October 2023

Fundamental IT Engineer Examination (Afternoon)

ให้ทำข้อสอบตามรายละเอียดต่อไปนี้

หมายเลขคำถาม	Q1	Q2 – Q5	Q6	Q7, Q8
การเลือกคำถาม	คำถามบังคับ	เลือก 2 ใน 4	คำถามบังคับ	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 Q2 ถึง Q5 และ Q7 กับ Q8 ให้เลือก Ⓢ ของข้อที่คุณเลือกที่จะตอบในช่อง "Selection Column" บนกระดาษคำตอบของคุณ

(4) **คำตอบ**
ให้ทำเครื่องหมายตรงคำตอบที่เลือกตามตัวอย่างที่แสดงอยู่ด้านล่าง

[คำถามตัวอย่าง]
ข้อใดต่อไปนี้เป็นสิ่งที่ควรใช้ทำเครื่องหมายเพื่อเลือกข้อที่ต้องการในกระดาษคำตอบ

กลุ่มคำตอบ
a) ปากกาลูกลื่น b) สีเทียน c) ปากกาหมึกซึม d)ดินสอ

เนื่องจากคำตอบที่ถูกคือ "d)" (ดินสอ), ดังนั้นให้ทำเครื่องหมายดังแสดงด้านล่างนี้:

[ตัวอย่างคำตอบ]

Sample	ⓐ	ⓑ	ⓒ	●	ⓔ	ⓕ	ⓖ	ⓗ	ⓘ	ⓙ

ห้ามเปิดดูข้อสอบก่อนได้รับอนุญาต
ข้อสงสัยที่เกี่ยวข้องกับคำถามในข้อสอบอาจจะไม่ถูกตอบ

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 */ or // 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** เป็น **คำถามบังคับ**

Q1. อ่านคำอธิบายกระบวนการแลกเปลี่ยนกุญแจสำหรับการเข้ารหัสลับต่อไปนี้ แล้วตอบคำถามย่อย 1 และ 2

อัลกอริทึมดิฟฟี-เฮลแมน (Diffie-Hellman) เป็นหนึ่งในวิธีการแลกเปลี่ยนกุญแจสำหรับการเข้ารหัสลับในเครือข่ายที่ไม่ปลอดภัยที่ช่วยให้ผู้ใช้สองรายสามารถแลกเปลี่ยนกุญแจที่สามารถนำมาใช้ในการเข้ารหัสลับแบบสมมาตร (symmetric encryption) กับข้อความต่าง ๆ ในลำดับต่อไปได้

อัลกอริทึมนี้ใช้กระบวนการมอดุโล (modulo) ซึ่งกระบวนการ "$a \bmod b$" จะคำนวณหาเศษเหลือ (remainder) ที่ได้ เมื่อจำนวนเต็ม a ถูกหารด้วยจำนวนเต็ม b ตัวอย่างเช่น $29 \bmod 5$ ได้ 4 และ $21 \bmod 3$ ได้ 0

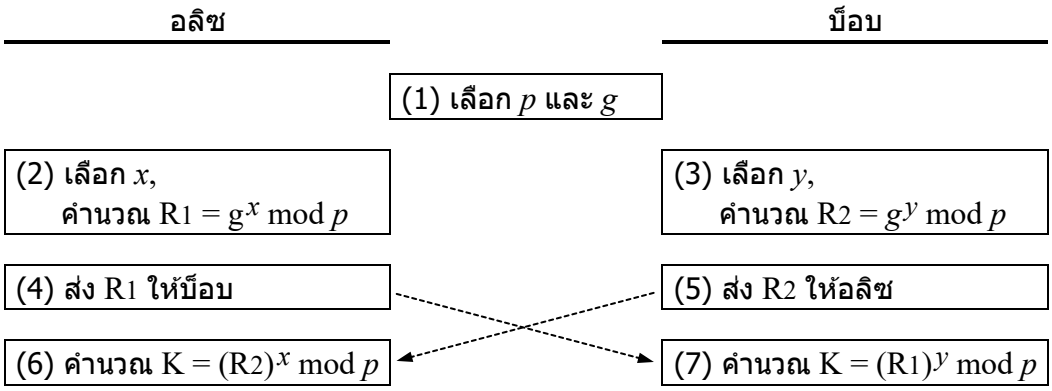
เพื่อให้ได้กุญแจลับ (secret key) ที่จะใช้ร่วมกัน ในอันดับแรก ผู้ใช้ทั้งสองจะต้องเลือกตัวเลขขึ้นมาสองจำนวน: จำนวนเฉพาะ (prime number) p และจำนวนเต็ม g ที่ค่าของ $g^n \bmod p$ ไม่ซ้อนทับกันในช่วง $1 \le n \le p - 1$ หรืออีกนัยหนึ่งคือแต่ละค่าของ n ($n = 1, 2, \ldots, p - 1$) ที่ปรากฏใน $g^n \bmod p$ ตารางที่ 1 แสดงตัวอย่างของคู่จำนวนเต็มดังกล่าว: $p = 11$ และ $g = 2$ ซึ่งแถวล่างแสดงชุดตัวเลข $1, 2, \ldots, 10$

ตารางที่ 1 ตัวอย่างของ $g^n \bmod p$ ที่มีคู่จำนวนเต็ม $p = 11$ และ $g = 2$

n	1	2	3	4	5	6	7	8	9	10
2^n	2	4	8	16	32	64	128	256	512	1024
$2^n \bmod 11$	2	4	8	5	10	9	7	3	6	1

อัลกอริทึมดิฟฟี-เฮลแมนถูกแสดงไว้ในรูปที่ 1 และในขั้นตอนต่อไปนี้
(1) อลิซและบ็อบเลือกจำนวนเฉพาะ p และจำนวนเต็ม g ดังอธิบายข้างต้น
(2) อลิซเลือกตัวเลขสุ่มขึ้นมาหนึ่งจำนวน, x ($1 \le x \le p - 1$), แล้วคำนวณหา $R1 = g^x \bmod p$
(3) บ็อบเลือกตัวเลขสุ่มขึ้นมาหนึ่งจำนวน, y ($1 \le y \le p - 1$), แล้วคำนวณหา $R2 = g^y \bmod p$
(4) อลิซส่ง R1 ให้บ็อบ โปรดสังเกตว่าอลิซไม่ได้ส่งค่า x
(5) บ็อบส่ง R2 ให้อลิซ โปรดสังเกตว่าบ็อบไม่ได้ส่งค่า y
(6) อลิซคำนวณกุญแจลับที่ใช้ร่วมกัน $K = (R2)^x \bmod p$
(7) บ็อบคำนวณกุญแจลับที่ใช้ร่วมกัน $K = (R1)^y \bmod p$

