บทที่ 8
Program Testing

อาจจะกล่าวได้ว่า program testing นั้นเป็นงานที่ต้องอาศัยทั้งศิลปและวิทยาศาสตร์ผนวกเข้าด้วยกัน

กระบวนการ testing นั้น งานขั้นตอนหนึ่งที่สำคัญมากต่อการพัฒนาระบบ ซึ่งบางครั้งเราอาจจะประสบปัญหาอย่างใหญ่หลวงและสูญเสียเวลาไปมากมายต่องานขั้นนี้ งานขั้นตอนนี้ไม่เหมือนกับการ debugging ที่เคยกล่าวมาแล้ว ทั้งนี้เพราะการ debugging ก็คือการขจัด errors หรือ bugs ทั้งปวงให้หมดไป อาจจะกล่าวได้ว่า errors ที่ขจัดไปในการ debugging นี้ส่วนใหญ่จะเป็น syntax errors และ incorrect coding ภายหลังเมื่อขจัด errors หมดไปแล้วและโปรแกรม สามารถปฏิบัติงานได้ผลถูกต้องเราจึงจะดำเนินงานในขั้นตอนต่อไปคือขั้นที่เรียกว่า testing

จะเห็นได้ว่าโปรแกรมซึ่งเครื่องคอมพิวเตอร์รันเข้าไปแล้ว และสามารถรันให้ได้ผลถูกต้องนั้น เราจะสรุปมาทันทีว่า โปรแกรมนั้นถูกต้องเรียบร้อยสมบูรณ์แล้วไม่ได้ เราจำเป็นจะต้องมีมาตรการในการทดสอบให้แน่ใจเสียก่อนว่า โปรแกรมนั้นถูกต้อง ให้ผลสมบูรณ์จริงๆ ไม่ได้มี logic error บางอย่างที่ยังซ่อนเร้นแฝงอยู่

ปัจจัยที่จะกำหนดขอบเขตและขนาดของ testing จะประกอบด้วย

1. ความสำคัญของระดับความถูกต้องในโปรแกรมนั้น
2. จำนวนครั้งของการนำโปรแกรมนั้นไปใช้งาน
3. ระยะเวลาที่นำโปรแกรมนั้นไปใช้งาน

จะสังเกตได้ว่าโปรแกรมซึ่งเขียนขึ้นเพื่อใช้งานเพียงครั้งหรือสองครั้งแล้วก็ทิ้งไปนั้นจะมีการ test น้อยกว่าโปรแกรมประเภทที่นำไปใช้งานอยู่บ่อย ๆ ปัจจัยถัดไปก็คือระดับความถูกต้อง ในงานบางอันนั้นจะต้องให้มีความถูกต้องอยู่ในระดับสูงมาก นั่นย่อมหมายความว่า เราจำเป็นจะต้อง test มากขึ้น

จำนวนข้อมูลที่จะทดสอบนั้น ก็นับว่าเป็นปัจจัยหนึ่งที่ต้องคำนึงถึงในการ testing เราจะใช้จำนวนระเบียนข้อมูลเป็นมาตรฐานในการกำหนด data ที่จะทดสอบไม่ได้ ตัวอย่างเช่น งานชิ้นหนึ่ง testing โดยใช้ข้อมูลอยู่ 100 ระเบียนข้อมูล แต่ปรากฏว่าเมื่อเราพิจารณารายละเอียดของระเบียนข้อมูลทั้ง 100 ระเบียนแล้ว ปรากฏว่า 90 ระเบียนข้อมูลอยู่ในประเภทเดียวกัน ส่วน

