April 2019

Fundamental IT Engineer Examination (Afternoon)

ให้ทำข้อสอบตามรายละเอียดต่อไปนี้

หมายเลขคำถาม	Q1 – Q6	Q7 , Q8
การเลือกคำถาม	ทำทุกข้อ	เลือก 1 ใน 2
เวลาสอบ	13:30 - 16:00 (150 นาที)	

ข้อปฏิบัติ:

1. ให้ใช้ดินสอตอบ ถ้าต้องการเปลี่ยนคำตอบ ให้ลบคำตอบเก่าให้สะอาดก่อนโดยไม่ให้มีคราบยางลบหลงเหลือ
2. ให้ทำเครื่องหมายบอกข้อมูลผู้สอบและคำตอบของแบบทดสอบ ตามคำสั่งด้านล่างอย่างเคร่งครัด หากทำเครื่องหมายไม่เหมาะสม คำตอบของท่านอาจไม่ได้รับการตรวจ ห้ามทำเครื่องหมาย หรือเขียนตอบนอกพื้นที่ที่กำหนดไว้

(1) **หมายเลขผู้สอบ (Examinee Number)**
ให้เขียนหมายเลขผู้สอบลงในช่องที่เตรียมไว้ให้ และทำเครื่องหมายในช่องว่างที่เหมาะสมที่อยู่ใต้ตัวเลขแต่ละตัว

(2) **วันเกิด (Date of Birth)**
ให้เขียนวันเกิดของผู้สอบ (เป็นตัวเลข) ลงในช่องที่เตรียมไว้ ให้ตรงกับที่พิมพ์อยู่ในบัตรเข้าห้องสอบ และทำเครื่องหมายในช่องว่างที่เหมาะสมที่อยู่ใต้ตัวเลขแต่ละตัว

(3) **การเลือกคำตอบ**
สำหรับ Q7 และ Q8 ให้เลือก ⓢ ที่คุณเลือกที่จะตอบในช่อง “Selection Column” บนกระดาษคำตอบของคุณ

(4) **คำตอบ**
ให้ทำเครื่องหมายตรงคำตอบที่เลือกตามตัวอย่างที่แสดงอยู่ด้านล่าง
[คำถามตัวอย่าง]
ข้อสอบวัดระดับ Fundamental IT Engineer Examination รอบฤดูใบไม้ร่วงจัดในเดือนใด

กลุ่มคำตอบ

a) มีนาคม b) เมษายน c) พฤษภาคม d) มิถุนายน

เนื่องจากคำตอบที่ถูกเป็น "b) เมษายน" ดังนั้นให้ทำเครื่องหมายดังข้างล่างนี้

[ตัวอย่างคำตอบ]

Sample	ⓐ ● ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ

Do not open the exam booklet until instructed to do so.
Inquiries about the exam questions will not be answered.

Notations used in the pseudo-language

In questions that use pseudo-language, the following notations are used unless otherwise stated:

[Declaration, comment, and process]

Notation		Description
type: *var1*, …, *array1*[], …		Declares variables *var1*, … , and/or arrays *array1*[], … , by data *type* such as INT and CHAR.
FUNCTION: *function*(*type*: *arg1*, …)		Declares a *function* and its arguments *arg1*, … .
/* comment */		Describes a comment.
Process	*variable* ← *expression* ;	Assigns the value of the *expression* to the *variable*.
	function (*arg1*, …) ;	Calls the *function* by passing / receiving the arguments *arg1*, … .

	IF (*condition*) {
	process1
	}
	ELSE {
	process2
	}
		Indicates the selection process.
		If the *condition* is true, then *process1* is executed.
		If the *condition* is false, then *process2* is executed, when the optional ELSE clause is present.

	WHILE (*condition*) {
	process
	}
		Indicates the "WHILE" iteration process.
		While the *condition* is true, the *process* is executed repeatedly.

	DO { *process* } WHILE (*condition*);	Indicates the "DO - WHILE" iteration process. The *process* is executed once, and then while the *condition* is true, the *process* is executed repeatedly.

	FOR (*init* ; *condition* ; *incr*) {
	process
	}
		Indicates the "FOR" iteration process.
		While the *condition* is true, the *process* is executed repeatedly.
		At the start of the first iteration, the process *init* is executed before testing the *condition*.
		At the end of each iteration, the process *incr* is executed before testing the *condition*.

[Logical constants]

true, false

[Operators and their precedence]

Type of operation	Unary	Arithmetic		Relational	Logical	
Operators	+, −, not	×, ÷, %	+, −	>, <, ≥, ≤, =, ≠	and	or
Precedence	High ◄				► Low	

Note: With division of integers, an integer quotient is returned as a result.
The "%" operator indicates a remainder operation.

คำถาม **Q1** ถึง **Q6** เป็นคำถาม **บังคับ** ให้ตอบทุกคำถาม

Q1. อ่านคำอธิบายเกี่ยวกับรูปแบบการเข้ารหัสลับแบบไฮบริด แล้วตอบคำถามย่อย

ในระบบเข้ารหัสลับแบบกุญแจสาธารณะ (public-key cryptosystem) การเข้ารหัสลับแบบไฮบริด (hybrid encryption) ใช้ทั้งกุญแจสาธารณะและกุญแจสมมาตร (symmetric-key) ซึ่งในรูปแบบไฮบริดนี้ กุญแจสาธารณะถูกใช้ในการห่อหุ้มกุญแจ (key encapsulation) ขณะที่กุญแจสมมาตรถูกใช้เพื่อห่อหุ้มข้อมูล (data encapsulation) กุญแจที่ใช้เพื่อห่อหุ้มข้อมูลนี้ถูกห่อหุ้มด้วยการเข้ารหัสลับแบบกุญแจสาธารณะและถูกเรียกว่ากุญแจเซสชัน (session key)
การเข้ารหัสลับแบบไฮบริดมีรูปแบบดังที่แสดงอยู่ด้านล่างนี้

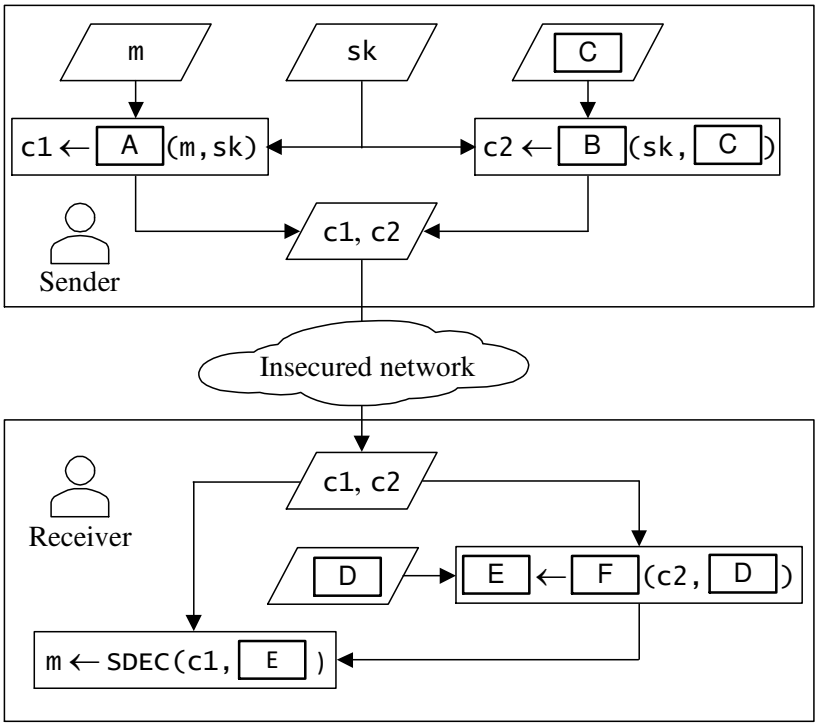
[กระบวนการทางผู้ส่ง]

(1) สร้างกุญแจเซสชัน sk สำหรับการเข้ารหัสลับ โดยกุญแจเซสชัน sk นี้สามารถสร้างขึ้นมาแบบสุ่มได้
(2) นำข้อความปกติ (plain message) m มาเข้ารหัสลับด้วยกุญแจเซสชัน sk โดยใช้ฟังก์ชันเข้ารหัสลับแบบกุญแจสมมาตร SENC
(3) นำกุญแจเซสชัน sk มาเข้ารหัสลับโดยใช้ฟังก์ชันเข้ารหัสลับแบบกุญแจสาธารณะ PENC
(4) ส่งข้อความที่ถูกเข้ารหัสลับ c1 กละกุญแจเซสชันที่ถูกเข้ารหัสลับ c2 ให้กับผู้รับ

[กระบวนการทางผู้รับ]

(1) รับข้อความที่ถูกเข้ารหัสลับ c1 และกุญแจเซสชันที่ถูกเข้ารหัสลับ c2
(2) ถอดรหัสลับกุญแจเซสชัน c2 ที่ที่ได้รับมา
(3) ถอดรหัสลับข้อความ c1 ที่ได้รับมา ได้เป็นข้อความปกติ m ที่ถูกถอดรหัสแล้ว

รูปที่ 1 แสดงรูปแบบการเข้ารหัสลับแบบไฮบริด โดยรูปสี่เหลี่ยมมุมฉาก (rectangle) ใช้แสดงฟังก์ชัน และสี่เหลี่ยมด้านขนาน (parallelogram) ใช้แสดงตัวแปร



รูปที่ 1 รูปแบบการเข้ารหัสลับแบบไฮบริด

คำต่อไปนี้ถูกใช้อยู่ในรูปที่ 1

คำที่ใช้	คำอธิบาย
ฟังก์ชัน	
PENC	ฟังก์ชันเข้ารหัสลับแบบกุญแจสาธารณะ
PDEC	ฟังก์ชันถอดรหัสลับแบบกุญแจสาธารณะ (ตรงข้ามกับ PENC)
SENC	ฟังก์ชันเข้ารหัสลับแบบกุญแจสมมารต
SDEC	ฟังก์ชันถอดรหัสลับแบบกุญแจสมมารต (ตรงข้ามกับ SENC)
ตัวแปร	
m	ข้อความปกติ (plain message)
c1, c2	ข้อความที่ถูกเข้ารหัสลับไว้ (encrypted message)
ssk	กุญแจส่วนตัวของผู้ส่ง (sender's private key)
spk	กุญแจสาธารณะของผู้ส่ง (sender's public key)
rsk	กุญแจส่วนตัวของผู้รับ (receiver's private key)
rpk	กุญแจสาธารณะของผู้รับ (receiver's public key)
sk	กุญแจเซสชัน (session key)