ดังนั้น ทั้งอลิซและบ็อบจึงมีกุญแจลับที่ใช้ร่วมกัน K สำหรับเซสชันนี้



รูปที่ 1 อัลกอริทึมแลกเปลี่ยนกุญแจดิฟฟี-เฮลแมน

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

กุญแจลับที่ใช้ร่วมกัน (shared secret key) กำลังถูกแลกเปลี่ยนระหว่างอลิซและบ็อบด้วยอัลกอริทึมดิฟฟี-เฮลแมน กำหนดให้ทั้งสองตกลงร่วมกันที่จะใช้จำนวนเฉพาะ $p = 11$ และจำนวนเต็ม $g = 2$ หากบ็อบได้รับกุญแจสาธารณะ $R1 = 3$ จากอลิซแล้ว ตัวเลขสุ่ม x ที่อลิซเป็นผู้เลือกคือ [A] และหากอลิซได้รับกุญแจสาธารณะ $R2 = 9$ จากบ็อบ ตัวเลขสุ่ม y ที่บ็อบเลือกก็คือ [B] ในกรณีนี้, นั่นคือ $R1 = 3$ และ $R2 = 9$, กุญแจลับที่ทั้งอลิซและบ็อบได้รับคือ [C] ซึ่งทั้งสองสามารถนำไปใช้ในการเข้ารหัสลับแบบสมมาตรกับข้อความต่าง ๆ ในลำดับต่อไป

กลุ่มคำตอบ

a) 1	b) 2	c) 3	d) 4	e) 5
f) 6	g) 7	h) 8	i) 9	j) 10

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

อลิซและบ็อบตัดสินใจที่จะเปลี่ยนค่าของ g, x และ y จากนั้น ทั้งคู่จึงแลกเปลี่ยนค่า $R1$ และ $R2$ ที่คำนวณได้และในที่สุดก็จะได้กุญแจลับที่ใช้ร่วมกัน K
เมื่ออีฟซึ่งเป็นผู้ไม่หวังดีทราบค่าที่ไม่ได้เป็นความลับสามค่า: $p = 11$, $g = 7$ และ $R1 = 3$ และหลังจากนั้น อีฟใช้วิธีมิชอบทำให้ทราบว่าค่า $y = 3$ ดังนั้นจึงเป็นไปได้ที่อีฟจะหาได้ว่ากุญแจลับที่ใช้ร่วมกัน K คือ [D]

ในกรณีนี้ ค่า p มีขนาดเล็กจนทำให้สามารถหาค่า x จาก $R1$ หรือ y จาก $R2$ ได้ในเวลาอันสั้น อย่างไรก็ตาม เนื่องจากวิธีการที่จะคำนวณในลักษณะนี้ได้อย่างมีประสิทธิภาพนั้นยังไม่ถูกค้นพบ การที่จะหาค่า x จาก $R1$ จำเป็นต้องใช้วิธีการค้นหาแบบบรูทฟอร์ส (brute-force search) เพื่อหาค่า x ที่ทำให้ $g^x \bmod p = R1$ ซึ่งหากการคำนวณหาค่า x สามารถทำได้ 2^{90} ครั้งในหนึ่งปีแล้ว ก็จะใช้เวลาประมาณ [E] ปีเพื่อหาค่า x หาก p ที่มีขนาดประมาณ 2^{120} ถูกนำมาใช้
สำหรับในการใช้งานจริงนั้น จำนวนเฉพาะที่ถูกใช้เป็น p จะมีขนาด 2^{2048} ($\approx 3.2 \times 10^{616}$) หรือมากกว่า

กลุ่มคำตอบสำหรับ D

a) 1	b) 2	c) 3	d) 4	e) 5
f) 6	g) 7	h) 8	i) 9	j) 10

กลุ่มคำตอบสำหรับ E

a) 2	b) 5	c) 10	d) 100
e) 1000	f) 1 ล้าน	g) 1 พันล้าน	

สำหรับข้อสอบข้อที่ Q2 ถึง Q5 ให้เลือกทำเพียงสองในสี่ข้อ
จากนั้น ให้ระบายทับในวงกลม Ⓢ ในกระดาษคำตอบสำหรับข้อที่เลือกทำ
หากเลือกตั้งแต่สามข้อขึ้นไป จะตรวจให้คะแนนเฉพาะสองข้อแรกเท่านั้น

Q2. **อ่านคำอธิบายเกี่ยวกับโครงแบบหน่วยความจำแคชแบบเชื่อมโยงโดยตรง (direct-mapped cache memory configuration) ต่อไปนี้ แล้วตอบคำถามย่อย**
ในคำถามนี้ ตัวอักษรที่เป็น*ตัวเอียง* เช่น *001* และ *ttt* แสดงถึงตัวเลขฐานสอง และตัวอักษรกอทิก (Gothic) เช่น 0F และ xx แสดงถึงตัวเลขฐานสิบหก

หน่วยความจำแคชถูกวางไว้ระหว่างซีพียูกับหน่วยความจำหลัก หน่วยความจำแคชถูกออกแบบเพื่อให้สามารถเข้าถึงข้อมูลที่ถูกเลือกไว้บางส่วนจากหน่วยความจำหลัก เนื่องจากขนาดของหน่วยความจำแคชนั้นเล็กกว่าหน่วยความจำหลักมาก ดังนั้น ข้อมูลขนาดใหญ่ที่ต้องนำมาใช้ในการประมวลผลจึงไม่สามารถจัดเก็บในหน่วยความจำแคชพร้อม ๆ กันได้

[แคชแบบเชื่อมโยงโดยตรง (Direct-mapped cache)]
วิธีนี้จำเป็นต้องใช้การเชื่อมโยง (mapping) เมื่อต้องการคัดลอกข้อมูลจากหน่วยความจำหลักมายังหน่วยความจำแคช โดยหนึ่งบล็อกในหน่วยความจำหลักจะถูกเชื่อมโยงเข้ากับตำแหน่งที่กำหนดไว้เพียงหนึ่งตำแหน่งในหน่วยความจำแคช ซึ่งนี่คือหนึ่งในหลายวิธีที่ใช้ทั่วไปในฟังก์ชันเชื่อมโยง

พิจาณาแบบจำลองหน่วยความจำอย่างง่ายดังนี้:
หน่วยความจำหลัก: ขนาด 256 ไบต์, ตำแหน่ง 0 ถึง 255 แต่ละตำแหน่งจัดเก็บข้อมูล 1 ไบต์
หน่วยความจำแคช: ขนาด 32 ไบต์ (=8 เวิร์ด) สำหรับเก็บข้อมูล
ข้อมูลถูกถ่ายโอนระหว่างหน่วยความจำหลักกับหน่วยความจำแคชในหน่วยเวิร์ด
ในแบบจำลองนี้ 1 เวิร์ดถูกกำหนดให้เป็นพื้นที่ 4 ไบต์ต่อเนื่องกันในหน่วยความจำหลัก เริ่มต้น ณ ตำแหน่งที่เป็นผลคูณของ 4