อีก 10 ระเบียนข้อมูลก็อยู่ในกรณีเดียวกัน ดังนั้นเราสรุปได้ว่า การทดสอบโดยใช้ข้อมูลในตัวอย่างนี้จะมีข้อมูลแค่ 2 case เท่านั้นเอง วิธีการกำหนดว่าจะใช้ข้อมูลคือ test data จะไม่กำหนดด้วยจำนวนของระเบียนข้อมูล แต่จะกำหนดด้วยจำนวน case ของระเบียนข้อมูลทั้งหมดซึ่งจะต้องครอบคลุมทุกกรณีที่เป็นไปได้ ตัวอย่างเช่น ในการเขียนโปรแกรมเพื่อคำนวณหาค่าน้ำประปาที่จะต้องเสียนั้น เรามีอัตราการคิดจำนวนเงินดังนี้คือ

0 - 10 ยูนิต	ใช้ฟรี
11 - 50	ยูนิตละ 3 บาท
51 - 100	ยูนิตละ 4 บาท
มากกว่า 100 ยูนิต	ยูนิตละ 5 บาท

ตามปัญหานี้การกำหนดระเบียนข้อมูลของผู้ใช้น้ำประปานั้นก็ทำการกำหนดเพียง 5 cases คือ ผู้ใช้น้ำประปาไม่เกิน 10 ยูนิต, ผู้ใช้น้ำประปาไม่เกิน 50 ยูนิต แต่สูงกว่า 11 ยูนิต, ผู้ใช้น้ำประปาในช่วง 51-100 ยูนิต และผู้ใช้น้ำประปาสูงกว่า 100 ยูนิต

Exhausive Testing

Exhausive testing หมายถึงการ test ที่มีข้อจำกัดอันเนื่องจากปัจจัยต่างๆ เช่นค่าใช้จ่าย หรืออาจจะเป็นการ test ซึ่งประสบปัญหาว่าไม่สามารถทำได้กับข้อมูลทุก case ที่เป็นไปได้ดังที่กล่าวมาแล้ว ยกตัวอย่างเช่น ในโปรแกรมภาพที่ 2.1 นั้น เรามีตัวแปรปรากฏอยู่ในโปรแกรมนั้น 2 ตัว คือ X และ Y โดยสมมติว่า เก็บข้อมูลในรูปของตลข 32 บิท ดังนั้นตัวแปร X และ Y จะมีค่าเป็นไปได้ถึงตัวแปรละ 2^{32} จำนวนที่เป็นไปได้ ซึ่งถ้าคิดรวมระหว่าง X และ Y แล้วจะได้ถึง $2^{32} \times 2^{32} = 2^{64}$ ค่าที่เป็นไปได้ สมมติว่าโปรแกรมที่จะดำเนินการนี้ใช้เวลาในการรันครั้งละ 1 millisecond แล้วเราจะต้องใช้เวลาในการ test ถึง 50 ร้อยล้านปี จากตัวอย่างที่นำมาแสดง ให้ดูนี้ทำให้เราสามารถสรุปได้ว่าในบางกรณีการจะใช้ all possible case of data นั้นเป็นไปไม่ได้ เนื่องจากข้อจำกัดดังที่กล่าวมานี้ นอกจากนี้ในบางสถานะการณ์ลักษณะการทำงานภายในของโปรแกรมจะเป็นสาเหตุทำให้เกิดกรณีของ exhausive testing ได้ดังตัวอย่างภาพที่ 2.2และ 2.3

