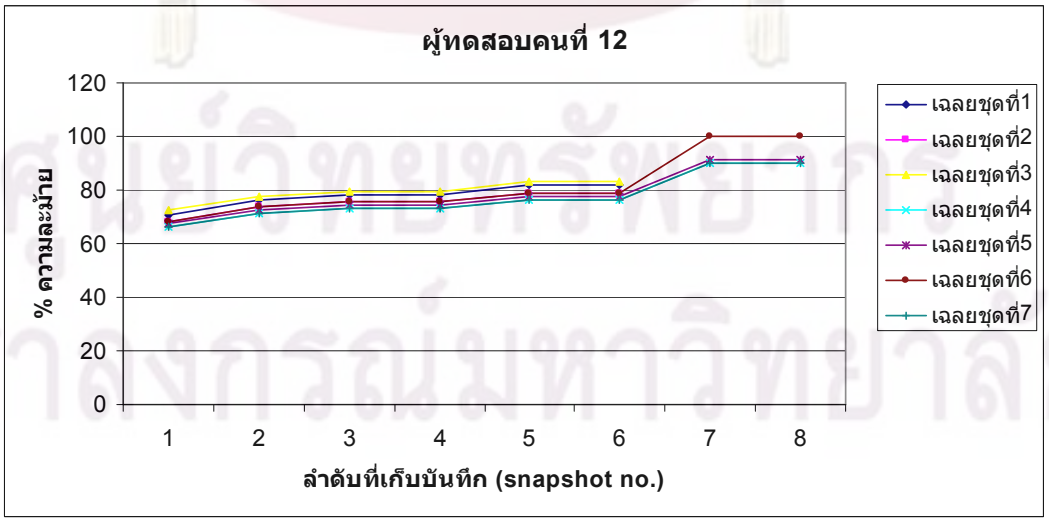
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 8 กับชุดเฉลย



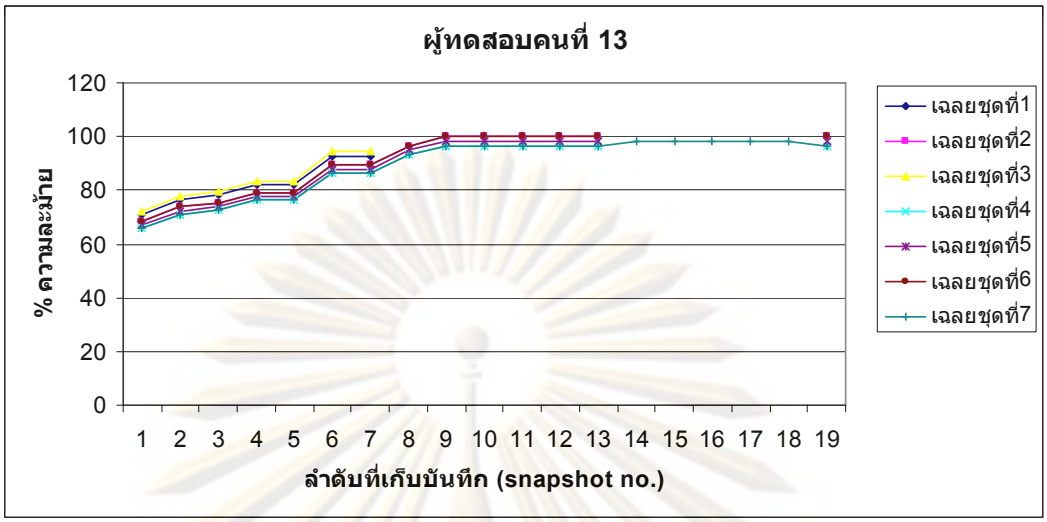
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 9 กับชุดเฉลย



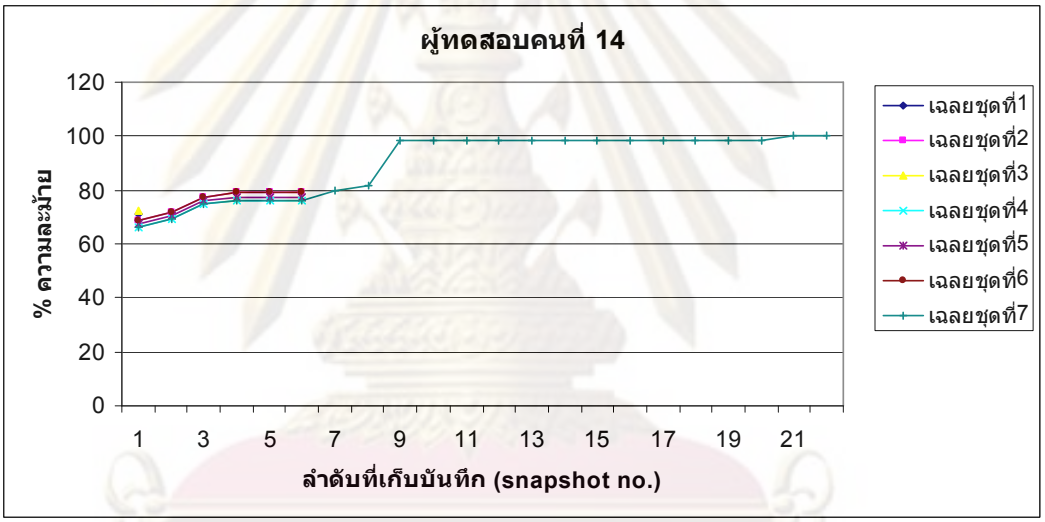
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 10 กับชุดเฉลย