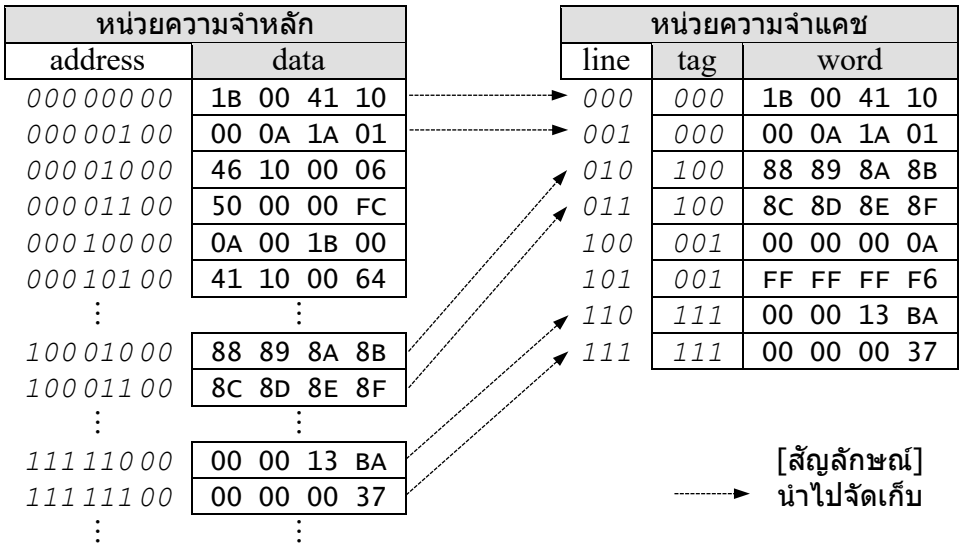
รูปที่ 1 แสดงความสัมพันธ์ระหว่างหน่วยความจำหลักกับหน่วยความจำแคช
ตำแหน่งที่อยู่ (address) 8-บิต ของหน่วยความจำหลักถูกแบ่งออกเป็น 3 ส่วน: tag-bits (3 บิต), line-bits (3 บิต) และ byte-bits (2 บิต) โดย tag-bits *ttt* ถูกใช้เพื่อระบุตำแหน่งของหน่วยความจำหลักในแคช line-bits *lll* ถูกใช้สำหรับการเชื่อมโยง; นั่นคือหนึ่งเวิร์ดในหน่วยความจำหลักที่มีตำแหน่ง *aaalll00* (*a*:ค่าใดก็ได้) จะถูกเชื่อมโยงเข้ากับตำแหน่ง *lll* ในแคชเสมอ ส่วน byte-bits *bb* ระบุออฟเซตของไบต์ในเวิร์ด

รูปที่ 1 ความสัมพันธ์ระหว่างหน่วยความจำหลักกับหน่วยความจำแคช

เมื่อซีพียูจะอ่านข้อมูลหนึ่งไบต์จากตำแหน่ง *tttlllbb* ซึ่งไม่ได้ถูกบรรจุอยู่ในหน่วยความจำแคช หน่วยประมวลผลก็จะถ่ายโอนเวิร์ดที่มีไบต์เป้าหมายขึ้นมาจากหน่วยความจำหลักไปยังตำแหน่งบรรทัด *lll* ในหน่วยความจำแคชที่มี tag-bits *ttt*

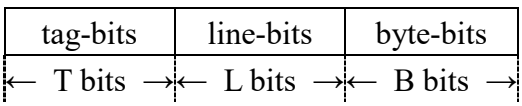
รูปที่ 2 แสดงแบบจำลองหน่วยความจำที่อธิบายไว้ข้างต้นพร้อมข้อมูลตัวอย่าง
ตัวอย่างเช่น เมื่อซีพียูจะอ่านข้อมูลไบต์ 1B ณ ตำแหน่ง 18 แล้ว [A] ในหน่วยความจำแคช หน่วยประมวลผลสามารถระบุตำแหน่งในหน่วยความจำหลักของเวิร์ดในหน่วยความจำแคชได้โดยใช้ตำแหน่งบรรทัด *lll* และ tag-bits *ttt* ตัวอย่างเช่น เวิร์ด FF FF FF F6 บนบรรทัด *101* นั้น สามารถระบุได้ว่าถูกบรรจุขึ้นมาจากหน่วยความจำหลักที่ตำแหน่ง [B]

เมื่อใดที่ซีพียูต้องการข้อมูลจากหน่วยความจำหลัก จะตรวจสอบว่าข้อมูลถูกบรรจุอยู่ในหน่วยความจำแคชแล้วหรือไม่ ถ้าข้อมูลอยู่ในหน่วยความจำแคชอยู่แล้ว จะเรียกสถานะนั้นว่า "cache hit" หรือพบในแคช; ไม่เช่นนั้นจะเรียกว่า "cache miss" หรือไม่พบในแคช
หากหน่วยความจำหลักและหน่วยความจำแคชมีข้อมูลดังแสดงในรูปที่ 2 เมื่อซีพียูอ่านข้อมูลขึ้นมาอย่างต่อเนื่องจากตำแหน่ง 0, 32, 2 และ 34 ตามลำดับดังนี้แล้ว ลำดับของสถานะที่เกิดขึ้นก็จะเป็น cache hit → cache miss → [C]



รูปที่ 2 แบบจำลองหน่วยความจำอย่างง่ายพร้อมข้อมูลตัวอย่าง

โดยทั่วไปแล้ว เมื่อหน่วยความจำหลักถูกแบ่งออกเป็นสามส่วนดังนี้:



คุณลักษณะของโครงแบบหน่วยความจำแคชแบบเชื่อมโยงโดยตรงดังที่ได้อธิบายข้างต้น สามารถแสดงได้ดังในตารางที่ 1 กำหนดให้แต่ละตำแหน่งที่อยู่ในหน่วยความจำหลักเก็บข้อมูลขนาด 1 ไบต์

รายการ	ค่า
ช่วงของที่อยู่ในหน่วยความจำหลัก	0 ถึง [D] - 1
จำนวนของเวิร์ดในหน่วยความจำหลัก	2^{T+L}
จำนวนของตำแหน่งบรรทัดในหน่วยความจำแคช	2^{L}
ขนาดของหน่วยความจำแคช (สำหรับส่วนของข้อมูลเวิร์ด)	2^{L+B} ไบต์
ขนาดของหน่วยความจำแคช (สำหรับส่วนของ tag-bits)	[E] บิต