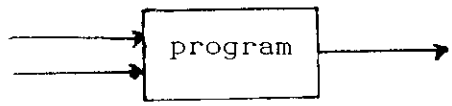

Figure 2.1

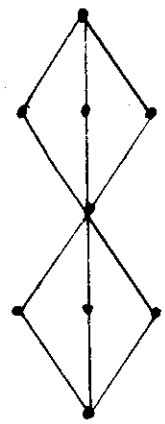

Figure 2.3

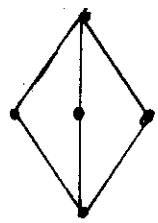

Figure 2.2

จากภาพที่ 2.2 จะเห็นได้ว่า ในไดอะแกรมนั้นมีทางที่จะเป็นไปได้ในการปฏิบัติถึง3 ทางด้วยกัน ถ้าหากเรานำเอาไดอะแกรมภาพนี้มาซ้อนเป็น 2 รูป จะปรากฏดังภาพที่ 2.3 ซึ่งจะประกอบด้วย ถึง 9 หนทาง และถ้าเรานำภาพที่ 2.3 มาซ้อนกันเป็น 2 รูป (เท่ากับ 4 รูปของภาพ 2.2) เราจะได้จำนวนหนทางทั้งหมด เท่ากับ 9^2 ทาง และถ้าหากปฏิบัติการของ 9^2 ทางนี้เราได้ดำเนินการใน loop ถึง 10 ครั้ง เราจะได้จำนวนปฏิบัติงาน $(9^2)^{10}$ ทางที่เป็นไปได้ดังนั้นเงื่อนไขของตัวอย่างนี้ก็คงจะเข้าข่าย exhausive testing

ในกรณีที่เกิดปัญหาของ exhausive testing เราก็คงจะต้องเลือกวิธีการ test ที่ชาญฉลาดขึ้นเข้ามาช่วย

Testing Methods

กฎเกณฑ์ทั่ว ๆ ไปของการ testing ดังที่ได้กล่าวมาแล้วว่า เราจะ test กับโปรแกรม โดยใช้ test data ที่กำหนดไว้แล้ว ยกเว้นแต่เฉพาะกรณีที่เกิดปัญหา exhausive testing โดยปกติแล้วการ testing จะมีโครงสร้างแบ่งออกเป็นแนวใดแนวหนึ่งใน 2 แนวดังต่อไปนี้คือ

1. **Bottom-Up testing**

วิธีนี้เป็นวิธีที่เรานิยมใช้กันในการ test โปรแกรม ลักษณะของการ test โดยวิธีนี้จะคล้าย ๆ กับการสไตล์การเขียนโปรแกรมในรูปแบบของ Bottom-Up ที่เราเคยรู้จักกันมาก่อนแล้ว โดยเริ่มการตรวจสอบจากโมดูลในระดับล่างที่สุดเสียก่อน ภายหลังเมื่อตรวจสอบเรียบร้อยแล้วจึงค่อย ๆ ไล่ขึ้นมาสู่โมดูลในระดับบน ลำดับปฏิบัติการ test นี้จะดำเนินตามลำดับล่างขึ้นมาเรื่อยๆจนกระทั่งสิ้นสุดโปรแกรม กรรมวิธีนี้บางคนอาจจะไม่ชอบ เพราะไม่ต้องกับนิสัยในการพิจารณาการไหลของงาน ซึ่งมักจะพิจารณาจากระดับบนมาสู่ล่าง กรรมวิธีนี้มีข้อเสียในส่วนของการกำหนด test data ใหม่ทุกครั้ง ที่มีการเชื่อมโยงกับส่วนอื่นของซับรูทีน

2. **Top-Down Testing**

การใช้ Top-Down Testing มีผลมาจากรูปแบบของ Top-Down Design ที่ใช้เป็นรูปแบบที่มี test เป็นลักษณะเดียวกับการเขียนโปรแกรมแบบ Top-Down โดยปกติการ test แบบนี้เป็นวิธีที่เราใช้กันบ่อย ๆ การกำหนด data เพื่อใช้ test จะอยู่ในรูป pooling data คือ data ที่ใช้ test นั้นสามารถนำไปใช้ตั้งแต่ตอนเริ่มต้นของโปรแกรมหลักจนถึงส่วนของโมดูลย่อยต่าง ๆ ซึ่งแตกต่างไป จากกรรมวิธีของ Bottom Up ที่จะกำหนด test data ของแต่ละโมดูล แยกแตกต่างไปต่างหากเลย ซึ่งอาจจะเรียกว่าเป็นกรรมวิธีที่ใช้ separate test data ไม่ใช่ pooling test data ข้อดีอีกประการหนึ่งของกรรมวิธีแบบ Top Down ก็คือมีการ test main logic ของโปรแกรมตลอดจนความสัมพันธ์กับโมดูลย่อยอื่น ๆ ในขณะที่วิธี Bottom Up ไม่สามารถทำได้ และกรรมวิธีของ Top Down สามารถ Test ได้ในลักษณะของ distributed ในขณะวิธี Bottom Up นั้นจะต้องรอให้ระบบงานนั้นเสร็จเรียบร้อยเสียก่อนจึงจะ test ได้ ดังนั้นในแต่ละช่วงของการ test แบบ Top Down เราจึงได้ผลออกมาตรวจสอบไปตลอดตั้งแต่เริ่มงานจนถึงสุดท้ายและอาจจะมีการเพิ่มโมดูลย่อยเข้ามาภายหลังได้โดยไม่มีผลกระทบต่อการ testing ในระดับบน