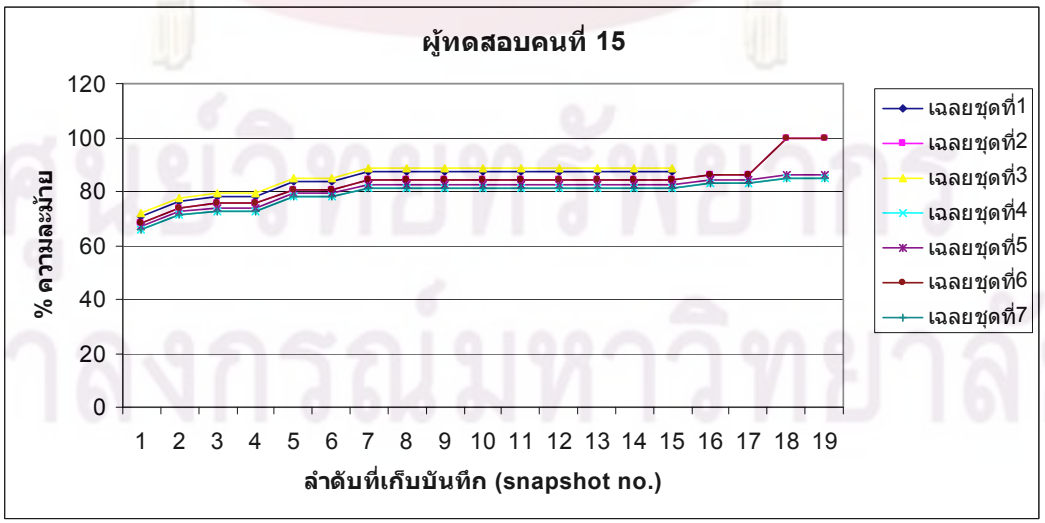
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 12 กับชุดเฉลย



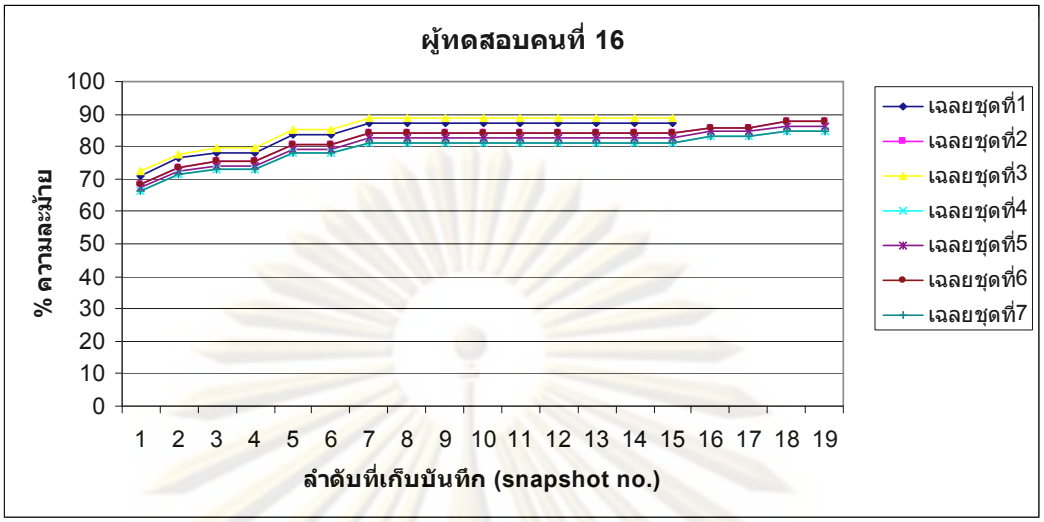
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 13 กับชุดเฉลย



ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 14 กับชุดเฉลย

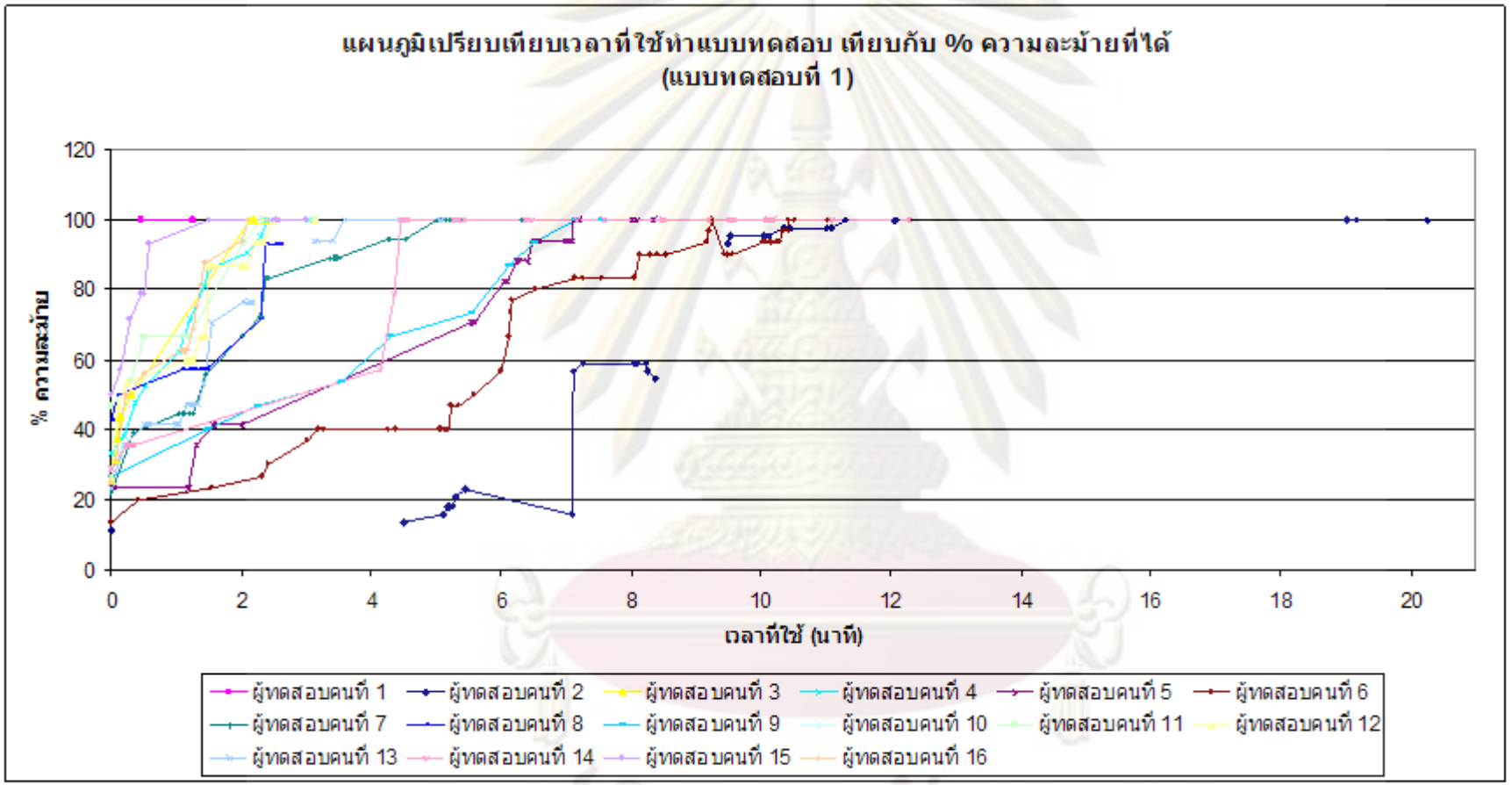


ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 15 กับชุดเฉลย



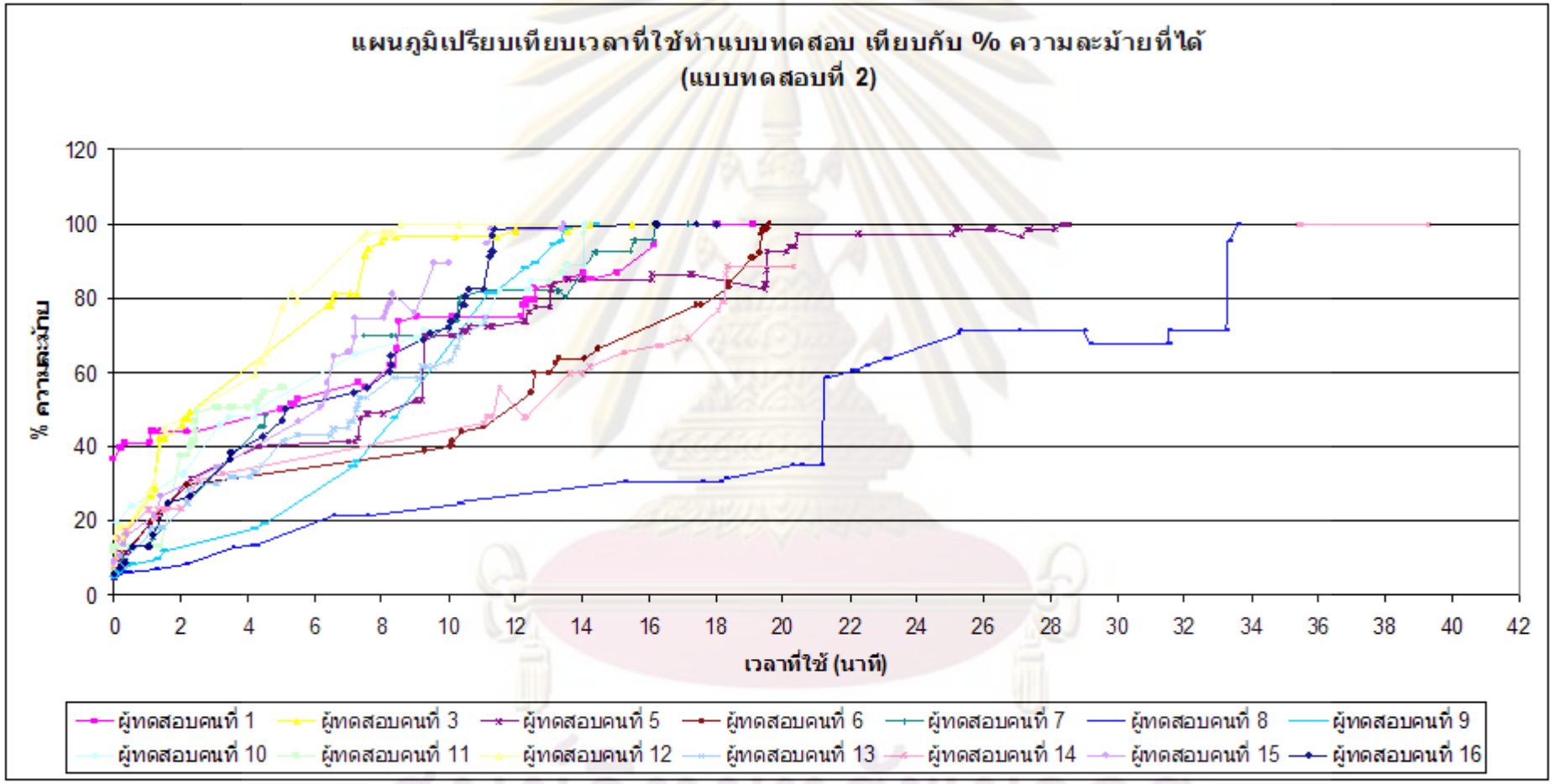
ผลการเปรียบเทียบความละม้ายของแบบทดสอบที่ 5 ของผู้ทดสอบคนที่ 16 กับชุดเฉลย