คำถามย่อย

จากกลุ่มคำตอบด้านล่างนี้ ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงใน [ ] ในรูปด้านบน

กลุ่มคำตอบสำหรับ A, B และ F

a) PDEC
b) PENC
c) SDEC
d) SENC

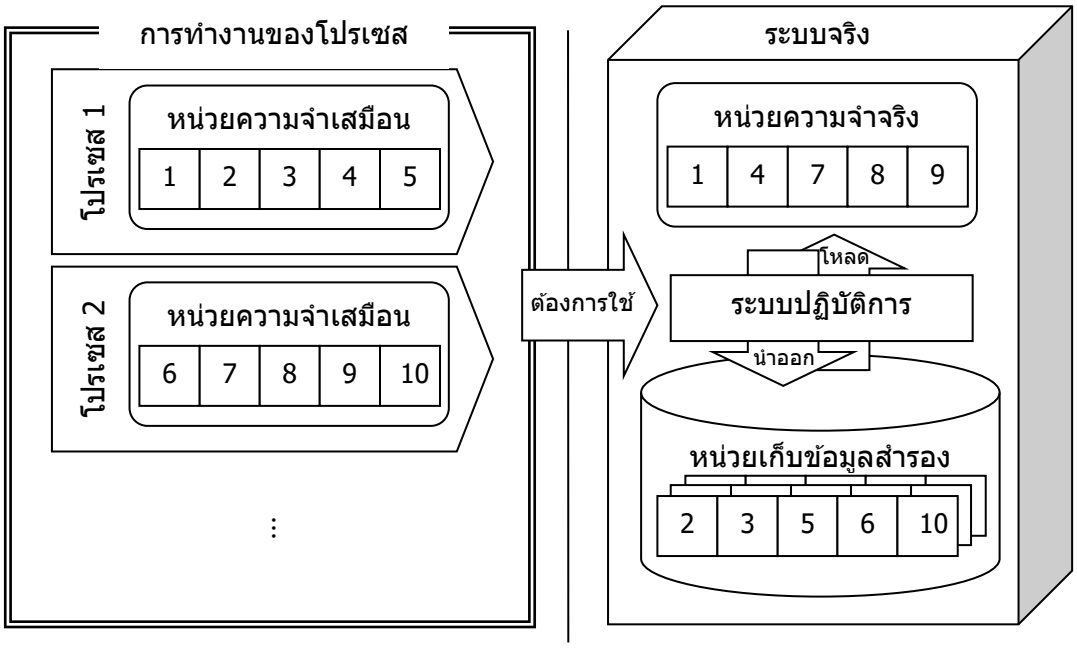
กลุ่มคำตอบสำหรับ C, D และ E

a) rpk
b) rsk
c) sk
d) spk
e) ssk

Q2. อ่านคำอธิบายเกี่ยวกับหน่วยความจำเสมือน (virtual memory) และการสลับหน้า (paging) แล้วตอบคำถามย่อย 1 และ 2

หน่วยความจำเสมือนเป็นเทคนิคที่ช่วยให้โปรเซสใช้หน่วยความจำได้มากกว่าที่ระบบมีอยู่จริง ด้วยเทคนิคนี้ แต่ละโปรเซสมีหน่วยความจำเสมือนของตัวเองที่ประกอบด้วยส่วนย่อย ๆ ที่มีขนาดเท่ากัน เรียกว่าหน้าหรือเพจ (page) ขณะที่หน่วยความจำจริงถูกแบ่งเป็นส่วนย่อย ๆ ที่มีขนาดเท่ากันเรียกว่ากรอบหรือเฟรม (frame) เพจที่ถูกใช้งานอยู่หรือถูกใช้งานมาไม่นานก่อนหน้านี้ถูกเก็บไว้ในกรอบของหน้าหรือเพจเฟรม (page frame) ในหน่วยความจำจริง และเพจที่เหลือถูกเก็บไว้ในหน่วยเก็บข้อมูลสำรองเช่นฮาร์ดดิสก์ การจับคู่ระหว่างเพจกับเพจเฟรมถูกบริหารจัดการโดยระบบปฏิบัติการ เมื่อโปรเซสหนึ่งต้องการใช้เพจที่ไม่ได้อยู่ในหน่วยความจำจริง จะเกิดการผิดหน้าหรือเพจฟอลต์ (page fault) ขึ้น และระบบปฏิบัติการจะโหลดเพจที่ต้องการขึ้นมาจากหน่วยเก็บข้อมูลสำรองแล้วเก็บไว้ในเพจเฟรม กลยุทธ์จัดการหน่วยความจำเช่นนี้รู้จักกันในชื่อดีมานด์เพจจิง (demand paging) หรือการสลับหน้าเมื่อต้องการใช้งาน

รูปที่ 1 แสดงแนวคิดของการสลับหน้าเมื่อต้องการใช้งานหรือดีมานด์เพจจิง



รูปที่ 1 แนวคิดของดีมานด์เพจจิง

เมื่อเพจหนึ่งไม่อยู่ในหน่วยความจำจริง จะต้องใช้เวลาเพื่อโหลดเพจนั้นจากหน่วยเก็บข้อมูลสำรอง ดังนั้น การเกิดเพจฟอลต์บ่อยครั้งอาจทำให้ประสิทธิภาพลดลงเป็นอย่างมาก
เวลาที่คาดว่าต้องใช้ในการเข้าถึงข้อมูลในเพจหนึ่ง ๆ หรือ T สามารถเขียนเป็นสูตรได้ดังนี้:

$$T = ((1 - R_f) \times T_m) + (R_f \times T_a)$$

ในที่นี้:

R_f : อัตราการเกิดเพจฟอลต์
T_m : เวลาที่ใช้เข้าถึงข้อมูลเมื่อเพจนั้นอยู่ในหน่วยความจำจริง
T_a : เวลาที่ใช้เข้าถึงข้อมูลเมื่อเพจนั้นอยู่ในหน่วยเก็บข้อมูลสำรอง

ในระบบหนึ่งที่มี T_m เป็น 200 นาโนวินาที และ T_a เป็น 4 มิลลิวินาที ตารางที่ 1 แสดงตัวอย่างของอัตราการเกิดเพจฟอลต์และเวลาที่คาดว่าต้องใช้เพื่อเข้าถึงข้อมูล กำหนดให้มีเพจเฟรมว่างเพียงพอในหน่วยความจำจริง

ตารางที่ 1 ตัวอย่างของอัตราการเกิดเพจฟอลต์และเวลาที่คาดว่าต้องใช้เพื่อเข้าถึงข้อมูล

อัตราการเกิดเพจฟอลต์	เวลาที่คาดว่าต้องใช้เพื่อเข้าถึงข้อมูล (*)
1%	40.2 ไมโครวินาที
0.1%	[A] ไมโครวินาที
0.01%	[B] นาโนวินาที

หมายเหตุ: (*) ปัดเศษให้มีตัวเลขนัยสำคัญ 3 ตัว

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในตารางที่ 1

กลุ่มคำตอบ

a) 4.02	b) 4.20	c) 6.00
d) 40.2	e) 42.0	f) 60.0
g) 402	h) 420	i) 600

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายด้านล่าง

[การแทนที่เพจ]

เมื่อเกิดเพจฟอลต์แต่ไม่มีเพจเฟรมว่างเหลืออยู่ ระบบปฏิบัติการจะนำเนื้อหาของเพจเฟรมหนึ่งออกไปยังหน่วยเก็บข้อมูลสำรองเพื่อให้มีพื้นที่ว่างสำหรับเพจที่ต้องการใช้งาน เพจเฟรมที่ถูกนำออกไปนี้เรียกว่าเหยื่อ (victim)

มีหลายอัลกอริธึมที่สามารถใช้เลือกว่าเพจเฟรมใดจะตกเป็นเหยื่อได้ แต่ในที่นี้จะพิจารณาเพียงสองอัลกอริธึมเท่านั้น ได้แก่ เข้าก่อนออกก่อน (FIFO: first-in, first-out) และที่ไม่ถูกเรียกใช้มานานที่สุด (LRU: least recently used)

ในอัลกอริธึม FIFO เพจเฟรมที่เก็บหน้าเดิมไว้เป็นเวลานานที่สุดจะถูกเลือกให้เป็นเหยื่อ ขณะที่ในอัลกอริธึม LRU เพจเฟรมที่เก็บหน้าที่ไม่ถูกเรียกใช้งานมาเป็นเวลานานที่สุดจะถูกเลือกให้เป็นเหยื่อ

รูปที่ 2 แสดงตัวอย่างของระบบหนึ่งที่ทำงานด้วยอัลกอริธึมทั้งสองดังกล่าวแล้วข้างต้น ในระบบนี้มี 3 เพจเฟรม ได้แก่ X, Y และ Z ในหน่วยความจำจริง และโปรเซสที่ทำงานอยู่ในระบบมี 5 เพจ (เพจ 1 ถึง 5) โดยถูกเรียกใช้งานตามลำดับดังนี้ 1 2 3 4 1 2 5 1 2 3 4 5 ในที่นี้ เพจต่าง ๆ ที่ถูกโหลดเมื่อเกิดเพจฟอลต์ถูกขีดเส้นใต้เอาไว้

อัลกอริธึม FIFO

ต้องการใช้เพจ		1	2	3	4	1	2	5	1	2	3	4	5
เพจเฟรม	X	**1**	1	1	**4**	4	4	**5**	5	5	5	5	5
	Y		**2**	2	2	**1**	1	1	1	1	**3**	3	3
	Z			**3**	3	3	**2**	2	2	2	2	**4**	4

อัลกอริธึม LRU

ต้องการใช้เพจ		1	2	3	4	1	2	5	1	2	3	4▼	5
เพจเฟรม	X	**1**	1	1	**4**	4	4	**5**	5				
	Y		**2**	2	2	**1**	1	1	1				
	Z			**3**	3	3	**2**	2	2				