Types of Test Data

การสร้าง test data จะแบ่งออกเป็น 3 ประเภทคือ

1. constructed data
2. actual data with modification
3. actual data in volume

1. **constructed data** นับเป็นข้อมูลประเภทแรกที่เรารู้จักคุ้นเคย ข้อมูลประเภทนี้หมายถึงข้อมูลที่โปรแกรมเมอร์สร้างขึ้นมาเอง เพื่อใช้รันกับโปรแกรม เรายังแบ่ง constructed data ออกมาเป็น 2 ประเภทคือ controlled data และ random data

controlled data จะถูกใช้ถ้าหากเราพบว่า โปรแกรมอยู่ในสภาพที่สามารถทำงานด้วยดี ข้อมูลประเภทนี้จะมีข้อดีฝนแง่ที่ว่า เราสามารถสร้างได้เหมาะสมกับสถานะการณ์ของข้อมูลจริงๆ ที่จะเกิดขึ้น และทั้งยังสามารถสร้างข้อมูลประเภทที่อาจจะเกิดขึ้นได้ แต่โอกาสจะเกิดมีน้อย ในลักษณะเช่นนี้เราสามารถสร้าง test data ที่ครอบคลุมทุกกรณีในสถานะการณ์จริงและสามารถกำหนดข้อมูลขั้นต่ำ-ขั้นสูง ได้ด้วย โดยลักษณะเช่นนี้ทำให้เรา test โปรแกรมได้กับข้อมูลทุก cases โดยใช้เวลาน้อย

random test data ถูกสร้างขึ้นโดยใช้โปรแกรมเขียนสร้างขึ้นมาเพื่อให้ตรงกับวัตถุประสงค์ของการใช้งาน ข้อดีของ random test data คือสร้างขึ้นใช้ได้ง่ายและสามารถสร้างได้มากเท่าใดก็ได้ตามที่ต้องการ แต่ก็ยังมีข้อเสีย คือเราไม่สามารถจะตรวจสอบข้อมูลได้ หรือถ้าจะตรวจสอบ test data ก็ทำได้ยาก

2. **modified actual data** นับว่าเป็น data ที่มีข้อดีกว่า data ทั้ง 2 ประเภทที่กล่าวมาแล้ว แต่ควรมีข้อระมัดระวังในการเลือก modified actual data เพื่อให้ครอบคลุมทุกสถานะการณ์ที่เป็นไปได้ของข้อมูลจริง ๆ ข้อดีของวิธีการใช้ข้อมูลประเภทนี้ก็คือ จะทำให้ลดปริมาณข้อมูลที่จะทดสอบแทนที่จะใช้ real data จุดประสงค์ที่เป็นประโยชน์ที่ได้รับอีกประการของการใช้ modified actual data ก็คือ ใช้สำหรับทดสอบซับรูทีนต่าง ๆ