ภาคผนวก จ

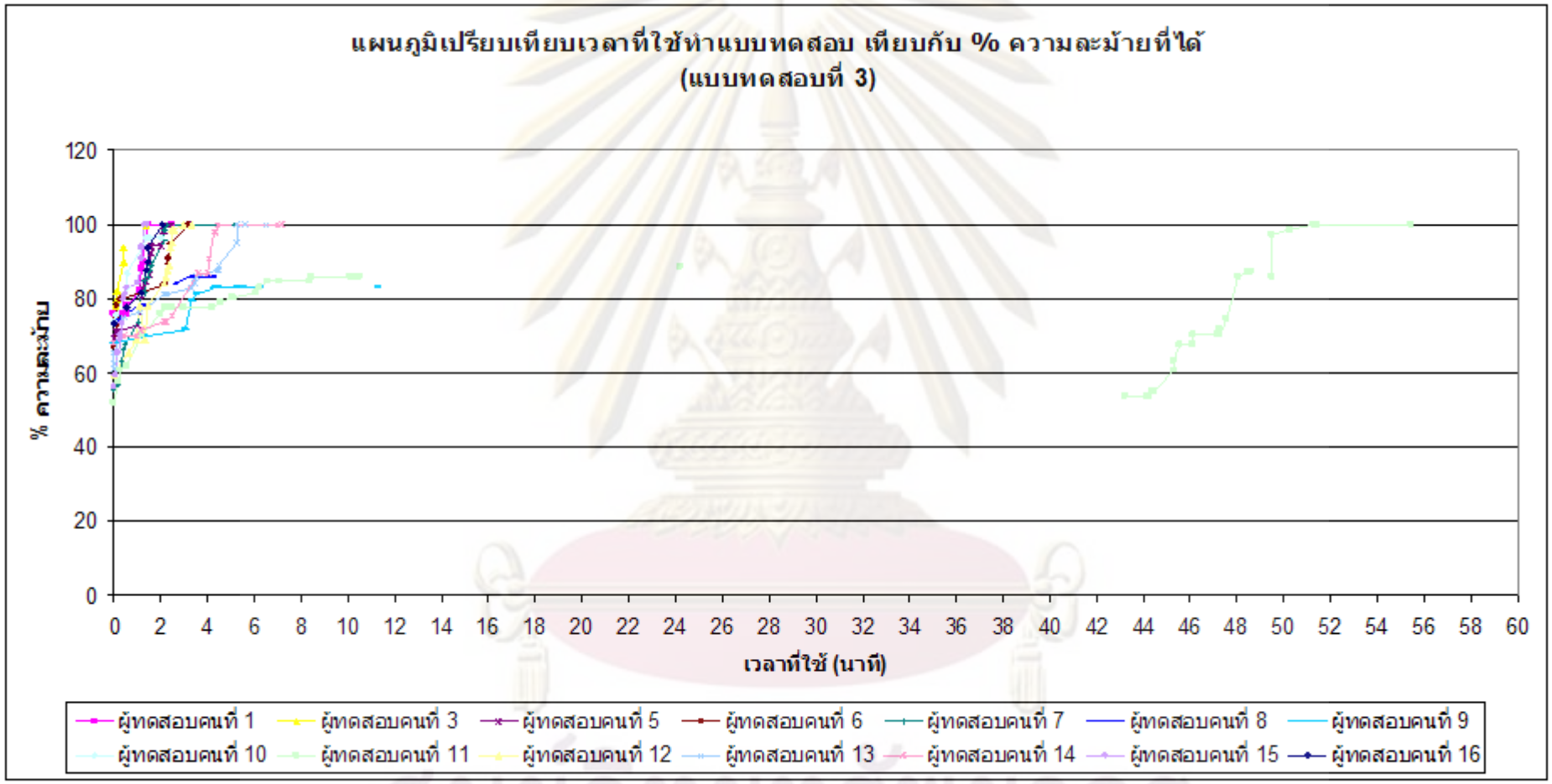


แผนภูมิเปรียบเทียบเวลาที่ผู้ทดสอบใช้ทำแบบทดสอบ เทียบกับค่า % ความละม้ายที่ได้ (แบบทดสอบที่ 1)

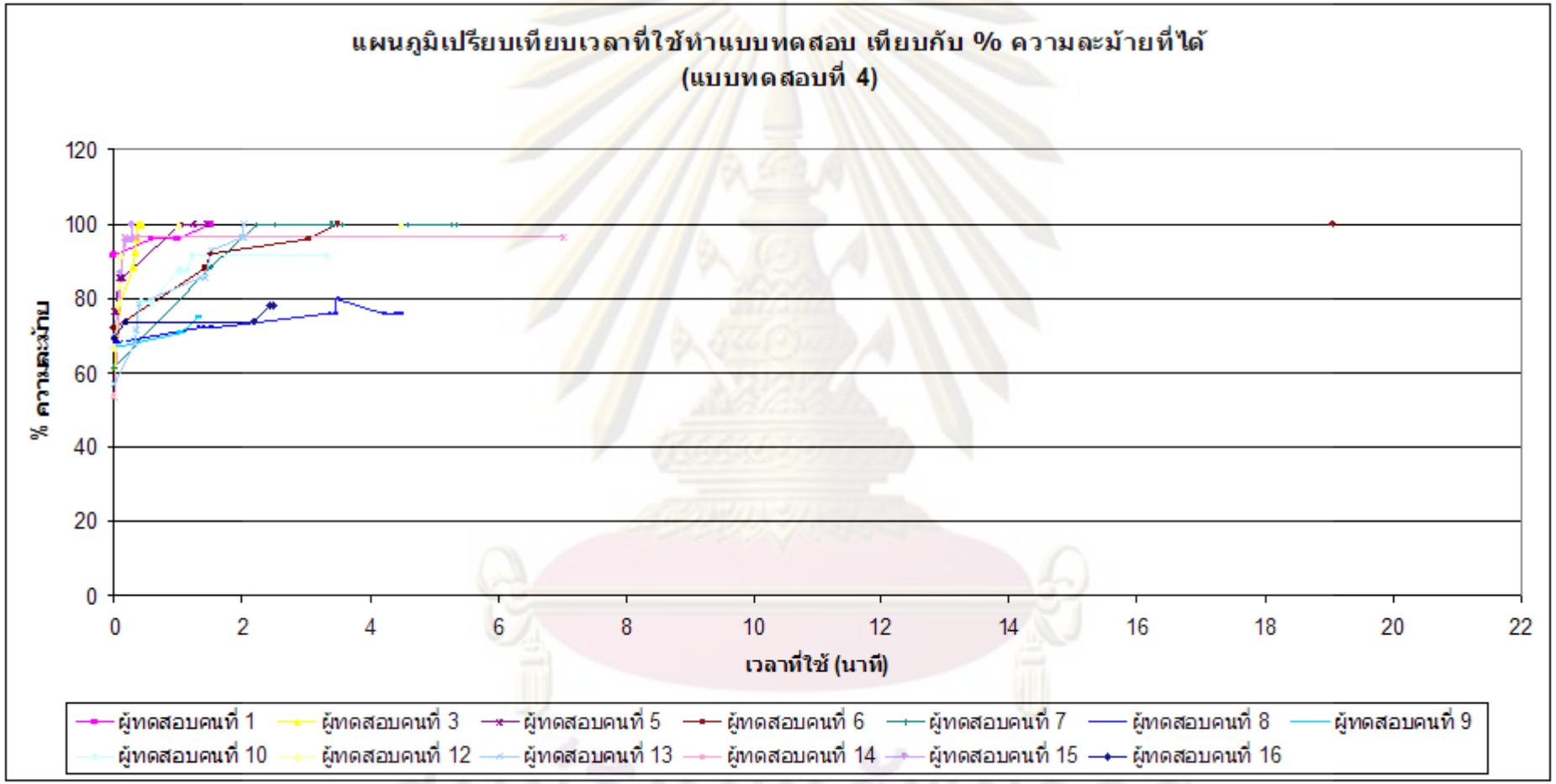
แผนภูมิเปรียบเทียบเวลาที่ผู้ทดสอบใช้ทำแบบทดสอบ เทียบกับค่า % ความละม้ายที่ได้ (แบบทดสอบที่ 1)