ตารางที่ 1 คุณสมบัติของโครงแบบหน่วยความจำแคชแบบเชื่อมโยงโดยตรง

คำถามย่อย

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายข้างต้น

กลุ่มคำตอบสำหรับ A

a) **จะไม่เกิดการถ่ายโอนข้อมูลขึ้น เนื่องจากเวิร์ดที่จัดเก็บไบต์เป้าหมายนั้นมีอยู่แล้ว**
b) **เวิร์ด** 0A 00 1B 00 **และ** tag-bits *000* **จะถูกนำไปจัดเก็บที่ตำแหน่งบรรทัด** *100*
c) **เวิร์ด** 0A 00 1B 00 **และ** tag-bits *100* **จะถูกนำไปจัดเก็บที่ตำแหน่งบรรทัด** *000*
d) **เวิร์ด** 1B 00 41 10 **และ** tag-bits *000* **จะถูกนำไปจัดเก็บที่ตำแหน่งบรรทัด** *100*
e) **เวิร์ด** 1B 00 41 10 **และ** tag-bits *100* **จะถูกนำไปจัดเก็บที่ตำแหน่งบรรทัด** *000*

กลุ่มคำตอบสำหรับ B

a) 52 **ถึง** 55 (34 **ถึง** 37)
b) 60 **ถึง** 63 (3C **ถึง** 3F)
c) 164 **ถึง** 167 (A4 **ถึง** A7)
d) 204 **ถึง** 207 (CC **ถึง** CF)

กลุ่มคำตอบสำหรับ C

a) cache hit → cache hit
b) cache hit → cache miss
c) cache miss → cache hit
d) cache miss → cache miss

กลุ่มคำตอบสำหรับ D และ E

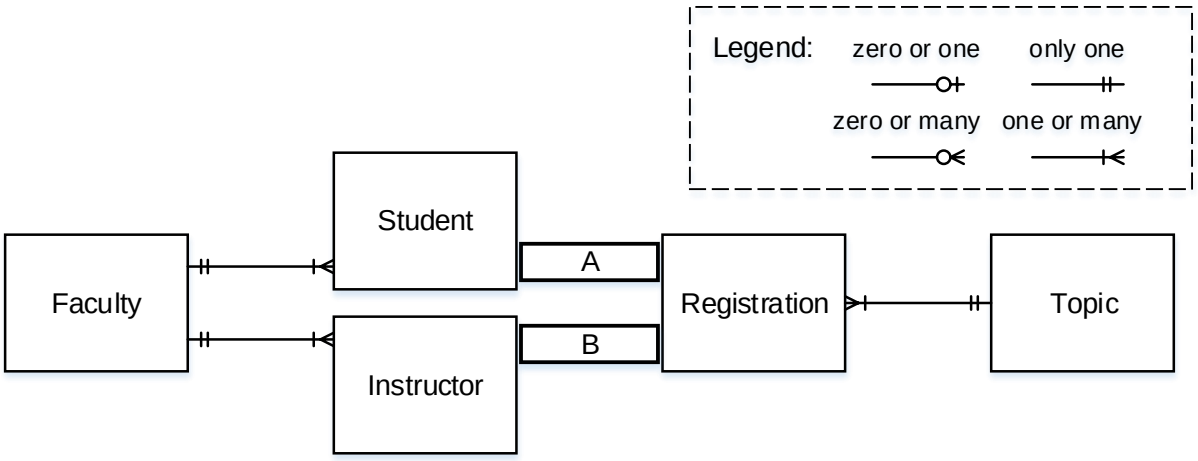
a) $2^{L-1} \times T$
b) $2^{T-1} \times L$
c) $2^{L} \times T$
d) $2^{T} \times L$
e) 2^{T+L-1}
f) 2^{T+L}
g) $2^{T+L+B-1}$
h) 2^{T+L+B}

สำหรับข้อสอบข้อที่ **Q2** ถึง **Q5 ให้เลือกทำเพียงสอง**ในสี่ข้อ

Q3. อ่านคำอธิบายของระบบจัดการวิทยานิพนธ์ของมหาวิทยาลัยแห่งหนึ่งต่อไปนี้ แล้วตอบคำถามย่อย 1 ถึง 3

มหาวิทยาลัย T มีหลายคณะ (Faculty) แต่ละคณะมีผู้สอน (Instructor) และนักศึกษา (Student) หลายคน
มหาวิทยาลัย T จัดการสอบวิทยานิพนธ์และพิธีสำเร็จการศึกษาปีละหนึ่งครั้ง โดยจะมีช่วงเวลาสำหรับการลงทะเบียนหัวข้อ (Topic) ที่นักศึกษาสามารถลงทะเบียนได้ว่าจะทำวิทยานิพนธ์เรื่องใดเพื่อสำเร็จการศึกษาก่อนถึงเวลาจัดการสอบวิทยานิพนธ์ นักศึกษาต้องติดต่อผู้สอนในคณะเพื่อขอให้เป็นที่ปรึกษาวิทยานิพนธ์ ผู้สอนแต่ละคนสามารถเป็นที่ปรึกษาให้นักศึกษาได้ไม่เกิน 5 คน นักศึกษาแต่ละคนจะต้องมีผู้สอนเป็นที่ปรึกษาหนึ่งคน หลังจากติดต่อผู้สอนแล้ว นักศึกษาต้องเข้าปรึกษาผู้สอนเพื่อกำหนดหัวข้อวิทยานิพนธ์ นักศึกษาแต่ละคนจะเลือกหนึ่งหัวข้อ โดยนักศึกษาสามารถทำวิทยานิพนธ์เพียงคนเดียว หรือจะทำเป็นกลุ่มไม่เกินกลุ่มละ 4 คนก็ได้ เมื่อนักศึกษาได้เลือกหัวข้อวิทยานิพนธ์แล้ว ก็จะลงทะเบียนหัวข้อนั้น ๆ กับแผนกจัดการศึกษาก่อนหมดเวลาลงทะเบียนตามที่กำหนดไว้

มหาวิทยาลัย T ต้องการสร้างระบบจัดการวิทยานิพนธ์ที่ช่วยให้นักศึกษาสามารถลงทะเบียนหัวข้อวิทยานิพนธ์ของตนได้ออนไลน์ รวมทั้งสามารถตรวจสอบผลการเรียนได้หลังจากได้สอบวิทยานิพนธ์แล้ว เพื่อพัฒนาระบบนี้ มหาวิทยาลัย T ได้สร้างแผนภาพ E-R อย่างง่ายขึ้นมาดังแสดงในรูปที่ 1