3. **actual data** เป็นข้อมูลที่เรามักจะใช้ในการทดสอบเปรียบเทียบระหว่างระบบใหม่ซึ่งเราสร้างขึ้นมากับระบบเก่าเพื่อดูว่าระบบใหม่สามารถทำงานได้ถูกต้องหรือไม่ ลักษณะการ test แบบนี้เรียกว่า parallel run test ซึ่งเป็นวิธีการใช้ปริมาณข้อมูลมาก ผลจากการรันระหว่างระบบใหม่กับระบบเก่านำมาเปรียบเทียบกัน การใช้ข้อมูลประเภทนี้ถ้าไม่ใช่งานประเภท parallel run แล้วก็ไม่ควรใช้วิธีการนี้ เพราะจะเสียเวลามากและไม่มี output ให้เปรียบเทียบ

ข้อมูลแต่ละประเภททั้ง 3 ประเภทดังกล่าวมาแล้วนั้น จะมีความเหมาะสมกับสภาพของปัญหาซึ่งแตกต่างกันไป ดังนั้นจึงอยู่ในวินิจฉัยของโปรแกรมเมอร์ว่าควรจะเลือกใช้ data ประเภทใด

ข้อมูลที่จะนำมาทดสอบนั้น เราจะแบ่งได้ออกเป็น 3 กลุ่มดังนี้คือ

1. Testing the normal cases
2. Testing the extremes
3. Testing the exceptions

1. **Testing Normal Cases**

หมายถึง ชุดของข้อมูลซึ่งอยู่ในกรณีของรูปแบบข้อมูลทั่ว ๆ ไป ซึ่งโปรแกรมปฏิบัติงานอยู่เป็นปกติเป็นส่วนใหญ่

2. **Testing Extremes**

กระบวนการนี้เป็นการ tesing ลำดับที่ 2 ถัดจากลำดับแรกที่ใช้ข้อมูลประเภท normal cases ได้ตรวจสอบไปแล้ว ข้อมูลประเภทนี้ยังถือว่าเป็น valid data อยู่ แต่เป็น data ที่ค่า extreme lower value หรือ extreme upper value ในกรณีของข้อมูลคณิตศาสตร์ นั้นก็หมายถึงการกำหนดค่าของข้อมูลที่เล็กที่สุด หรือต่ำที่สุดที่เครื่องสามารถรับไปปฏิบัติการกับโปรแกรมนั้น ๆ

กรรมวิธีของการใช้ค่า extreme ในการ test นั้นเรียกว่า boundary test การ test เพื่อตรวจสอบค่า extreme ว่าจะเกิดปัญหาในการทำงานของโปรแกรมหรือไม่ ยกตัวอย่างเช่น สมมติว่าเป็นการ update file นั้นปรากฏว่า ใน transaction file มีระเบียนข้อมูลเพียงอันเดียวหรือไม่มีเลย โปรแกรมจะเกิดปัญหาหรือไม่ หรือไม่กรณีของการคิดดอกเบี้ย ถ้าสมมติว่าอัตราดอกเบี้ยเป็น 0 แล้วจะเกิดอะไร

กับโปรแกรมของเราในการทำงาน ตัวอย่างการกำหนด extreme ในโปรแกรมหนึ่งซึ่งมีตัวแปรอยู่ 5 ตัว แต่ละตัวเป็นเลขจำนวนเต็มบวกและเป็นเลขหลักเดียว เราจะกำหนดค่า extreme value ดังนี้คือ

	A	B	C	D
Test group 1	0	0	0	0
Test group 2	9	9	9	9

ตัวอย่างข้อมูลที่กำหนดมานี้จะเป็น extreme value ถ้าหากว่าภายในโปรแกรมมีการคำนวณในลักษณะนี้คือ

ANSWER = A + B + C + D

แต่ถ้าหากว่าเงื่อนไขการคำนวณเปลี่ยนไปเป็น

$$ANSWER = \left(\frac{A+B}{C}\right)**D$$

แล้วค่าของ C = 1 จะถือค่าเล็กที่สุดที่เป็นไปได้

การจะเลือก extreme value มาทดสอบนั้น ผู้เลือกจำเป็นจะต้องมีความคุ้นเคยกับสูตรหรือกระบวนการต่าง ๆ ที่ใช้ปฏิบัติการในโปรแกรม การกำหนดค่า extreme value นั้นเราไม่ได้พิจารณาแต่ input data เท่านั้น บางครั้งเราต้องพิจารณาถึง output data ประกอบด้วย