หมายเหตุ: ส่วนที่ถูกแรเงาไว้คือส่วนที่ถูกซ่อนไว้

รูปที่ 2 ตัวอย่างของระบบที่ทำงานตามอัลกอริธึม FIFO และ LRU

ในอัลกอริธึม FIFO เกิดเพจฟอลต์ 9 ครั้งในรูปที่ 2 ขณะที่ในอัลกอริธึม LRU เมื่อถึงจุดที่ต้องการใช้เพจ 4 (ณ จุดที่มี "▼") เพจที่อยู่ในเพจเฟรม [C] จะถูกแทนที่ ซึ่งโดยรวมแล้ว เมื่อใช้อัลกอริธึม LRU จะเกิดเพจฟอลต์ทั้งสิ้น [D] ครั้ง ในรูปที่ 2

รูปที่ 3 แสดงอีกหนึ่งตัวอย่างของระบบที่ทำงานตามสองอัลกอริธึมดังกล่าวข้างต้น โดยรูปที่ 3 นี้ทำงานในลักษณะเดียวกันกับรูปที่ 2 แต่มีเพจเฟรมเพิ่มขึ้นมา 1 เพจเฟรมคือ W

อัลกอริธึม FIFO

ต้องการใช้เพจ		1	2	3	4	1	2	5	1	2	3	4	5
เพจเฟรม	W	**1**	1	1	1	1	1	**5**	5				
	X		**2**	2	2	2	2	2	**1**				
	Y			**3**	3	3	3	3	3				
	Z				**4**	4	4	4	4				

อัลกอริธึม LRU

ต้องการใช้เพจ		1	2	3	4	1	2	5	1	2	3	4	5
เพจเฟรม	W	**1**	1	1	1	1	1	1	1	1	1	1	**5**
	X		**2**	2	2	2	2	2	2	2	2	2	2
	Y			**3**	3	3	3	**5**	5	5	5	**4**	4
	Z				**4**	4	4	4	4	4	**3**	3	3

รูปที่ 2 อีกหนึ่งตัวอย่างของระบบที่ทำงานตามอัลกอริธึม FIFO และ LRU

ในรูปที่ 3 เกิดเพจฟอลต์ทั้งสิ้น E **ครั้งสำหรับอัลกอริธึม FIFO และเกิดเพจฟอลต์ 8 ครั้งสำหรับอัลกอริธึม LRU**

กลุ่มคำตอบสำหรับ C

a) X b) Y c) Z

กลุ่มคำตอบสำหรับ D และ E

a) 5 b) 6 c) 7
d) 8 e) 9 f) 10

Q3. **อ่านคำอธิบายเกี่ยวกับฐานข้อมูลสำหรับการจัดการรายวิชา (course) และเกรด (grade) ของนักศึกษาในมหาวิทยาลัยแห่งหนึ่ง แล้วให้ตอบคำถามย่อย 1 ถึง 3**

มหาวิทยาลัยแห่งหนึ่งใช้ฐานข้อมูลเพื่อเก็บประวัตินักศึกษา รายวิชา กลุ่มเรียนหรือเซคชัน (section) ต่าง ๆ ที่เปิดสอน รวมถึงเกรดที่ได้

ฐานข้อมูลนี้ประกอบด้วยตาราง 4 ตาราง โครงสร้างของตารางรวมทั้งข้อมูลตัวอย่างถูกแสดงอยู่ด้านล่าง ในที่นี้คีย์หลัก (primary key) จะถูกขีดเส้นใต้ไว้

(1) ตารางนักศึกษา (Student table)

StudentNo	Name	Department	Class	Program
1599221	Steve Kam	CS	Sophomore	BSC
1599222	Mathew Ken	CS	Sophomore	BSC
1599223	Allen Strew	CS	Sophomore	BSC
1599224	Stephen Ford	CS	Sophomore	BSC

(2) ตารางรายวิชา (Course table)

CourseNo	CourseName	CreditHours	Department
CS173	Discrete Mathematics	3	CS
CS225	DataStructures	3	CS
CS311	Database Systems	3	CS
CS377	Algorithms	3	CS

(3) ตารางกลุ่มเรียน (Section table)

SectionNo	CourseNo	Instructor	Semester	Year
1112	CS 311	John	Summer	2018
1113	CS 173	Clark	Fall	2018
1114	CS 377	Abraham	Fall	2018
1115	CS 225	Clark	Spring	2019
1116	CS 311	John	Summer	2019

(4) ตารางรายงานเกรด (GradeReport table)

StudentNo	SectionNo	Grade
1599221	1112	B
1599221	1113	C
1599222	1113	B
1599222	1115	A
1599223	1116	(null)
1599224	1112	A
1599224	1114	A

หนึ่งรายวิชา (course) อาจมีได้หนึ่งกลุ่มเรียน (section) หรือมากกว่า โดยมี SectionNo แสดงความเป็นอัตลักษณ์ของแต่ละกลุ่มเรียนหรือเซคชัน
เกรดแสดงถึงผลการเรียนที่นักศึกษาได้รับในแต่ละกลุ่มเรียน มีรูปแบบเป็นรหัสขนาดหนึ่งตัวอักษร เช่น A (ดีเยี่ยม), B (ดี), C (พอใช้), ..., หรือ "null" หากการเรียนไม่สมบูรณ์
นักศึกษาทั้งหมดลงทะเบียนเรียนในอย่างน้อยหนึ่งกลุ่มเรียน นั่นคือจะต้องมี StudentNo ทั้งหมดจากตารางนักศึกษา (Student table) ปรากฏอยู่ในตารางรายงานเกรด (GradeReport table)

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำสั่ง SQL ต่อไปนี้

คำสั่ง SQL1 แสดงรหัสประจำตัวนักศึกษา (student number) ชื่อ (name) และแผนก (department) ของนักศึกษาที่ได้เกรด A ในทุกรายวิชาที่เรียน

```
-- SQL1 --
SELECT StudentNo, Name, Department
FROM Student S
WHERE [    A    ]
      (SELECT StudentNo
       FROM GradeReport
       WHERE StudentNo = S.StudentNo AND [    B    ])
```

จากข้อมูลตัวอย่างของแต่ละตารางที่แสดงในคำอธิบาย SQL1 จะให้ผลลัพธ์ดังนี้:

StudentNo	Name	Department
1599224	Stephen Ford	CS

กลุ่มคำตอบสำหรับ A

a) `EXISTS`
b) `IN`
c) `NOT EXISTS`
d) `NOT IN`

กลุ่มคำตอบสำหรับ B

a) `Grade != 'A'`
b) `Grade < 'A'`
c) `Grade = 'A'`

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในรูปที่ 1

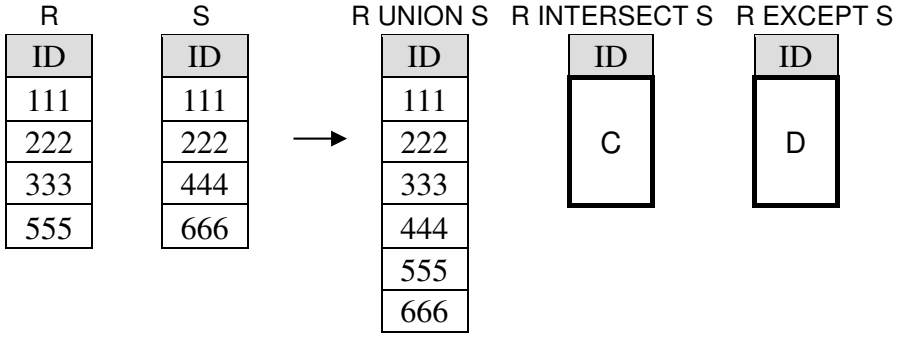
ใน SQL: UNION, INTERSECT และ EXCEPT เป็นการดำเนินการชนิดทวิภาคเกี่ยวกับความสัมพันธ์ ในที่นี้ สมมุติให้มีสองตารางหรือรีเลชัน (relation) ได้แก่ R และ S

(1) R UNION S ส่งผลให้ได้รีเลชันที่มีทูเพิล/รายการ (tuple/record) ที่อยู่ใน R หรือใน S รวมทั้งที่อยู่ในทั้งสองรีเลชัน

(2) R INTERSECT S ส่งผลให้ได้รีเลชันที่มีทูเพิล/รายการที่อยู่ในทั้ง R และ S

(3) R EXCEPT S ส่งผลให้ได้รีเลชันที่มีทูเพิล/รายการที่อยู่ใน R แต่ไม่อยู่ใน S

รูปที่ 1 แสดงตัวอย่างการดำเนินการชนิดทวิภาค (binary operations)



รูปที่ 1 ตัวอย่างของการดำเนินการชนิดทวิภาค

กลุ่มคำตอบสำหรับ C และ D

a)

111
222

b)

333
444
555
666

c)

333
555

d)

444
666

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำสั่ง SQL ต่อไปนี้

คำสั่ง SQL2 แสดงรายการนักศึกษาที่เรียนสำเร็จในรายวิชาทั้งหมดที่สอนโดยผู้สอน (instructor) ชื่อ Clark ในปี 2018 และ 2019

```
-- SQL2 --
SELECT StudentNo, Name, Department
FROM Student S
WHERE [ E ] (
        (SELECT SectionNo
         FROM Section
         WHERE Instructor = 'Clark'
           AND (Year = 2018 OR Year = 2019))
        [ F ]
        (SELECT SectionNo
         FROM GradeReport G
         WHERE G.StudentNo = S.StudentNo)  )
```

จากข้อมูลตัวอย่างของแต่ละตารางเดียวกันกับที่แสดงในคำอธิบาย คำสั่ง SQL2 จะแสดงผลลัพธ์ดังนี้:

StudentNo	Name	Department
1599222	Mathew Ken	CS

กลุ่มคำตอบสำหรับ E

a) EXISTS
b) IN
c) NOT EXISTS
d) NOT IN

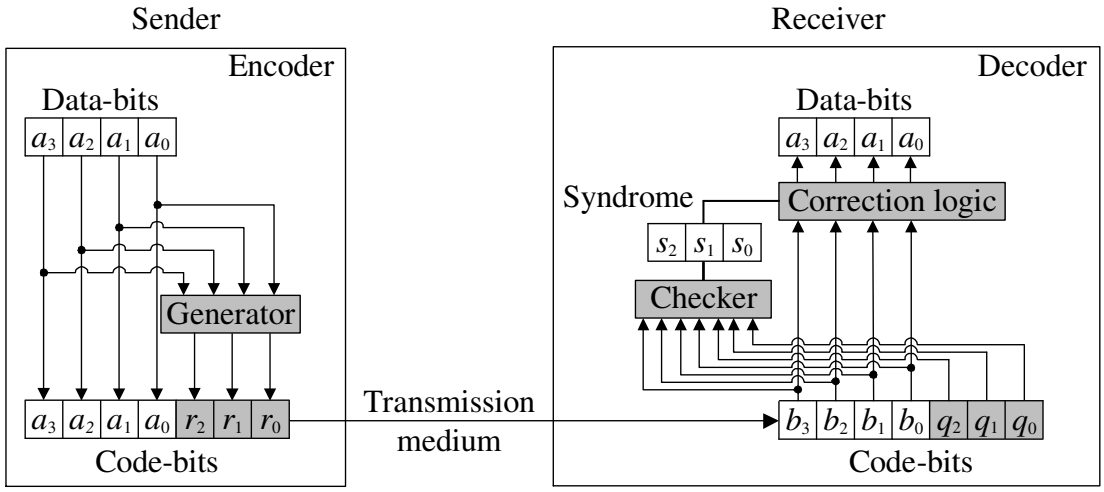
กลุ่มคำตอบสำหรับ F

a) EXCEPT
b) INTERSECT
c) UNION

Q4. อ่านคำอธิบายของแฮมมิงโค้ด (Hamming code) แล้วตอบคำถามย่อย 1 และ 2

แฮมมิงโค้ด (Hamming code) คือข้อมูลเพิ่มเติมที่จะถูกเพิ่งเข้าไปในข้อมูลเพื่อตรวจสอบ/แก้ไขข้อผิดพลาดที่อาจเกิดขึ้นระหว่างการส่งข้อมูล

ตัวอย่างเช่น แฮมมิงโค้ด C(7, 4) หมายความว่าขณะส่งบิตข้อมูล (data-bits) ขนาด 4 บิต ผู้ส่ง (Sender) จะคำนวณบิตพาริตีเพิ่มเติม 3 บิต เพื่อประกอบเข้ากับข้อมูลและแปลงให้เป็นบิตเข้ารหัส (code-bits) ขนาด 7 บิต และส่งบิตเข้ารหัสนี้ไปให้ผู้รับ (Receiver) โดยผู้รับจะคำนวณบิตพาริตี 3 บิตขึ้นมาจากบิตเข้ารหัสขนาด 7 บิตที่ได้รับมาและตรวจหาข้อผิดพลาด หากไม่เกิดความผิดพลาดในการส่งข้อมูล ผู้รับจะได้บิตข้อมูลที่ถูกต้องทั้ง 4 บิต แต่หากเกิดความผิดพลาดในบิตใด ค่าของบิตพาริตี 3 บิตจะบอกถึงตำแหน่งที่เกิดความผิดพลาด ซึ่งสามารถแก้ไขได้ทันที



รูปที่ 1 โครงสร้างของการเข้ารหัส (encoder) และถอดรหัส (decoder) แฮมมิงโค้ด

รูปที่ 1 แสดงโครงสร้างของการเข้ารหัส (encoder) ทางฝั่งผู้ส่ง และการถอดรหัส (decoder) ทางฝั่งผู้รับ สำหรับแฮมมิงโค้ด C(7, 4) ทางฝั่งผู้ส่ง บิตพาริตี 3 บิต r_2, r_1, และ r_0 ถูกสร้างขึ้น (generated) จากบิตข้อมูล 4 บิต a_3, a_2, a_1, และ a_0 และถูกต่อท้ายเข้ากับบิตข้อมูลก่อนการส่ง บิตพาริตี r_2, r_1 และ r_0 ถูกสร้างขึ้นด้วยสมการด้านล่างนี้:

$$r_2 = a_3 \oplus a_1 \oplus a_0$$
$$r_1 = a_3 \oplus a_2 \oplus a_1$$
$$r_0 = a_2 \oplus a_1 \oplus a_0$$

โดยสัญลักษณ์ “⊕” หมายถึงกระบวนการ XOR (exclusive-OR)

ทางฝั่งผู้รับ เมื่อได้รับบิตเข้ารหัส 7 บิต b_3, b_2, b_1, b_0, q_2, q_1 และ q_0 แล้ว ส่วนตรวจสอบ (checker) จะสร้างบิตซินโดรม (syndrome-bits) 3 บิต s_2, s_1 และ s_0 ด้วยสมการที่ให้ไว้ด้านล่างนี้ บิตซินโดรมจะช่วยบ่งบอกผลการรับข้อมูลและตำแหน่งของความผิดพลาดขนาดหนึ่งบิตได้

$$s_2 = b_3 \oplus b_1 \oplus b_0 \oplus q_2$$
$$s_1 = b_3 \oplus b_2 \oplus b_1 \oplus q_1$$
$$s_0 = b_2 \oplus b_1 \oplus b_0 \oplus q_0$$

รูปแบบของบิตซินโดรม 3 บิตจะบ่งบอกถึงตำแหน่งที่เกิดความผิดพลาดในบิตเข้ารหัสที่ได้รับ หากมีความผิดพลาดเกิดขึ้น

ตารางที่ 1 แสดงค่าของบิตซินโดรมกับจุดที่เกิดความผิดพลาด ตัวอย่างเช่น หากเกิดความผิดพลาดที่ q_0 จะมี s_0 เพียงบิตเดียวที่ได้รับผลกระทบ และบิตซินโดรมจะมีค่าเป็น 001

ตารางที่ 1 แสดงการจับคู่บิตซินโดรมกับตำแหน่งที่เกิดความผิดพลาด

บิตซินโดรม (s_2, s_1 , s_0)	ตำแหน่งที่ผิดพลาด
000	ไม่เกิดความผิดพลาด
001	q_0
...	...
111	b_1

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

บิตข้อมูล (data-bits) 1001 กลายเป็นบิตเข้ารหัส (code-bits) [A] เมื่อผู้รับได้รับบิตเข้ารหัสแล้ว ส่วนตรวจสอบ (checker) สร้างซินโดรมบิต (syndrome-bits) [B] และบอกได้ว่าบิตข้อมูลคือ 1001

เมื่อบิตข้อมูล 0110 กลายเป็นบิตเข้ารหัส 0110100 แต่ทางผู้รับกลับได้รับบิตเข้ารหัสเป็น 1110100 จากความผิดพลาดในการส่งข้อมูล ส่วนตรวจสอบจึงสร้างซินโดรมบิตขึ้นมาเป็น [C]

กลุ่มคำตอบสำหรับ A

a) 1001000
b) 1001011
c) 1001100
d) 1001110

กลุ่มคำตอบสำหรับ B และ C

a) 000
b) 010
c) 011
d) 100
e) 110
f) 111

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

เมื่อส่วนตรวจสอบทางฝั่งผู้รับสร้างซินโดรมบิตเป็น 110 ส่วนตรวจสอบพบว่าเกิดความผิดพลาดในบิต [D]

แฮมมิงโค้ด C(7, 4) ไม่สามารถแก้ไขข้อผิดพลาดขนาดสองบิตได้ สมมุติว่าผู้ส่งสร้างบิตเข้ารหัส 1101000 จากบิตข้อมูล 1101 และส่งให้กับผู้รับ แต่เกิดความผิดพลาดขนาดสองบิตขึ้นกับบิตเข้ารหัสระหว่างการส่งข้อมูล และผู้รับได้รับบิตเข้ารหัสเป็น 0001000 จากนั้น ส่วนตรวจสอบสร้างบิตซินโดรมขึ้นมาได้เป็น 101 และเข้าใจได้ว่าเกิดความผิดพลาดในบิต [E] และในที่สุด ผู้รับจะได้บิตข้อมูลที่ไม่ถูกต้องเป็น [F] หลังจากการแก้ไขข้อผิดพลาดแล้ว

กลุ่มคำตอบสำหรับ D และ E

a) b_0
b) b_1
c) b_2
d) b_3

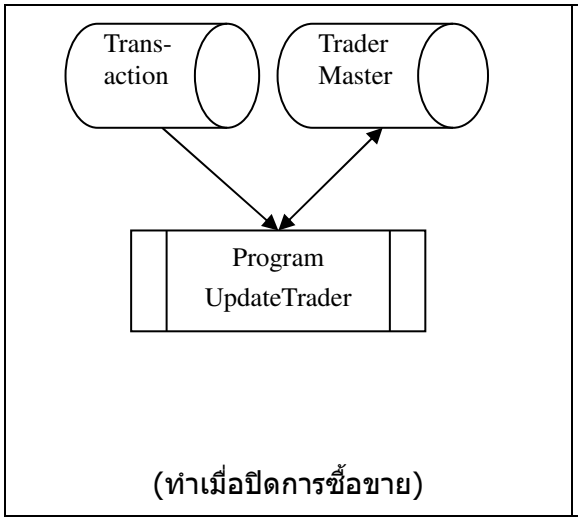
กลุ่มคำตอบสำหรับ F

a) 0000
b) 0011
c) 0101
d) 1001
e) 1101
f) 1110

Q5. อ่านคำอธิบายเกี่ยวกับการออกแบบซอฟต์แวร์แอปพลิเคชันเพื่อการซื้อขายหุ้น แล้วตอบคำถามย่อย 1 และ 2

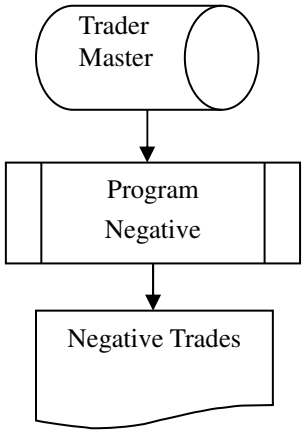
การซื้อขายหุ้น (stock trading) เกิดจากการที่ผู้ค้า (trader) ทำการซื้อและขายหุ้นรายการต่าง ๆ ตลอดทั้งวัน ผู้ค้าแต่ละรายมีหุ้นรายการต่าง ๆ ในจำนวนที่แตกต่างกัน และมีเงินสดสำหรับการซื้อหุ้น ระบบนี้อนุญาตให้ทำการขายชอร์ต (short selling) โดยผู้ค้าสามารถมีเงินสดหรือมีหุ้นในจำนวนที่ติดลบได้ระหว่างวัน แต่ควรกลายเป็นบวกเมื่อสิ้นสุดวัน ลักษณะเช่นนี้เกิดขึ้นเมื่อผู้ค้าขอยืมหุ้นเพื่อทำการขายหรือขอยืมเงินสดจากผู้ค้ารายอื่นในระหว่างวันและคืนให้เมื่อสิ้นสุดวัน