รูปที่ 1 แผนภาพ E-R (อย่างง่าย)

โครงสร้างตารางและตัวอย่างข้อมูลที่ถูกจัดเก็บไว้เป็นดังต่อไปนี้:

ตาราง Faculty

ID	FacultyName
FIT	Faculty of Information Technology
FEE	Faculty of Electronical Engineering

ตาราง Student

ID	StudentName	FacultyID	Email	DefenseTerm
S2101	Sarah	FIT	sarah@example.edu	2023
S2102	Karina	FIT	karina@example.edu	2023
S2103	Mike	FIT	mike@example.edu	2023
S2104	Selena	FEE	selena@example.edu	2023
S2105	Peter	FEE	peter@example.edu	2023
S2201	Roger	FEE	roger@example.edu	2024

ตาราง Instructor

ID	InstructorName	Title	FacultyID
I01	Christopher	Professor	FIT
I02	Lily	Associate Professor	FIT
I03	Marie	Professor	FEE
I04	Robert	Lecturer	FEE

ตาราง Topic

ID	TopicName
T01	Personal Area Networks using Zigbee Technology
T02	Rich Internet Application for Weekly College Timetable Generation
T03	General-purpose Online Ticket Reservation System
T04	Fiber Optic Communication

ตาราง Registration

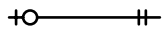
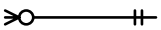
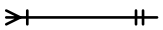
ณ เวลาลงทะเบียน ฟิลด์ Grade **จะถูกกำหนดให้เป็น** `null`

ID	StudentID	TopicID	InstructorID	Grade
R01	S2101	T02	I02	
R02	S2102	T03	I01	
R03	S2103	T03	I01	
R04	S2105	T04	I03	

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในรูปที่ 1

กลุ่มคำตอบสำหรับ A และ B

a) 
b) 
c) 
d) 
e) 
f) 

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำสั่ง SQL1

การสอบในปี ค.ศ. 2023 จะจัดขึ้นในเดือนธันวาคม ช่วงเวลาลงทะเบียนได้เริ่มขึ้นแล้วและกำลังจะหมดเวลาในไม่ช้า
แผนกจัดการศึกษาได้วางแผนส่งอีเมลแจ้งเตือนให้กับนักศึกษา
คำสั่ง SQL1 ต่อไปนี้ จะให้ผลลัพธ์เป็นนักศึกษาที่มีกำหนดสอบ (defense term) ในปี ค.ศ. 2023 และยังไม่ได้ลงทะเบียนหัวข้อวิทยานิพนธ์ไว้

```
-- SQL1 --
SELECT s.ID, s.StudentName, s.Email
FROM Student s [ C ] Registration r
     ON s.ID = r.StudentID
WHERE s.DefenseTerm = 2023 AND [ D ]
```

เมื่อคำสั่ง SQL1 ถูกเรียกใช้กับข้อมูลตัวอย่างที่แสดงในตารางข้างต้น คำสั่ง SQL1 จะให้ผลลัพธ์ดังนี้:

ID	StudentName	Email
S2104	Selena	selena@example.edu

หมายเหตุ:

`INNER JOIN` **คืนค่าเป็นแถวที่มีค่าตรงกันจากตารางด้านซ้ายและด้านขวา**

`LEFT OUTER JOIN` **คืนค่าเป็นแถวทั้งหมดจากตารางด้านซ้ายและแถวที่ตรงกันจากตารางด้านขวา โดยคอลัมน์ของแถวที่มีค่าไม่ตรงกันจะถูกกำหนดให้เป็น** `NULL`

`RIGHT OUTER JOIN` **คืนค่าเป็นแถวทั้งหมดจากตารางด้านขวาและแถวที่ตรงกันจากตารางด้านซ้าย โดยคอลัมน์ของแถวที่มีค่าไม่ตรงกันจะถูกกำหนดให้เป็น** `NULL`

กลุ่มคำตอบสำหรับ C

a) `INNER JOIN`
b) `LEFT OUTER JOIN`
c) `RIGHT OUTER JOIN`

กลุ่มคำตอบสำหรับ D

a) `r.ID IS NOT NULL`
b) `r.ID IS NULL`
c) `s.ID IS NOT NULL`
d) `s.ID IS NULL`

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

ในการสอบวิทยานิพนธ์ คณะกรรมการจะประเมินและให้คะแนนวิทยานิพนธ์ของนักศึกษา หลังสอบเสร็จแล้ว ผลการเรียนจะถูกบันทึกลงในตารางการลงทะเบียน (Registration)
ตารางต่อไปนี้แสดงข้อมูล ณ ปัจจุบันของตารางการลงทะเบียน โดยข้อมูลในตารางอื่น ๆ สี่ตารางนอกเหนือจากตารางการลงทะเบียนนี้ไม่มีการเปลี่ยนแปลง

ตาราง Registration

ID	StudentID	TopicID	InstructorID	Grade
R01	S2101	T02	I02	85
R02	S2102	T03	I01	90
R03	S2103	T03	I01	90
R04	S2105	T04	I03	80
R05	S2104	T01	I04	85

มหาวิทยาลัย T จะมอบรางวัลวิทยานิพนธ์ยอดเยี่ยมให้กับนักศึกษาในพิธีสำเร็จการศึกษา แผนกจัดการศึกษาจะประเมินผู้ได้รับรางวัลโดยใช้คำสั่ง SQL2 (ไม่แสดง) และ SQL3
อันดับแรก SQL2 (ไม่แสดง) จะแก้ไขตารางการลงทะเบียนด้วยการนำคอลัมน์ ID, TopicID และ InstructorID ออกไป แล้วจะเพิ่มคอลัมน์ StudentName และ FacultyID เข้ามา จากนั้นจะจัดเรียงแถวต่าง ๆ ตาม StudentID คำสั่ง SQL2 จะให้ผลลัพธ์เป็นตารางผลลัพธ์ (Result) ดังต่อไปนี้:

ตาราง Result

StudentID	StudentName	FacultyID	Grade
S2101	Sarah	FIT	85
S2102	Karina	FIT	90
S2103	Mike	FIT	90
S2104	Selena	FEE	85
S2105	Peter	FEE	80