3. **Testing Exceptions**

เป็นขั้นตอนสุดท้ายของการทดสอบโปรแกรม จะเห็นได้ว่าโดยโครงสร้างของโปรแกรมทั้งหลายนั้นมักจะมีการเขียนโดยตั้งข้อจำกัดของข้อมูลที่จะรับเข้ามาปฏิบัติงาน ยกตัวอย่างเช่น ไม่สามารถรับค่าติดลบ หรือเลข 0 เข้ามาปฏิบัติงานได้ แต่ถ้าสมมติในสถานะการณ์จริงเกิดมีปัญหาว่าข้อมูลลักษณะดังกล่าวเกิดปรากฏขึ้นมาแล้ว จะเกิดผลอะไรขึ้นกับโปรแกรม หรือถ้าเราเกิด inputข้อมูลที่เล็กหรือโตกว่าที่เครื่องจะยอมรับได้จะเกิดอะไร ลักษณะดังตัวอย่างที่ยกมานี้ล้วนแต่เป็นสภาพเลวร้ายที่อาจเกิดขึ้นได้ ดังนั้น

ทางออกที่ดีของเราก็คือเขียนโปรแกรมให้อยู่ในลักษณะที่เตรียมรับข้อมูลประเภทนี้ไว้ โดยจะทำการ reject data ที่โปรแกรมไม่สามารถนำไปปฏิบัติการได้

ตัวอย่างของการ Testing

ตัวอย่างที่ 1

ภาพ 2.4 จะแสดงภาพของกล่องซึ่งมีเส้นทแยงมุมลากผ่านจากจุดหนึ่งของด้านบนไปยังจุดหนึ่งของด้านล่าง

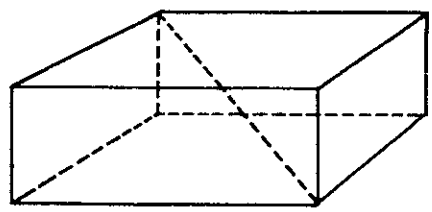

Figure 2.4

โดยที่เส้นทแยงมุมดังกล่าวจะมีขนาด $(A^2 + C^2)^2 + B^2$ การจะกำหนด test data โดยกำหนด normal cases, extremes และ exceptions ดังนี้

	Sizes of a Box			Remarks
1.	1	1	1	A good first test.
2.	1	2	3	Another normal test.
3.	0	0	0	Should give you a zero answers
4.	0	1	2	Not a Box. What happens?
5.	1	0	3	Not a Box. What happens?
6.	2	1	0	Not a Box. What happens?
7.	1	-6	3	Incorrect data.

จากข้อมูลใน test data ที่แสดงมาให้ดูนั้น test 1 และ 2 นั้นว่าเป็น normal cases ส่วน test 3-7 นั้นจะใช้เป็น extremes หรือ exceptions ถ้าหากว่าทั้ง 7 test นี้ให้ผลถูกต้องในการคำนวณเราเชื่อได้ว่าโปรแกรมนี้สามารถทำงานได้ถูกต้อง ขอให้นักศึกษาลองสร้าง test data เองจากตัวอย่างนี้

ตัวอย่างที่ 2 สมการกำลังสองคือ $AX^2 + BX + C = 0$

นั้นเราจะคำนวณหาค่า X จากสูตร $R = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

โดยให้โปรแกรมรับค่าของ A, B, C เข้าไปคำนวณออกมา เราจะใช้ test data ดังนี้

	Coefficients			Remarks
1.	1	1	-2	A good first tesdt.
2.	1	0	0.25	Another normal test.
3.	0	0	0	What happens here?
4.	0	2	1	Should get only one root
5.	2	1	0	Should be OK.
6.	1	1	1	Complex roots.
7.	0	0	2	Not a valid equation.
8.	0	2	0	Should get one root.
9.	2	0	0	Should get two roots.