โปรแกรมหนึ่งถูกเขียนขึ้นเพื่ออัพเดตเรคคอร์ดของผู้ค้า (รูปที่ 1) และรายงานจำนวนหุ้นหรือเงินสดติดลบทั้งหมด (รูปที่ 2) เมื่อสิ้นสุดวัน



(ทำเมื่อปิดการซื้อขาย)

รูปที่ 1 อัพเดตเรคคอร์ดผู้ค้า (Update trader record)



(ทำเมื่อปิดการซื้อขาย)

รูปที่ 2 รายงานการค้าที่เป็นลบทั้งหมด (Report all negative trades)

[คำอธิบายโปรแกรมสำหรับ UpdateTrader]

หลังจากปิดการซื้อขายแล้ว โปรแกรมนี้จะอัพเดตไฟล์ TraderMaster ด้วยไฟล์ Transaction

(1) ไฟล์แบบเรียงลำดับ (sequential file) Transaction มีบันทึกการซื้อขายทั้งหมด (รายการหุ้นที่ถูกซื้อหรือถูกขาย) เรียงตามเวลาซื้อขายในวันนั้น ๆ

File: Transaction (TDate, TTime, TID, TItemID, TQty, TPrice, TBuyID, TSellID)

ฟิลด์	คำอธิบาย	ฟิลด์	คำอธิบาย
TDate	วันที่เกิด transaction	TQty	จำนวนที่ซื้อ/ขาย
TTime	เวลาเกิด transaction	TPrice	ราคาต่อหน่วยของหุ้น
TID	Unique transaction ID	TBuyID	Trader ID ของผู้ซื้อ
TItemID	ID ของหุ้นที่ถูกซื้อขาย	TSellID	Trader ID ของผู้ขาย

(2) ไฟล์ที่ทำดัชนีไว้ (indexed file) TraderMaster (ผู้ซื้อและผู้ขาย) จัดเก็บหุ้นต่าง ๆ ที่ถืออยู่โดยผู้ค้า (trader) ไฟล์นี้ทำดัชนีไว้ด้วย MID และ MItemID แต่ละเรคคอร์ดจัดเก็บหุ้นแต่ละรายการที่ผู้ค้ารายหนึ่งถือ (หรือเป็นเจ้าของ) อยู่ ในที่นี้ เงินสด (Cash) ถือเป็นรายการหนึ่ง และไฟล์นี้จะถูกอัพเดตหลังจากปิดการซื้อขาย

File: TraderMaster (MID, MItemID, MQty, MPrice)

ฟิลด์	คำอธิบาย
MID	Unique ID ของผู้ค้า (trader)
MItemID	Unique ID ของรายการหุ้น (stock item) ในรายการที่เป็นเงินสด ค่านี้จะเป็น "000"
MQty	จำนวนของหุ้นที่ถืออยู่โดยผู้ค้า ในรายการที่เป็นเงินสด ค่านี้จะเป็น 1
MPrice	ราคาต่อหน่วยของหุ้น (ราคาล่าสุด) ในรายการที่เป็นเงินสด ค่านี้แสดงยอดเงินคงเหลือ (balance)

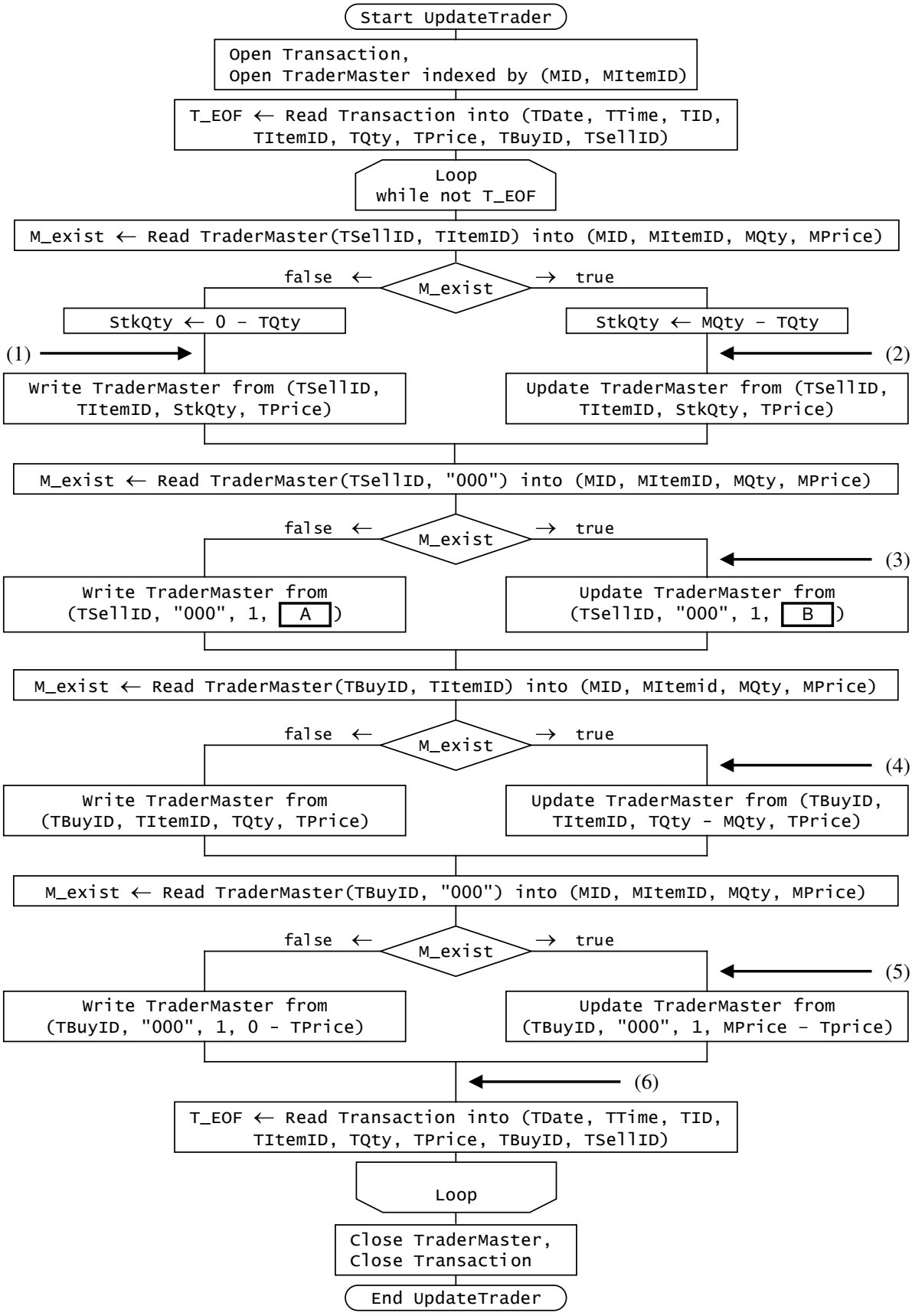
[คำอธิบายโปรแกรมสำหรับ Negative]

(1) โปรแกรมนี้แสดงผู้ค้าทั้งหมดที่มีสถานะเงินสดติดลบหรือมีจำนวนหุ้นติดลบ ณ เวลาปิดการซื้อขาย โปรแกรมนี้จะถูกประมวลผลหลังจาก UpdateTrader

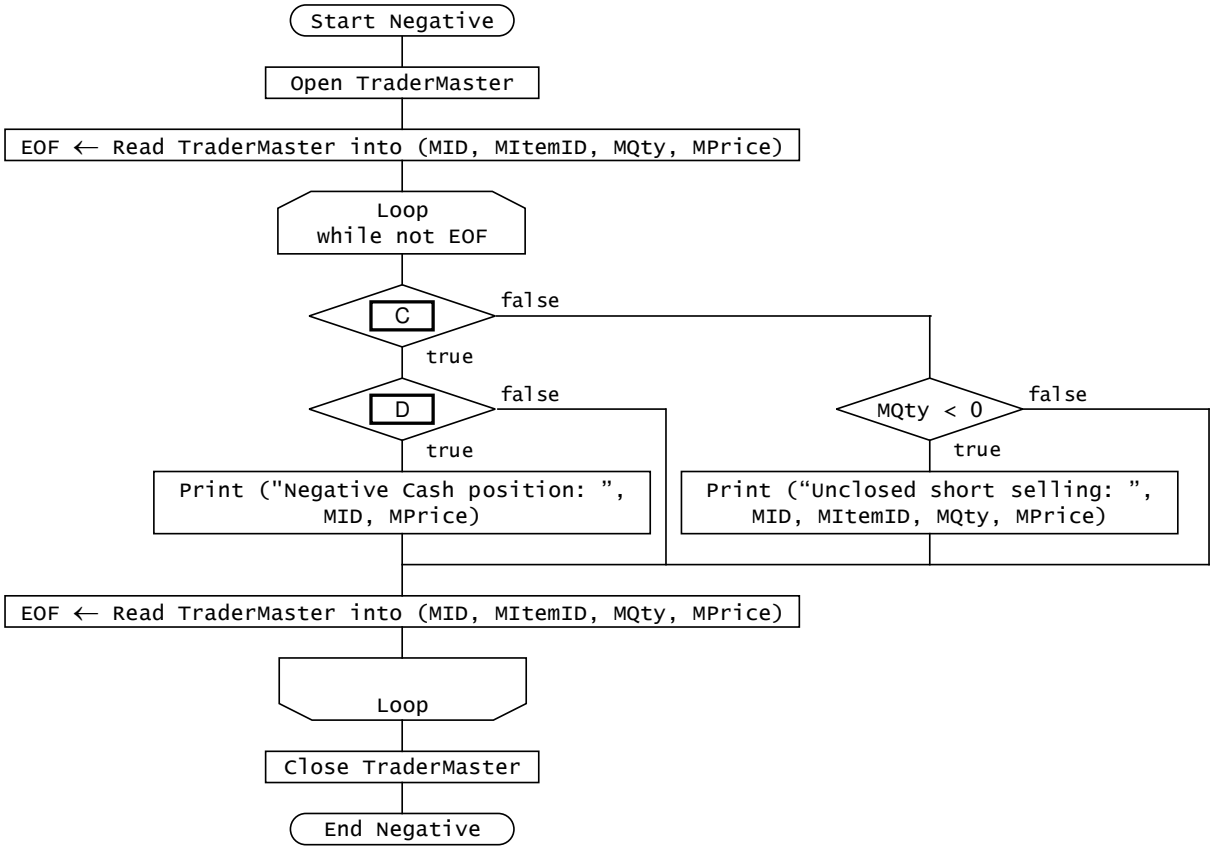
ในโปรแกรม UpdateTrader และ Negative มีคำสั่งที่ใช้ดังต่อไปนี้:

- Open *filename* [indexed by (*index1*, *index2*, …)]
  Open filename เปิด filename เพื่ออ่านแบบเรียงลำดับ และ Open filename indexed by (index1, index2, …) เปิด filename เพื่อการอ่านแบบมีดัชนีโดยใช้ฟิลด์ที่ระบุเป็นดัชนี
- Read *filename* into (*field1*, *field2*, …)
  อ่านเรคคอร์ดหนึ่งขึ้นมาจาก filename โดยฟิลด์ต่าง ๆ ในเรคคอร์ดถูกเก็บในตัวแปร field1, field2, … เมื่ออ่านถึงจุดสิ้นสุดไฟล์ คำสั่งนี้จะคืนค่าเป็น true
- Read *filename* (*index1*, *index2*, …) into (*field1*, *field2*, …)
  ไฟล์ filename จะถูกค้นหาเพื่อให้ได้เรคคอร์ดที่มีดัชนีตรงกับ index1, index2, … ฟิลด์ต่าง ๆ ในเรคคอร์ดจะถูกเก็บในตัวแปร field1, field2, … และคำสั่งนี้คืนค่าเป็น true หากพบเรคคอร์ดที่ระบุ
- Write *filename* from (*value1*, *value2*, …)
  เขียนหนึ่งเรคคอร์ดลงใน filename โดยฟิลด์ในเรคคอร์ดถูกสร้างขึ้นจากรายการ values value1, value2, …
- Update *filename* from (*value1*, *value2*, …)
  อัพเดตเรคคอร์ดที่ระบุใน filename โดยฟิลด์ในเรคคอร์ดถูกสร้างขึ้นจากรายการ values value1, value2, …
- Close *filename*
  ปิด filename

[โปรแกรม UpdateTrader]



[โปรแกรม Negative]



คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในตัวโปรแกรม

กลุ่มคำตอบสำหรับ A และ B

a) `- MPrice`
b) `- TPrice`
c) `MPrice`
d) `TPrice`
e) `MPrice + TPrice`
f) `MPrice - TPrice`

กลุ่มคำตอบสำหรับ C และ D

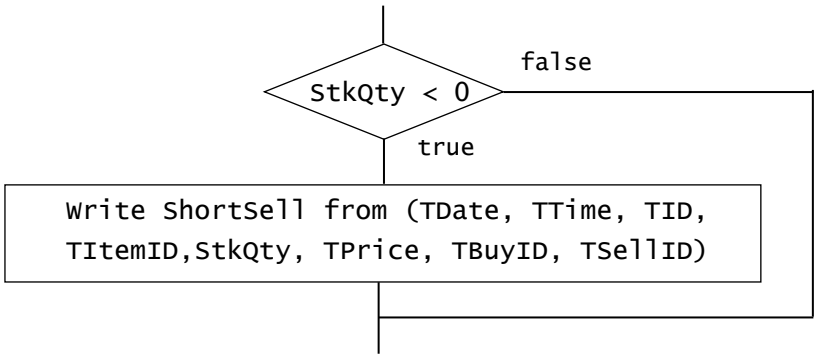
a) `MItemID = "000"`
b) `MPrice < 0`
c) `MPrice ≥ 0`
d) `MQty < 0`
e) `MQty ≥ 0`

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

เพื่อดูแลการขายชอร์ต (short selling) ไฟล์เรียงลำดับ ShortSell ถูกใช้ในโปรแกรม UpdateTrader โดยไฟล์ ShortSell จะเก็บรายการทั้งหมดที่เป็นการขายชอร์ตหรือที่จำนวนการขายมีค่าเป็นลบ ฟิลด์ของไฟล์ ShortSell นี้เหมือนกับของไฟล์ Transaction เว้นแต่ TQty จะเก็บค่าติดลบสำหรับจำนวน (quantity) ในอีกไฟล์หนึ่ง

เพื่อดูแลการขายชอร์ต กระบวนการต่อไปนี้ควรถูกแทรกลงในจุดที่ชี้อยู่ด้วย [E] ในที่นี้ ไฟล์ ShortSell ถูกเปิดและปิดไว้อย่างถูกต้องแล้วในตัวโปรแกรม



กลุ่มคำตอบสำหรับ E

a) (1) b) (2) c) (3)
d) (4) e) (5) f) (6)

Q6. อ่านคำอธิบายของโปรแกรมเพื่อแก้ไขโจทย์ปัญหาผลรวมของอาร์เรย์ย่อย จากนั้นตอบคำถามย่อย 1 ถึง 3 (สามารถดูสัญลักษณ์ที่ใช้ในภาษาเทียม ในด้านหน้าของสมุดคำถามนี้)

สมมติมีอาร์เรย์จำนวนเต็มบวกหนึ่งตัวและค่าผลรวมที่ต้องการ (target sum) โจทย์ปัญหาก็คือการค้นหาอาร์เรย์ย่อย (subarray) ทั้งหมดที่เป็นไปได้ที่มีผลรวมเท่ากับค่าผลรวมที่ต้องการ ในที่นี้อาร์เรย์ย่อย (subarray) ก็คือบางส่วนของอาร์เรย์ที่กำหนดให้และต้องเป็นเอลิเมนต์ที่ต้องต่อเนื่องกัน ตัวอย่างเช่น สมมติกำหนดผลรวมที่ต้องการเป็น 14 อาร์เรย์ย่อยที่ถูกแรเงาในรูปที่ 1 นั้นมีคุณสมบัติตรงตามเงื่อนไขที่กำหนด (คือมีผลรวมเป็น 14) อย่างไรก็ตามการใช้พอยน์เตอร์ 2 ตัว (2-point method) ถือว่าเป็นวิธีการที่มีประสิทธิภาพในการแก้ปัญหาแบบนี้มากกว่าวิธีการแบบดั้งเดิม

6	3	2	5	1	1	7	3

รูปที่ 1 ตัวอย่างอาร์เรย์

[คำอธิบายโปรแกรม]

(1) โปรแกรมย่อย (Subprogram) SSumNaive จะตรวจสอบว่าผลรวมย่อยที่ทั้งหมดเป็นไปได้ ว่ามีค่าเท่ากับผลรวมที่ต้องการหรือไม่ ถ้าหากพบอาร์เรย์ที่ตรงตามเงื่อนไขนี้ โปรแกรมก็จะแสดงหมายเลขดัชนีเริ่มต้นและหมายเลขดัชนีสิ้นสุด มิฉะนั้นก็จะแสดงข้อความว่าไม่พบอาร์เรย์ย่อย "No subarray found"

(2) โปรแกรมย่อย SSum2point จะใช้พอยน์เตอร์ (Pointer) จำนวน 2 ตัว คือ ดัชนีเริ่มต้นและดัชนีสุดท้ายของอาร์เรย์ย่อย โดยตอนเริ่มต้นจะกำหนดให้ดัชนีเริ่มต้นและดัชนีสุดท้ายมีค่าเป็น 1 ทั้งคู่ และผลรวมปัจจุบันของอาร์เรย์ย่อย (current sum) คือ เอลิเมนต์ตัวแรกของอาร์เรย์

   (i) ถ้าหากผลรวมปัจจุบันของอาร์เรย์ย่อยมีค่ามากกว่าผลรวมเป้าหมาย และดัชนีเริ่มต้นมีจำนวนน้อยกว่าดัชนีตัวสุดท้ายแล้ว ก็ให้ลบเอลิเมนต์ตัวแรกออกจากผลรวมปัจจุบันและขยับดัชนีเริ่มต้นไปทางขวาเรื่อย ๆ จนกว่าจะมีค่าน้อยกว่าหรือเท่ากับผลรวมเป้าหมาย

   (ii) ถ้าหากผลรวมปัจจุบันเท่ากับผลรวมเป้าหมาย ก็จะแสดงหมายเลขของพอยน์เตอร์ทั้งสองตัว

   (iii) เอลิเมนต์ตัวถัดไปของอาร์เรย์ย่อยจะถูกบวกเข้าไปในผลรวมปัจจุบัน และขยับดัชนีตัวสุดท้ายไปทางขวาอีก 1 ตัว ซึ่งโปรแกรมย่อยจะหยุดการทำงาน เมื่อดัชนีตัวสุดท้ายนั้นมีค่ามากกว่าดัชนีสุดท้ายของอาร์เรย์ มิฉะนั้นก็จะกลับไปทำซ้ำขั้นตอนที่ (i)

   (iv) ถ้าหากไม่สามารถหาผลรวมเป้าหมายภายในอาร์เรย์ย่อยได้ โปรแกรมก็จะแสดงข้อความไม่พบอาร์เรย์ย่อย "No subarray found"

(3) ดัชนีของอาร์เรย์เริ่มต้นที่ 1

(4) โปรแกรมย่อย print( ) แสดงอินพุตพารามิเตอร์ (input parameter) ในอีกบรรทัด

(5) ตารางที่ 1 อธิบายตัวแปรที่ถูกใช้ในโปรแกรม

ตารางที่ 1 อธิบายตัวแปรที่ถูกใช้ในโปรแกรม

ตัวแปร	คำอธิบาย
N	ขนาดของอินพุตอาร์เรย์ เมื่อ N มีค่าไม่น้อยกว่า 1
S	ผลรวมที่ต้องการ
A[]	อินพุตอาร์เรย์ ซึ่งค่าตัวเลขที่แสดงในรูปที่ 1 นั้นถูกกำหนดเป็นเอลิเมนต์อาร์เรย์
Start	หมายเลขดัชนีเริ่มต้นของอาร์เรย์ย่อย
End	หมายเลขดัชนีตัวสุดท้ายของอาร์เรย์ย่อย
Sum	ผลรวมปัจจุบันของอาร์เรย์ย่อย