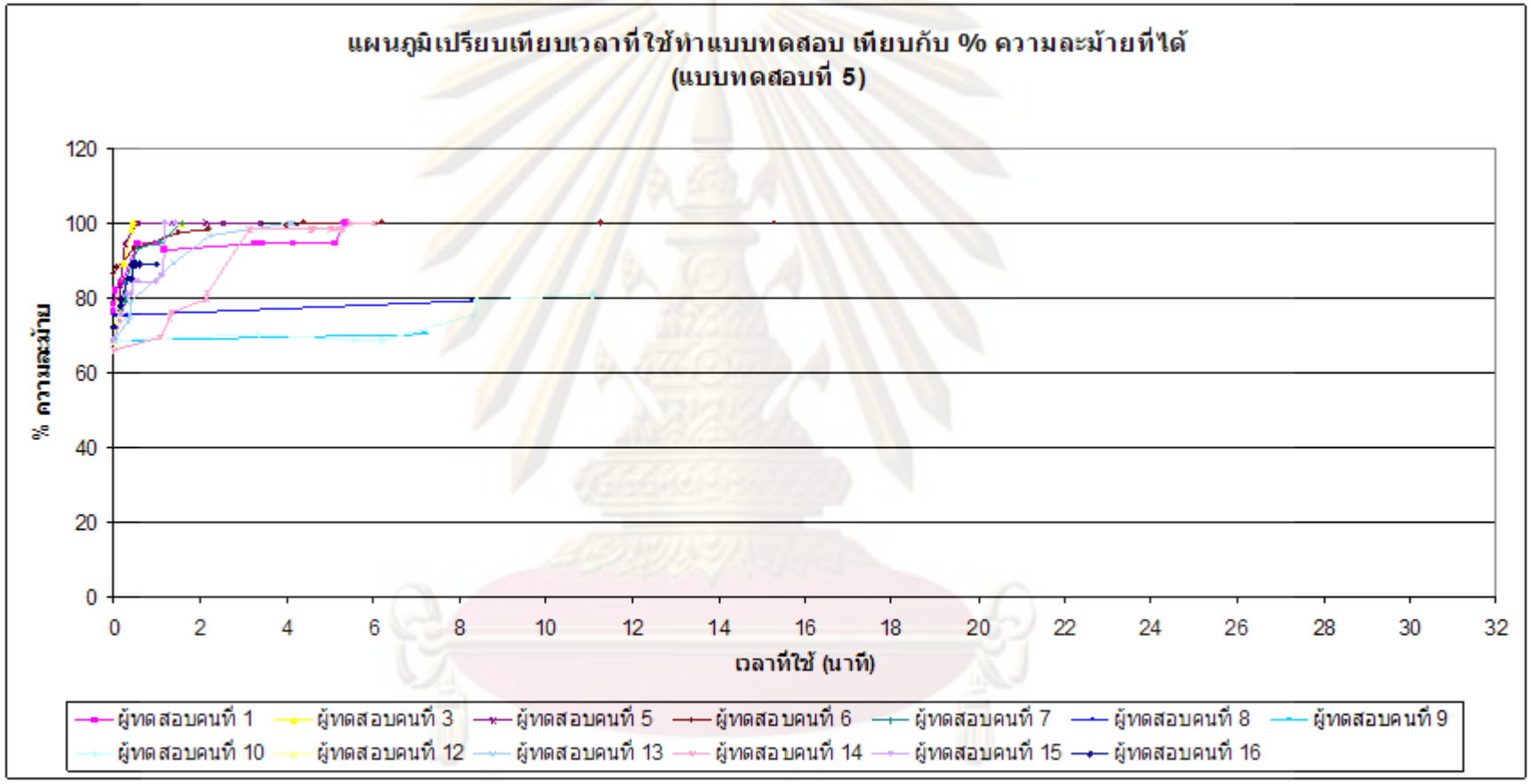
แผนภูมิเปรียบเทียบเวลาที่ผู้ทดสอบใช้ทำแบบทดสอบ เทียบกับค่า % ความละม้ายที่ได้ (แบบทดสอบที่ 2)



แผนภูมิเปรียบเทียบเวลาที่ผู้ทดสอบใช้ทำแบบทดสอบ เทียบกับค่า % ความละม้ายที่ได้ (แบบทดสอบที่ 3)



แผนภูมิเปรียบเทียบเวลาที่ผู้ทดสอบใช้ทำแบบทดสอบ เทียบกับค่า % ความละม้ายที่ได้ (แบบทดสอบที่ 4)



แผนภูมิเปรียบเทียบเวลาที่ผู้ทดสอบใช้ทำแบบทดสอบ เทียบกับค่า % ความละม้ายที่ได้ (แบบทดสอบที่ 5)

ภาคผนวก ฉ

การทดสอบการคำนวนค่าความพยายามตามมาตรวัดของฮอลสตีด

การทดสอบการคำนวนค่าความพยายามตามมาตรวัดของฮอลสตีด ทำการทดสอบโดยใช้วิธีการตัดชุดคำสั่งออกจากรหัสต้นฉบับของโปรแกรมที่สมบูรณ์ ครั้งละคำสั่งหรือหลายคำสั่งแบบสลับตำแหน่ง และนำมาคำนวนค่าความละม้ายและค่าความพยายามตามมาตรวัดของฮอลสตีด และศึกษาเปรียบเทียบผลที่วัดได้ และวาดเป็นกราฟเพื่อศึกษาลักษณะของการเปลี่ยนแปลงของผลคะแนนที่ได้