อันดับถัดไป SQL3 จะระบุนักศึกษาที่จะได้รับรางวัลวิทยานิพนธ์ยอดเยี่ยมจากตารางผลลัพธ์ (Result) ที่ถูกสร้างขึ้นโดย SQL2 และตารางคณะ (Faculty)

```
-- SQL3 --
SELECT f.FacultyName, st.StudentID, st.StudentName, st.Grade
FROM (SELECT r.FacultyID   AS FacultyID,
             r.StudentID   AS StudentID,
             r.StudentName AS StudentName,
             r.Grade       AS Grade
      FROM Result r
      INNER JOIN (SELECT FacultyID, MAX(Grade) AS MaxGrade
                  FROM Result
                  GROUP BY Result.FacultyID
                 ) AS mg ON r.FacultyID = mg.FacultyID
                      AND r.Grade = mg.MaxGrade
     ) AS st
INNER JOIN Faculty f ON f.ID = st.FacultyID
```

เมื่อคำสั่ง SQL3 ถูกเรียกใช้กับข้อมูลตัวอย่างที่แสดงข้างต้นแล้ว SQL3 ระบุว่า [E] เป็นผู้ได้รับรางวัลวิทยานิพนธ์ยอดเยี่ยม

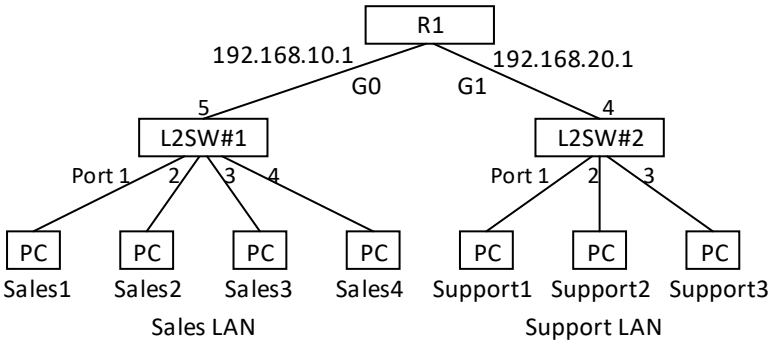
กลุ่มคำตอบสำหรับ E

a) นักศึกษาหนึ่งคนที่มี student ID เป็น S2102
b) นักศึกษาสองคนที่มี student ID เป็น S2102 และ S2103
c) นักศึกษาสองคนที่มี student ID เป็น S2102 และ S2104
d) นักศึกษาสามคนที่มี student ID เป็น S2102, S2103, และ S2104
e) นักศึกษาสามคนที่มี student ID เป็น S2102, S2104, และ S2105
f) นักศึกษาสี่คนที่มี student ID เป็น S2102, S2103, S2104, และ S2105

สำหรับข้อสอบข้อที่ **Q2** ถึง **Q5 ให้เลือกทำเพียงสอง**ในสี่ข้อ

Q4. อ่านคำอธิบายเกี่ยวกับเครือข่ายภายในของบริษัทแห่งหนึ่งต่อไปนี้ แล้วตอบคำถามย่อย 1 และ 2

บริษัท U เป็นผู้ให้บริการท่องเที่ยวขนาดเล็ก บริษัทนี้มีสองแผนก: แผนกขาย (Sales) และแผนกบริการ (Support) รูปที่ 1 แสดงโทโพโลยีเครือข่ายภายในของบริษัท U ที่มีสองเครือข่ายแลน แผนกขายมีเครือข่าย Sales LAN ที่มีพีซี 4 เครื่อง (Sales1, Sales2, Sales3, Sales4) เชื่อมต่ออยู่กับสวิทช์เลเยอร์ 2 L2SW#1 แผนกบริการมีเครือข่าย Support LAN ที่มีพีซี 3 เครื่อง (Support1, Support2, Support3) เชื่อมต่ออยู่กับสวิทช์เลเยอร์ 2 L2SW#2



รูปที่ 1 โทโพโลยีเครือข่ายภายในของบริษัท U ที่มีสองเครือข่ายแลน

[ตาราง MAC address]
สวิทช์เครือข่ายมีหลายพอร์ตที่อาจนำพีซีมาเชื่อมต่อได้ โดยแต่ละอุปกรณ์ที่เชื่อมต่ออยู่กับสวิทช์จะระบุตัวตนด้วยที่อยู่เลเยอร์ 2 ซึ่งก็คือ MAC address ของอุปกรณ์นั้น ๆ ดังนั้น สวิทช์แต่ละตัวจะติดตามอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่ด้วยโดยใช้ตารางแสดงการเชื่อมโยงที่เรียกว่าตาราง MAC address ตารางที่ 1 แสดงตัวอย่างตาราง MAC address ของสวิทช์

ตารางที่ 1 ตัวอย่างตาราง MAC address ของสวิทช์

พอร์ต	MAC address
1	00:00:5E:00:53:A2
2	00:00:5E:00:53:3B

ตาราง MAC address สามารถกำหนดได้ด้วยตนเองโดยผู้ดูแลเครือข่าย หรืออีกทางหนึ่งสวิทช์ก็สามารถเรียนรู้ MAC address ของอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออยู่กับพอร์ตของตนได้ในแบบไดนามิก ในกรณีของการเรียนรู้แบบไดนามิกนั้น ในตอนแรก ตาราง MAC address จะว่างอยู่ และสวิทช์จะไม่ทราบว่ามีอุปกรณ์ใดเชื่อมต่ออยู่กับพอร์ตใดของตนบ้าง ดังนั้น เมื่อสวิทช์ได้รับเฟรมใด ๆ ที่พอร์ต 1 สวิทช์ก็จะกระจายเฟรมไปยังทุก ๆ พอร์ต (ยกเว้นพอร์ต 1) เนื่องจากยังไม่ทราบหมายเลขพอร์ตที่เครื่องพีซีซึ่งมี MAC address ปลายทางเชื่อมต่ออยู่

[การเรียนรู้ที่อยู่ (address learning) ของสวิทช์]
เมื่อมีเฟรมมาถึงสวิทช์ MAC address ต้นทางจะถูกเพิ่มลงในตาราง MAC address ตัวอย่างเช่น เมื่อเครื่องพีซี Sales1 จะส่งเฟรมไปให้พีซี Sales4 แล้ว MAC address ต้นทางของ Sales1 จะถูกเชื่อมโยงเข้ากับพอร์ตหมายเลข 1 ของ L2SW#1 ดังนั้น ตาราง MAC address ของสวิทช์จะมีข้อมูลเพิ่มขึ้นโดยดูจากที่อยู่ต้นทางในเฟรมหรือ source MAC address เมื่อที่อยู่ปลายทางในเฟรมหรือ destination MAC address เป็นที่รู้จักของสวิทช์แล้ว เฟรมนั้นก็จะถูกส่งต่อไปยังพอร์ตที่สอดคล้องกันของสวิทช์ ดังนั้น การส่งต่อเฟรมจึงกระทำโดยดูจากตาราง MAC address ของสวิทช์