[โปรแกรม]

```
GLOBAL: INT: N ← 8, S ← 14
GLOBAL: INT: A[N] ← {6, 3, 2, 5, 1, 1, 7, 3}

SUBPROGRAM: SSumNaive() {
   INT: sum, start, end, found ← 0
   FOR (start ← 1; start ≤ N; start ← start + 1) {
      sum ← 0;
      FOR (end ←     A     ; sum ≤ S and end ≤ N; end ← end + 1) {
         sum ← sum + A[end];    /* α */
         IF (S = sum) {
            print(start + ", " + end);
            found ← 1;
         }
      }
   }
   IF (     B     ) {
      print("No subarray found");
   }
}
```

```
SUBPROGRAM: SSum2point() {
   INT: sum ← [ C ], start ← 1, end, found ← 0
   FOR (end ← 1; end ≤ N; end ← end + 1) {
      WHILE (S < sum and start < end) {
         sum ← sum - A[start];    /* β */
         start ← start + 1;
      }
      IF (S = sum) {
         print(start + ", " + end);
         found ← 1;
      }
      IF (end < N) {
         sum ← sum + A[ D ];
      }
   }
   IF ( B ) {
      print("No subarray found");
   }
}
```

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องที่จะเติมลงในแต่ละช่องว่าง [ ] แต่ละช่องในโปรแกรมด้านบน

กลุ่มคำตอบสำหรับ A และ D

a) 0
b) 1
c) end
d) end + 1
e) start
f) start + 1

กลุ่มคำตอบสำหรับ B

a) found = 0
b) found = 1
c) found = S
d) found > 1

กลุ่มคำตอบสำหรับ C

a) 0
b) A[1]
c) A[1] + A[2]
d) S

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องที่จะเติมลงในแต่ละช่องว่าง [ ] แต่ละช่องในคำบรรยายต่อไปนี้

เมื่อมีการเรียกโปรแกรมย่อย SSumNaive บรรทัด /* α */ จะถูกประมวลผล [E] ครั้ง
เมื่อมีการเรียกโปรแกรมย่อย SSum2point บรรทัด /* β */ จะถูกประมวลผล [F] ครั้ง

กลุ่มคำตอบสำหรับ E

a) 8
b) 28
c) 29
d) 30
e) 36
f) 56

กลุ่มคำตอบสำหรับ F

a) 1
b) 2
c) 3
d) 4
e) 5
f) 6
g) 7
h) 8

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่างนี้ ให้เลือกคู่คำตอบที่ถูกต้องของผลลัพธ์ในการประมวลผลของโปรแกรมย่อย SSumNaive และ SSum2point เมื่ออาร์เรย์ A [] มีค่าศูนย์หรือมีค่าลบ ในที่นี้กำหนดให้สัญลักษณ์ "O" หมายถึง ผลลัพธ์ในการประมวลผลถูกต้องในทุกกรณี และสัญลักษณ์ "X" หมายถึง ผลลัพธ์ในการประมวลผลอาจไม่ถูกต้องในบางกรณี

กลุ่มของคำตอบ

	โปรแกรมย่อย SSumNaive		โปรแกรมย่อย SSum2point	
	A[] มีค่า 0	A[] มีค่าติดลบ	A[] มีค่า 0	A[] มีค่าติดลบ
a)	O	O	O	O
b)	O	O	O	X
c)	O	O	X	X
d)	O	X	O	X
e)	O	X	X	X
f)	X	X	X	X

สำหรับข้อสอบข้อที่ Q7 และ Q8 ให้เลือกทำเพียงหนึ่งในสองข้อ
จากนั้น ให้ระบายทับในวงกลม Ⓢ ในกระดาษคำตอบสำหรับข้อที่เลือกทำ
หากเลือกไว้ทั้งสองข้อ จะตรวจให้คะแนนเฉพาะข้อแรกเท่านั้น

Q7. ให้อ่านรายละเอียดโปรแกรมภาษา C และตัวโปรแกรมต่อไปนี้ แล้วตอบคำถามย่อย 1 ถึง 3

ในทางคณิตศาสตร์ จำนวนสมบูรณ์ (perfect number) คือ จำนวนเต็มที่มีผลบวกของตัวหารต่าง ๆ เท่ากับสองเท่าของตัวมันเอง ตัวอย่างเช่น 6 เป็นจำนวนสมบูรณ์ เนื่องจากผลรวมของตัวหารมีค่าเป็นสองเท่าของ 6 (ตัวหารแท้ของ 6 คือ 1, 2, 3 และ 6) ซึ่งก็คือ 12 นั่นเอง

[รายละเอียดโปรแกรม]
โปรแกรมจะแสดงจำนวนสมบูรณ์ (perfect number) ระหว่าง 1 ถึง 100 ออกไปยังเอาท์พุตมาตรฐาน โดยฟังก์ชั่น divSum ทำหน้าที่คำนวณผลรวมของตัวหาร สำหรับตัวแปร integer num

[Program]

```
#include <stdio.h>

int divSum(int num) {
   int result = 0;
   int i;
   for (i = 1; i * i <= num; i++) {
      if (num % i == 0) {
         result += [    A    ];
         if (i != (num / i)) {
            result += [    B    ];
         }
      }
   }
   return result;
}

/* α */
int main() {
   int i;    /* β */
   printf("Perfect numbers that lie between 1 and 100:");
   for (i = 1; i <= 100; i++) {
      if (divSum(i) == 2 * i) {
         printf(" %d", i);
      }
   }
   printf("\n");
   /* γ */
   return 0;
}
```

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง _____ แต่ละช่องในโปรแกรมด้านบน

กลุ่มคำตอบสำหรับ A และ B

a) `i`
b) `num`
c) `num % i`
d) `num / i`
e) `result + i`
f) `result + i % num`
g) `result + i / num`

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง _____ แต่ละช่องในคำอธิบายต่อไปนี้

แนวคิดของจำนวนสมบูรณ์ สามารถถูกขยายเป็น จำนวนสมบูรณ์ (m, k)-perfect number
จำนวนเต็ม num จะเป็น (m, k)-perfect number เมื่อความสมมาตรของประพจน์ที่กำหนดให้ต่อไปนี้เป็นจริง โดยที่ m และ k เป็นจำนวนเต็มบวก

$$\underbrace{\texttt{divSum(divSum(...divSum(num)...))}}_{\text{Applied m times}} = \texttt{k * num}$$

ตัวอย่างเช่น 16 เป็นจำนวนสมบูรณ์ (2, 2)-perfect number เนื่องจาก divSum(16) = 1 + 2 + 4 + 8 + 16 = 31 และ divSum(31) = 1 + 31 = 32
ดังนั้นแล้ว ประพจน์ divSum(divSum(16)) = 2 * 16 จึงมีค่าเป็นจริง

เพื่อให้สามารถแสดงผล (m, k)-perfect numbers ได้เช่นเดียวกับค่าสมบูรณ์อื่น ๆ โปรแกรมด้านบนจึงได้รับการแก้ไขดังนี้

(1) แทนที่บรรทัดที่มีเครื่องหมาย /* α */ ด้วยฟังก์ชั่นใหม่ divSumM
จากนั้นเรียกใช้ divSum m ครั้งเพื่อให้ได้ จำนวนเต็มบวก num

```
int divSumM(int m, int num) {
   if (m > 1) {
      return    C    ;
   } else {
      return divSum(num);  /* δ */ }
}
```

(2) แทนที่บรรทัดที่มีเครื่องหมาย /* β */ ด้วยคำสั่งต่อไปนี้ เพื่อเพิ่มการประกาศตัวแปร m และ k

```
int i, m, k;
```

(3) แทนที่บรรทัดที่มีเครื่องหมาย /* γ */ ด้วยคำสั่งต่อไปนี้ เพื่อทำหน้าที่คำนวณจำนวนสมบูรณ์ (m, k)-perfect numbers

```
printf("Enter two positive integers: ");
scanf("%d %d", D );
printf("(%d, %d)-perfect numbers that lie between 1 and 100:",
       m, k);
for (i = 1; i <= 100; i++) {
   if (divSumM(m, i) == k * i) {
      printf(" %d", i);
   }
}
printf("\n");
```

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง _____ แต่ละช่องในคำอธิบายด้านบน

กลุ่มคำตอบสำหรับ C

a) `divSum(m)`
b) `divSum(m - 1)`
c) `divSum(num)`
d) `divSum(num - 1)`
e) `divSumM(m - 1, divSum(num))`
f) `divSumM(m - 1, divSum(num - 1))`
g) `divSumM(m - 1, num)`
h) `divSumM(m - 1, num - 1)`

กลุ่มคำตอบสำหรับ D

a) `m, k`
b) `*m, *k`
c) `*&m, *&k`
d) `&m, &k`
e) `&*m, &*k`

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [_____] แต่ละช่องในคำอธิบายต่อไปนี้

เมื่อทำการประมวลผลโปรแกรมที่ถูกแก้ไขแล้ว โปรแกรมจะแสดงผลดังต่อไปนี้

```
Perfect numbers that lie between 1 and 100: 6 28
Enter two positive integers: 3 6
(3, 6)-perfect numbers that lie between 1 and 100: 98
```

ในการประมวลผล บรรทัดที่มีเครื่องหมาย /* δ */ ถูกประมวลผล [___E___] ครั้ง

กลุ่มคำตอบ

a) 100
b) 200
c) 300
d) 400
e) 600
f) 700

Q8. อ่านคำอธิบายโปรแกรมภาษาจาวาและตัวโปรแกรม แล้วตอบคำถามย่อย 1 และ 2

โปรแกรมภาษาจาวา แสดงโครงสร้างของระบบไฟล์ในแบบต้นไม้ในลักษณะ POSIX-style ซึ่งจะเป็นโหนด โดยแต่ละโหนด จะหมายถึงไฟล์หรือไดเรกทอรี ซึ่งไดเรกทอรีสามารถมีไฟล์หรือไดเรกทอรีอื่น ๆ ได้อีก นอกจากนี้ รูทโหนด (root node) คือโหนดพิเศษที่แสดงถึงโหนดที่อยู่บนสุดของไดเรกทอรี ไฟล์และไดเรกทอรีทั้งหมดจะอยู่ภายใต้รูทโหนด นอกจากนี้โปรแกรมสามารถค้นหาโหนดได้ตามเงื่อนไขที่กำหนด