ผลที่ได้แสดงให้เห็นว่าการใช้ค่าความพยายามตามมาตรวัดของฮอลสตีดให้ผลของค่าเปอร์เซ็นต์ที่จะนำไปใช้คำนวนให้คะแนนที่เหมาะสมกว่าการใช้ค่าเปอร์เซ็นต์ความละม้ายที่คำนวนได้มาช่วยในการให้คะแนน ซึ่งการคำนวนหาค่าความละม้ายเป็นการเปรียบเทียบโทเค็นระหว่างรหัสต้นฉบับของโปรแกรม ซึ่งไม่ได้ให้น้ำหนักระหว่างชุดคำสั่งประเภทที่ต่างกัน ทำให้ผลการเปลี่ยนแปลงของค่าเปอร์เซ็นต์ที่ได้ไม่ค่อยเปลี่ยนแปลงมากนักเมื่อมีการตัดชุดคำสั่งออกสลับกัน ซึ่งแตกต่างจากการคำนวนค่าความพยายามตามมาตรวัดของฮอลสตีดซึ่งสะท้อนให้เห็นถึงน้ำหนักที่แตกต่างกันระหว่างชุดคำสั่งแต่ละชุดที่ตัดออก ซึ่งชุดคำสั่งที่ซับซ้อนมากกว่าควรจะมีผลต่อการให้คะแนนที่มากกว่าชุดคำสั่งที่ซับซ้อนน้อยกว่า

รายละเอียดของผลการทดสอบแบ่งเป็นดังนี้

- การทดสอบโปรแกรม HelloWorld

การทดสอบโปรแกรม HelloWorld เป็นการทดสอบกับรหัสต้นฉบับของโปรแกรมที่ชุดคำสั่งที่ใช้แต่ละชุดมีความซับซ้อนของแต่ละคำสั่งไม่แตกต่างกัน

- การทดสอบโปรแกรม Area

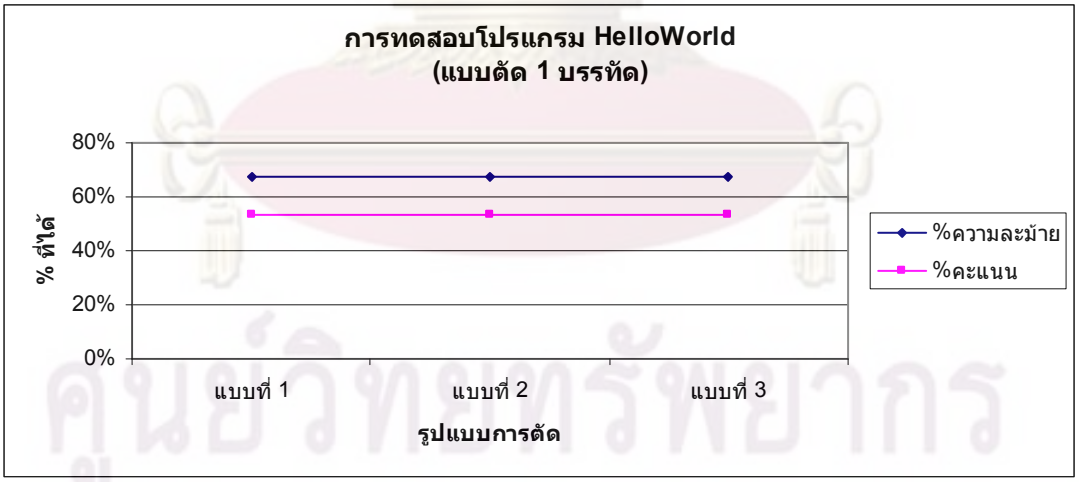
การทดสอบโปรแกรม Area เป็นการทดสอบกับรหัสต้นฉบับของโปรแกรมที่ชุดคำสั่งที่ใช้แต่ละชุดมีความซับซ้อนของแต่ละคำสั่งแตกต่างกัน

การทดสอบโปรแกรม HelloWorld

- การทดสอบแบบตัดชุดคำสั่ง 1 บรรทัด

	แบบที่ 1		แบบที่ 2		แบบที่ 3	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	67	53	67	53	67	53
public class HelloWorld {						
public static void main(String[] args) {						
System.out.print("Hello ");						
System.out.print("World ");						
System.out.print("A");						
}						
}						

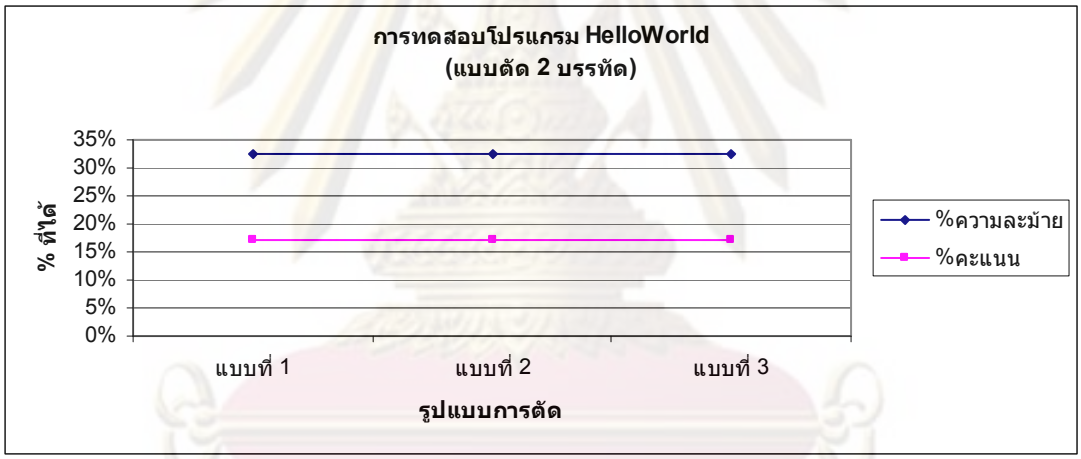
หมายเหตุ : พื้นที่สีดำแสดงถึงการตัดชุดคำสั่งของบรรทัดนั้นออก เช่นในแบบที่ 1 จะตัดบรรทัด System.out.print(“Hello “); ออก และนำไปคำนวนค่าเปอร์เซ็นต์ความละม้ายและค่าเปอร์เซ็นต์ความพยายามตามมาตรวัดของฮอลสตีด โดยเทียบกับรหัสต้นฉบับของโปรแกรม HelloWorld ที่เสร็จสมบูรณ์ ซึ่งคือรหัสต้นฉบับที่ไม่ถูกตัดชุดคำสั่งใด ๆ ออก