[การร้องขอ / ตอบกลับ ARP]
เมื่อพีซีเครื่องหนึ่งต้องการส่งเฟรมไปยังพีซีอีกเครื่องหนึ่งในแลนเดียวกัน จะต้องสร้างเฟรมเลเยอร์ 2 (layer-2 frame) รูปที่ 2 แสดงรูปแบบของเฟรมเลเยอร์ 2
พีซีที่เป็นผู้ส่ง ทราบ MAC address ของตนเองอยู่แล้ว อย่างไรก็ตาม ผู้ส่งอาจไม่ทราบ MAC address ปลายทางแม้จะทราบ IP address ปลายทางก็ตาม ดังนั้น ผู้ส่งจะต้องหา MAC address ของเครื่องพีซีปลายทางจาก IP address ปลายทางด้วยโพรโทคอล ARP

Destination MAC Address	Source MAC address	Payload

รูปที่ 2 รูปแบบของเฟรมเลเยอร์ 2

ในโพรโทคอล ARP นั้น อุปกรณ์ต้นทางจะกระจายข้อความที่เรียกว่า ARP request (การร้องขอ ARP) เพื่อให้ทราบ MAC address ของอุปกรณ์ปลายทาง ซึ่งอุปกรณ์อื่น ๆ นอกจากเครื่องปลายทางจะเพิกเฉยต่อการร้องขอนี้เนื่องจากทราบว่าตนไม่ใช่เป้าหมาย ขณะที่อุปกรณ์ปลายทางที่ถูกต้องนั้นจะตอบกลับไปยังอุปกรณ์ต้นทางด้วยข้อความแบบระบุผู้รับเครื่องเดียว (unicast) ที่เรียกว่า ARP reply (การตอบกลับ ARP) ดังนั้น อุปกรณ์ต้นทางก็จะได้รับ MAC address ของอุปกรณ์ปลายทาง

หากอุปกรณ์ปลายทางอยู่ในเครือข่ายแลนเดียวกัน อุปกรณ์ต้นทางก็จะหา MAC address ของอุปกรณ์ปลายทางและส่งเฟรมไปให้โดยตรง อย่างไรก็ตาม หากอุปกรณ์ปลายทางอยู่ต่างเครือข่ายแลน อุปกรณ์ต้นทางจะต้องส่งเฟรมไปยังที่อยู่ของเกตเวย์ตั้งต้น (default gateway) ก่อน (ซึ่งก็คือ IP address ของอินเทอร์เฟสของเราเตอร์) เมื่อเกตเวย์ได้รับเฟรมแล้ว ก็จะส่งต่อเฟรมดังกล่าวไปยังลิงก์ถัดไปในลักษณะเดียวกัน

ตารางที่ 2 และตารางที่ 3 แสดง IP address และ MAC address ของเครื่องพีซีต่าง ๆ ใน Sales LAN และ Support LAN
ที่อยู่เกตเวย์ตั้งต้นของ Sales LAN คือ 192.168.10.1 และที่อยู่เกตเวย์ตั้งต้นของ Support LAN คือ 192.168.20.1

ตารางที่ 2 IP address และ MAC address ของพีซีใน Sales LAN

ชื่อเครื่องพีซี	IP address	MAC Address
Sales1	192.168.10.2/24	00:00:5E:00:53:11
Sales2	192.168.10.3/24	00:00:5E:00:53:22
Sales3	192.168.10.4/24	00:00:5E:00:53:33
Sales4	192.168.10.5/24	00:00:5E:00:53:44

ตารางที่ 3 IP address และ MAC address ของพีซีใน Support LAN

ชื่อเครื่องพีซี	IP address	MAC Address
Support1	192.168.20.2/24	00:00:5E:00:53:55
Support2	192.168.20.3/24	00:00:5E:00:53:66
Support3	192.168.20.4/24	00:00:5E:00:53:77

ตารางที่ 4 แสดง IP address and MAC address ของอินเทอร์เฟสต่าง ๆ ของเราเตอร์ R1

ตารางที่ 4 IP address และ MAC address ของเราเตอร์ R1

ชื่ออุปกรณ์	อินเทอร์เฟส	IP address	MAC Address
R1	Gigabit 0 (G0)	192.168.10.1/24	00:00:5E:00:53:AA
	Gigabit 1 (G1)	192.168.20.1/24	00:00:5E:00:53:BB

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

กำหนดให้ตาราง MAC address ของ L2SW#1 ยังว่างอยู่ เมื่อเครื่อง Sales1 ส่งแพคเก็ต (packet) ไปยัง Sales3 L2SW#1 จะส่งการร้องขอ ARP ไปยังพอร์ตหมายเลข [A]

เมื่อ Sales1 ต้องการส่งแพคเก็ตไปยัง Support2 เป็นครั้งแรก ในอันดับแรก Sales1 จะส่ง [B] เพื่อหา [C] ของ [D] เนื่องจาก Sales1 และ Support2 ไม่ได้อยู่บนเครือข่ายเดียวกัน

กลุ่มคำตอบสำหรับ A

a) 1, 2, 3 และ 4
b) 1, 2, 3, 4 และ 5
c) 2, 3, 4 และ 5
d) 3 เพียงพอร์ตเดียว
e) 5 เพียงพอร์ตเดียว

กลุ่มคำตอบสำหรับ B

a) การเรียนรู้ที่อยู่
b) การตอบกลับ ARP
c) การร้องขอ ARP
d) ตาราง MAC address

กลุ่มคำตอบสำหรับ C และ D

a) เกตเวย์ตั้งต้น
b) IP address
c) L2SW#1
d) L2SW#2
e) MAC address
f) Sales1
g) Support2

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้ ในที่นี้ คำตอบที่จะถูกเติมลงใน E1 และ E2 ควรเลือกมาจากการจับคู่ที่ถูกต้องในกลุ่มคำตอบสำหรับ E และคำตอบที่จะถูกต้องลงใน F1 และ F2 ควรเลือกมาจากการจับคู่ที่ถูกต้องในกลุ่มคำตอบสำหรับ F

หลังจากได้ทำการหา ARP (ARP resolution) ในคำถามย่อย 1 หาก Sales1 ต้องการส่งแพคเก็ตไปยัง Support1 แล้ว แพคเก็ตจะถูกห่อหุ้ม (encapsulate) ที่ชั้นเชื่อมโยงข้อมูล (data link layer) ของ Sales1 ดังนี้:

ส่วนหัวของเฟรม		ส่วนหัวของ IP		
MAC ต้นทาง	MAC ปลายทาง	IP ต้นทาง	IP ปลายทาง	ข้อมูล
00:00:5E:00:53:11	E1	192.168.10.2	E2	Some data