(1) คลาส Node จะหมายถึงโหนดทั้งที่เป็นไฟล์และไดเรกทอรี โดยคลาสนี้จะประกอบด้วยแอทริบิวท์ ได้แก่
   (i) name: ชื่อของ Node นี้ เช่น "MyLog.log" โดยที่ชื่อจะต้องไม่มีเครื่องหมาย /
   (ii) extension: ถ้า Node หมายถึงไฟล์แล้ว จะต้องมีนามสกุลของไฟล์ ขึ้นต้นด้วย '.' (จุด) และตามด้วยชนิดของไฟล์ เช่น ".txt" สำหรับไฟล์ข้อความ ถ้าในชื่อไฟล์มี จุด มากกว่าหนึ่งจุด จุดตัวสุดท้ายจะหมายถึงนามสกุลของไฟล์ ถ้า Node หมายถึงไดเรกทอรีแล้ว แอทริบิวท์ จะมีค่าเป็น null
   (iii) fullPath: หมายถึง absolute path ของ Node นี้ ชื่อโหนดจะคั่นด้วย '/' เช่น "/home/user1/Desktop/note.txt"
   (iv) parent: หมายถึง parent directory ของ Node นี้ โดยที่ทุกโหนดจะต้องมี parents ยกเว้นรูทโหนด
   (v) children: หมายถึงรายการโหนดที่อยู่ภายใน Node นี้ โดยที่ชื่อของโหนดที่อยู่ภายในไดเรกทอรีเดียวกันจะต้องไม่ซ้ำกัน ถ้าโหนดคือไฟล์แล้ว ค่าแอทริบิวท์จะเป็น null
   (vi) คลาส ROOT (static) หมายถึงรูทโหนด
(2) อินเตอร์เฟส ICondition คือเงื่อนไขของการค้นหาโหนด ในขณะที่เมธอด isSatisfied จะคืนค่า true ถ้า Node ที่กำหนดตรงตามเงื่อนไข มิฉะนั้นจะคืนค่า false
(3) คลาส NameCondition จะอิมพลีเมนต์ อินเตอร์เฟส ICondition
   (i) แอทริบิวท์ name คือชื่อโหนดที่ผู้เรียก (Caller) ต้องการค้นหา
   (ii) เมธอด isSatisfied จะคืนค่า true ถ้าชื่อของ Node ที่กำหนดมีค่าเท่ากับ แอทริบิวท์ name มิฉะนั้นจะคืนค่า false
(4) คลาส Search จะประกอบด้วย static method ชื่อ searchByName ซึ่งจะค้นหาโหนดทุกโหนดภายในโหนดที่กำหนด โดยจะหาชื่อของโหนดที่มีค่าเท่ากับพารามิเตอร์ name
(5) กำหนดให้ว่าทุกคอนสตรัคเตอร์และเมธอดถูกเรียกใช้งานด้วยพารามิเตอร์ที่ถูกต้อง

ต่อไปนี้คือเมธอดที่เกี่ยวข้องกับสตริง (string) ของคลาส String ที่จะใช้ในคำถามข้อนี้

(1) String substring(int beginIndex)
   จะคืนค่าสตริงที่เป็นสตริงย่อย (substring) โดยสตริงย่อยจะขึ้นต้นด้วยอักขระที่กำหนดไว้ใน beginIndex จนถึงอักขระสุดท้ายของสตริง
(2) int indexOf(int ch)
   จะคืนค่าดัชนี (index) ของอักขระตัวแรกที่พบตามที่กำหนดไว้ใน ch หรือ จะคืนค่า -1 ถ้าอักขระดังกล่าวไม่ปรากฏ

(3) int lastIndexOf(int ch)

จะคืนค่าดัชนี (index) ของอักขระตัวสุดท้ายที่พบตามที่กำหนดไว้ใน ch หรือ จะคืนค่า -1 ถ้าอักขระดังกล่าวไม่ปรากฏ

ผลลัพธ์ต่อไปนี้ถูกสร้างขึ้นเมื่อเมธอด main ของคลาส Search ถูกเรียกใช้งาน

```
Some nodes info:
 "/"
 "/var/log/program.home.log" .log
 "/home/user1/document/note.txt" .txt
Result of searching nodes:
 ["/home/user1/document/note.txt" .txt]
```

[โปรแกรม 1]

```
import java.util.ArrayList;
import java.util.List;

public class Node {
   public static final Node ROOT = new Node("", null, true);

   private final String name;
   private final Node parent;
   private final String extension;
   private final List<Node> children;
   private final String fullPath;

   private Node(String name, Node parent, boolean directory) {
      this.name = name;
      this.parent = parent;
      if (directory) {
         extension = null;
         children = new ArrayList<>();
      } else {
         extension = name.substring(    A    );
         children = null;
      }
      if (parent == null || parent == ROOT) {
         fullPath = "/" + name;
      } else {
         fullPath =    B    + "/" + name;
      }
   }

   public static Node create(String name, Node parent,
                             boolean directory) {
      Node node = new Node(name, parent, directory);
      parent.children.add(node);
      return node;
   }
```

```
    @Override
    public String toString() {
       return "\"" + fullPath + "\""
              + (extension == null ? "" : " " + extension);
    }

    public String getName() { return name; }

    public String getExtension() { return extension; }

    public String getFullPath() { return fullPath; }

    public Node getParent() { return parent; }

    public boolean isDirectory() { return children != null; }

    public List<Node> getChildren() {
       return isDirectory() ? new ArrayList<>(children) : null;
    }
  }
```

[โปรแกรม 2]

```
  public interface ICondition {
     boolean isSatisfied(Node node);
  }
```

[โปรแกรม 3]

```
  public class NameCondition implements ICondition {
     private final String name;

     public NameCondition(String name) {
        this.name = name;
     }

     public String getName() { return name; }

     @Override
     public boolean isSatisfied(Node node) {
         return node.getName().equals(getName());
     }
  }
```

[โปรแกรม 4]

```
import java.util.ArrayList;
import java.util.List;

public class Search {
   private static List<Node> searchList(Node root,
                                        ICondition condition) {
      List<Node> result = new ArrayList<>();
      doSearchList(root, condition, result);
      return result;
   }

   private static void doSearchList(Node current, ICondition
condition,
                                    List<Node> result) {
      if (condition.isSatisfied(current)) {
         result.add(current);
      }
      if (    C    ) {
         for (Node child : current.getChildren()) {
            doSearchList(child, condition, result);
         }
      }
   }

   public static List<Node> searchByName(Node root, String name) {
      ICondition condition =    D    ;
      return    E    ;
   }

   public static void main(String[] args) {
      Node home = Node.create("home", Node.ROOT, true);
      Node user1 = Node.create("user1", home, true);
      Node document = Node.create("document", user1, true);
      Node note = Node.create("note.txt", document, false);
      Node var = Node.create("var", Node.ROOT, true);
      Node tmp = Node.create("tmp", var, true);
      Node log = Node.create("log", var, true);
      Node logfile = Node.create("program.home.log", log, false);

      System.out.printf("Some nodes info:%n %s%n %s%n %s%n%n",
            Node.ROOT, logfile, note);
      System.out.printf("Result of searching nodes:%n %s%n",
            searchByName(Node.ROOT, "note.txt"));
   }
}
```

คำถามย่อย 1

จากกลุ่มคำตอบที่กำหนดให้ต่อไปนี้ เลือกคำตอบที่ถูกต้องที่จะเติมลงในช่องว่าง ☐ แต่ละช่องในโปรแกรม 1 และ โปรแกรม 4

กลุ่มคำตอบสำหรับ A

a) `name.indexOf('.')`
b) `name.indexOf('.') + 1`
c) `name.indexOf('.') - 1`
d) `name.lastIndexOf('.')`
e) `name.lastIndexOf('.') + 1`
f) `name.lastIndexOf('.') - 1`

กลุ่มคำตอบสำหรับ B

a) `"/" + parent.fullPath`
b) `"/" + parent.name`
c) `fullPath`
d) `name`
e) `parent.fullPath`
f) `parent.name`

กลุ่มคำตอบสำหรับ C

a) `current == Node.ROOT`
b) `current.getChildren() == null`
c) `current.getChildren().size() > 0`
d) `current.isDirectory()`
e) `result == null`
f) `result.size() == 0`

กลุ่มคำตอบสำหรับ D

a) `ICondition(name)`
b) `NameCondition(name)`
c) `new ICondition(name)`
d) `new NameCondition(name)`

กลุ่มคำตอบสำหรับ E

a) `doSearchList(root, condition, new ArrayList<>())`
b) `doSearchList(root.getChildren(), condition, new ArrayList<>())`
c) `doSearchList(root.getParent(), condition, new ArrayList<>())`
d) `searchList(root, condition)`
e) `searchList(root.getChildren(), condition)`
f) `searchList(root.getParent(), condition)`

คำถามย่อย 2

จากกลุ่มคำตอบที่กำหนดให้ต่อไปนี้ เลือกคำตอบที่ถูกต้องที่จะเติมลงในช่องว่าง [ ] แต่ละช่องในโปรแกรม 5

โดยทั่วไป ผู้ใช้อาจจะต้องการค้นหาไฟล์หรือไดเรกทอรีโดยที่ชื่อบางส่วนนั้นจะตรงกับสตริงที่กำหนดมากกว่าจะหาด้วยชื่อเต็ม ยกตัวอย่างเช่น เมื่อต้องการหา "home" ด้วยอินสแตนซ์ Node ที่ถูกสร้างจากโปรแกรม 4 จะมีเพียงสองโหนดเท่านั้น ได้แก่ ไฟล์ที่ชื่อ program.home.log และ ไดเรกทอรี ที่ชื่อ home เพราะว่าทั้งสองโหนดมีสตริง "home" ทั้งนี้ โปรแกรม 5 จะอิมพลีเมนต์คลาสใหม่ที่ชื่อ PartialNameCondition สำหรับใช้ในการหาบางส่วนของชื่อ

[โปรแกรม 5]

```
public class PartialNameCondition extends NameCondition {
   public PartialNameCondition(String name) {
      [ F ];
   }

   @Override
   public boolean isSatisfied(Node node) {
      return [ G ];
   }
}
```

กลุ่มคำตอบสำหรับ F

a) `return`
b) `super()`
c) `super(name)`
d) `this(name)`
e) `this.name = name`

กลุ่มคำตอบสำหรับ G

a) `getFullPath().contains(node.getFullPath())`
b) `getName().contains(node.getName())`
c) `node.getFullPath().contains(getFullPath())`
d) `node.getFullPath().contains(getName())`
e) `node.getName().contains(getFullPath())`
f) `node.getName().contains(getName())`