แผนภูมิแสดงการทดสอบโปรแกรม HelloWorld แบบตัดชุดคำสั่ง 1 บรรทัด

- การทดสอบแบบตัดชุดคำสั่ง 2 บรรทัด

	แบบที่ 1		แบบที่ 2		แบบที่ 3	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
public class HelloWorld {	33	17	33	17	33	17
public static void main(String[] args) {						
System.out.print("Hello ");						
System.out.print("World ");						
System.out.print("A");						
}						
}						

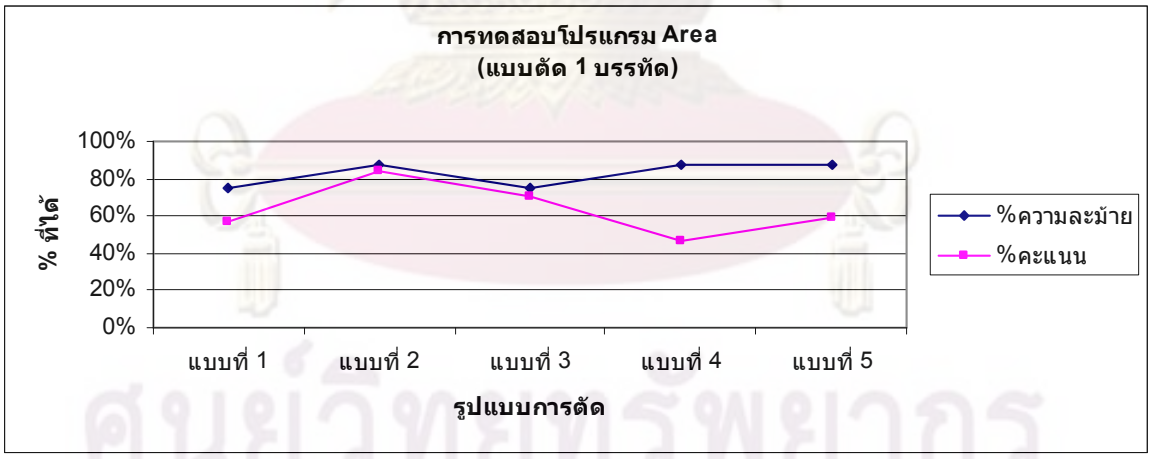


แผนภูมิแสดงการทดสอบโปรแกรม HelloWorld แบบตัดชุดคำสั่ง 2 บรรทัด

การทดสอบโปรแกรม Area

- การทดสอบแบบตัดชุดคำสั่ง 1 บรรทัด

	แบบที่ 1		แบบที่ 2		แบบที่ 3		แบบที่ 4		แบบที่ 5	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	75	57	87	84	75	70	87	47	87	59
import java.util.Scanner;										
public class Area {										
public static void main(String[] args) {										
Scanner sc = new Scanner(System.in);										
System.out.println("รัศมี = ")										
double r = sc.nextDouble();										
double area = Math.PI * r * r;										
System.out.println("พื้นที่ = "+area)										
}										
}										

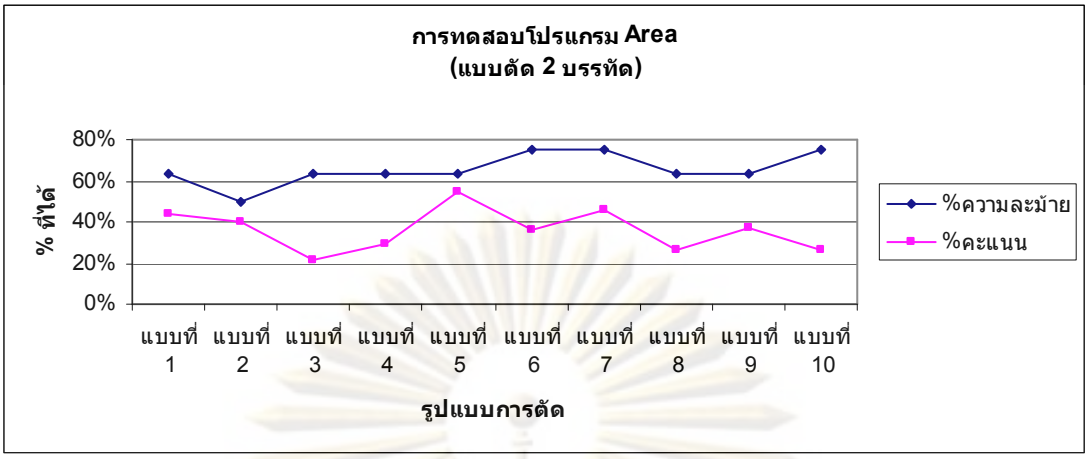


แผนภูมิแสดงการทดสอบโปรแกรม Area แบบตัดชุดคำสั่ง 1 บรรทัด

- การทดสอบแบบตัดชุดคำสั่ง 2 บรรทัด

	แบบที่ 1		แบบที่ 2		แบบที่ 3		แบบที่ 4		แบบที่ 5	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	63	44	49	40	63	21	63	29	63	55
import java.util.Scanner;										
public class Area {										
public static void main(String[] args) {										
Scanner sc = new Scanner(System.in);										
System.out.println("รัศมี = ")										
double r = sc.nextDouble();										
double area = Math.PI * r * r;										
System.out.println("พื้นที่ = "+area)										
}										
}										