แบบฝึกหัด

1. จงอธิบายถึงความแตกต่างระหว่าง debugging กับการ testing
2. การจะ testing โปรแกรมที่มีซับรูทีนซ้อนนั้น เราควรจะทำอย่างไร
3. เราสามารถกำหนด test data ได้กี่แบบ อะไรบ้าง
4. จงบอกถึงประโยชน์ของการใช้ actual data จำนวนมากในการ testing
5. จากเครื่องคอมพิวเตอร์ที่ท่านใช้อยู่ จงตรวจสอบค่าของข้อมูลที่เป็นไปได้ ซึ่งจะนำไปใช้กับฟังก์ชันแต่ละอันนี้คือ

   Log ฐาน 10 Log ฐาน C

   Cos. Sine

   Hyperbolic sine
6. จงหา error range ของ data ในฟังก์ชันต่อไปนี้

   SQRT, SIN, TAN
7. กำหนดให้ใช้สูตรต่อไปนี้

$$Yn = \frac{ab}{cd} - \frac{c + d}{a - b}$$

   จงเตรียม test data สำหรับกรณีต่อไปนี้

   a. the normal cases

   b. the extreme cases

   c. the exceptional cases
8. จงสร้าง test data สำหรับโปรแกรมต่อไปนี้ คือ

โปรแกรมทำหน้าที่สำรองที่นั่งบนเครื่องบิน โดยที่ในแต่ละวันจะมีอยู่ 5 flights คือ flights หมายเลข 142, 148, 153, 181 และ 191 บริษัทการบินแห่งนั้นจะรับสำรองที่นั่งก่อนล่วงหน้า 1 สัปดาห์ โดยที่โปรแกรมที่เขียนขึ้นนี้จะต้องทำหน้าที่รับสำรองที่นั่ง ถ้าเครื่องยังไม่เต็ม,หรือถอดถอนที่นั่ง ถ้าลูกค้าต้องการเลิกสำรอง, หรือปฏิเสธการสำรองถ้าที่นั่งเต็มแล้ว สมมติว่าใน flight หนึ่ง ๆ นั้น มีอยู่ 6 ที่นั่ง โดยแบ่งเป็น 3 ชั้น

คือ first, coach, student โปรแกรมที่เขียนขึ้นนี้สามารถยืดหยุ่นในการสำรองได้คือ ถ้า flight ใด ชั้น first เต็มก็ให้เลือกชั้น coach ได้ และขณะเดียวกันถ้าชั้น coach เต็มก็มีสิทธิจะไปเลือกชั้น student ได้

9. จงเขียนโปรแกรมเพื่อหาค่ารากที่สองจากสมการ quadratic form โดยใช้สูตร

$$\text{Roots} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

โดยใช้ test data ต่อไปนี้

a	b	c	a	b	c
1	2	1	6	5	-4
1	-1	-6	$6\star10^{30}$	$5\star10^{30}$	$-4\star10^{30}$
1	-10	1	10^{-30}	-10^{30}	10^{30}
1	-1000	1	1.0	-4.0	3.9999999
1	-10000	1	1.0	-4.0	4.0
1	2	-3	1.0	100.0001	0.01

10. สมมติว่า คอมพิวเตอร์ที่ท่านใช้อยู่มีความสามารถในการรับค่าได้สูงสุด 6 ตำแหน่ง (significant digits) แต่ถ้าท่านต้องการความถูกต้องถึง 8 ตำแหน่ง เราจะสามารถทำได้โดยการแยกค่าของข้อมูลออกเป็น 2 ส่วน แล้วนำไปปฏิบัติการดังตัวอย่างต่อไปนี้

00123456		0012	3456
	becomes		
65555556		6543	2100
		6555	5556

จงเขียนโปรแกรมเพื่อปฏิบัติงานนี้โดยการกำหนด test data ของท่านเองทดสอบโปรแกรมนี้