	แบบที่ 6		แบบที่ 7		แบบที่ 8		แบบที่ 9		แบบที่ 10	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	75	36	75	46	63	26	63	37	75	26
import java.util.Scanner;										
public class Area {										
public static void main(String[] args) {										
Scanner sc = new Scanner(System.in);										
System.out.println("รัศมี = ")										
double r = sc.nextDouble();										
double area = Math.PI * r * r;										
System.out.println("พื้นที่ = "+area)										
}										
}										

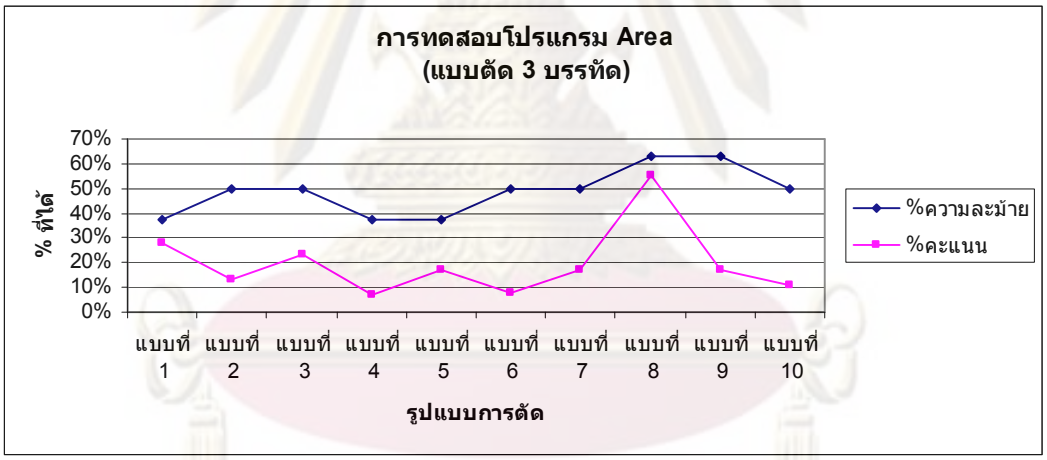


แผนภูมิแสดงการทดสอบโปรแกรม Area แบบตัดชุดคำสั่ง 2 บรรทัด

- การทดสอบแบบตัดชุดคำสั่ง 3 บรรทัด

	แบบที่ 1		แบบที่ 2		แบบที่ 3		แบบที่ 4		แบบที่ 5	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	37	28	49	13	49	23	37	7	37	17
import java.util.Scanner;										
public class Area {										
public static void main(String[] args) {										
Scanner sc = new Scanner(System.in);										
System.out.println("รัศมี = ")										
double r = sc.nextDouble();										
double area = Math.PI * r * r;										
System.out.println("พื้นที่ = "+area)										
}										
}										

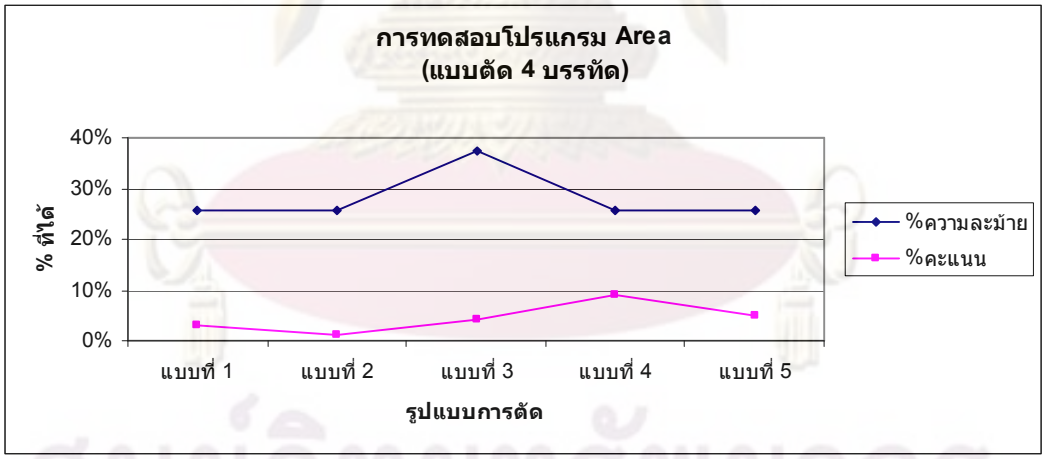
	แบบที่ 6		แบบที่ 7		แบบที่ 8		แบบที่ 9		แบบที่ 10	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	49	8	49	17	63	55	63	17	49	11
import java.util.Scanner;										
public class Area {										
public static void main(String[] args) {										
Scanner sc = new Scanner(System.in);										
System.out.println("รัศมี = ")										
double r = sc.nextDouble();										
double area = Math.PI * r * r;										
System.out.println("พื้นที่ = "+area)										
}										
}										



แผนภูมิแสดงการทดสอบโปรแกรม Area แบบตัดชุดคำสั่ง 3 บรรทัด

- การทดสอบแบบตัดชุดคำสั่ง 4 บรรทัด

	แบบที่ 1		แบบที่ 2		แบบที่ 3		แบบที่ 4		แบบที่ 5	
	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม	% ความละม้าย	% ความพยายาม
	25	3	25	1	37	4	25	9	25	5
import java.util.Scanner;										
public class Area {										
public static void main(String[] args) {										
Scanner sc = new Scanner(System.in);										
System.out.println("รัศมี = ")										
double r = sc.nextDouble();										
double area = Math.PI * r * r;										
System.out.println("พื้นที่ = "+area)										
}										
}										



แผนภูมิแสดงการทดสอบโปรแกรม Area แบบตัดชุดคำสั่ง 4 บรรทัด

ประวัติผู้เขียนวิทยานิพนธ์

นายศิวนันทน์ บุญประเสริฐ สำเร็จการศึกษาระดับมัธยมศึกษาจาก โรงเรียนสาธิตมหาวิทยาลัยศรีนครินทรวิโรฒ ปทุมวัน และสำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต ภาควิชาวิศวกรรมเคมี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2544 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีพุทธศักราช 2549 และปัจจุบันทำงานอยู่ที่บริษัทชอปปิ้งเว็บ ไทยแลนด์ (จำกัด) โดยดูแลในส่วนของการพัฒนาระบบ E-Commerce และ CMS ของบริษัท