เมื่อ Support1 ได้รับแพคเก็ตที่ถูกส่งมาจาก Sales1 แล้ว แพคเก็ตที่ Support1 จะเป็นดังนี้:

ส่วนหัวของเฟรม		ส่วนหัวของ IP		
MAC ต้นทาง	MAC ปลายทาง	IP ต้นทาง	IP ปลายทาง	ข้อมูล
F1	00:00:5E:00:53:55	F2	192.168.20.2	Some data

กลุ่มคำตอบสำหรับ E

	E1	E2
a)	00:00:5E:00:53:55	192.168.10.1
b)	00:00:5E:00:53:55	192.168.20.1
c)	00:00:5E:00:53:55	192.168.20.2
d)	00:00:5E:00:53:AA	192.168.10.1
e)	00:00:5E:00:53:AA	192.168.20.1
f)	00:00:5E:00:53:AA	192.168.20.2
g)	00:00:5E:00:53:BB	192.168.10.1
h)	00:00:5E:00:53:BB	192.168.20.1
i)	00:00:5E:00:53:BB	192.168.20.2

กลุ่มคำตอบสำหรับ F

	F1	F2
a)	00:00:5E:00:53:11	192.168.10.1
b)	00:00:5E:00:53:11	192.168.10.2
c)	00:00:5E:00:53:11	192.168.20.1
d)	00:00:5E:00:53:AA	192.168.10.1
e)	00:00:5E:00:53:AA	192.168.10.2
f)	00:00:5E:00:53:AA	192.168.20.1
g)	00:00:5E:00:53:BB	192.168.10.1
h)	00:00:5E:00:53:BB	192.168.10.2
i)	00:00:5E:00:53:BB	192.168.20.1

สำหรับข้อสอบข้อที่ Q2 ถึง Q5 ให้เลือกทำเพียงสองในสี่ข้อ

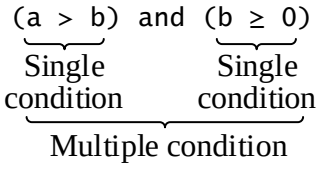
Q5. อ่านคำอธิบายเกี่ยวกับการออกแบบการทดสอบสำหรับซอฟต์แวร์ต่อไปนี้ แล้วตอบคำถามย่อย 1 ถึง 3

บริษัท V ซึ่งเป็นผู้พัฒนาซอฟต์แวร์ กำลังทบทวนวิธีการทดสอบเพื่อลดจำนวนจุดบกพร่อง (bug) ต่าง ๆ ในโปรแกรมบริษัทพัฒนาขึ้นมา

[คำอธิบายวิธีการทดสอบที่ถูกใช้ในบริษัท V]
บริษัท V ทดสอบโปรแกรมต่าง ๆ ที่พัฒนาขึ้นมาด้วยการทดสอบกระแสการควบคุม (control flow testing) ซึ่งเป็นหนึ่งในวิธีทดสอบแบบกล่องขาว (white box testing method)
การทดสอบกระแสควบคุมมุ่งไปที่หน่วยที่เล็กที่สุดต่าง ๆ ที่ประกอบกันขึ้นมาเป็นโปรแกรม เช่น เส้นทางคำสั่ง และเงื่อนไขการตัดสินใจ กรณีทดสอบและข้อมูลทดสอบจะถูกจัดเตรียมขึ้นมาให้สอดคล้องกับเงื่อนไขการควบคุม (coverage criteria) ที่ถูกกำหนดขึ้นระหว่างการวางแผนการทดสอบ จากนั้นจึงทำการตรวจสอบพฤติกรรมของโปรแกรมที่ถูกพัฒนาขึ้น
เกณฑ์การครอบคลุม (coverage criteria) มีทั้งการครอบคลุมคำสั่ง (statement coverage) ซึ่งทุก ๆ คำสั่งจะต้องทำงานอย่างน้อยหนึ่งครั้งในการทดสอบ และการครอบคลุมเงื่อนไขการตัดสินใจหรือการครอบคลุมทางเลือก (branch coverage) ซึ่งทุก ๆ เส้นทางที่เกิดขึ้นหลังจากแต่ละทางเลือกต้องได้รับการทดสอบอย่างน้อยหนึ่งครั้ง
บริษัท V ใช้การครอบคลุมทางเลือกเป็นเกณฑ์การครอบคลุม

[คำอธิบายของการประเมินทางลัด (short-cut evaluation)]
เงื่อนไขตัดสินใจสำหรับทางเลือกหนึ่ง ๆ มีทั้งเงื่อนไขเดี่ยว (single condition) ที่ประเมินเพียงหนึ่งเงื่อนไข และเงื่อนไขหลายกรณี (multiple condition) ที่จะประเมินในลักษณะของการนำเงื่อนไขเดี่ยวตั้งแต่สองเงื่อนไขขึ้นไปมารวมกันด้วย "และ (and)" หรือ "หรือ (or)" ตัวอย่างต่อไปนี้แสดงให้เห็นสองเงื่อนไขเดี่ยว (single condition) และหนึ่งเงื่อนไขหลายกรณี (multiple condition)

ตัวอย่าง:



โดยทั่วไป เมื่อโปรแกรมหนึ่งถูกประมวลผล การประเมินทางลัดจะถูกนำไปใช้กับเงื่อนไขหลายกรณี ในการประเมินทางลัด เงื่อนไขเดี่ยวต่าง ๆ ที่ประกอบกันขึ้นเป็นเงื่อนไขหลายกรณีนั้นจะถูกประเมินตามลำดับความสำคัญ เมื่อได้ผลลัพธ์ของเงื่อนไขหลายกรณีแล้ว ก็ไม่ต้องนำเงื่อนไขเดี่ยวอื่น ๆ มาพิจารณาอีกต่อไป ตัวอย่างเช่น ในกรณีที่เงื่อนไขหลายกรณีที่มีเงื่อนไขเดี่ยวสองเงื่อนไขมาใช้ร่วมกันด้วย "and" หากผลลัพธ์ของเงื่อนไขแรกเป็นเท็จ (false) แล้ว ผลลัพธ์ของเงื่อนไขหลายกรณีดังกล่าวก็จะเป็นเท็จ ไม่ว่าผลลัพธ์ของเงื่อนไขเดี่ยวที่สองจะเป็นอย่างไร ดังนั้น ในกรณีนี้ก็ไม่จำเป็นต้องทำการประเมินเงื่อนไขเดี่ยวที่สองอีกต่อไป

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

รูปที่ 1 แสดงโปรแกรมตัวอย่างที่จะทำการทดสอบ และตารางที่ 1 แสดงกรณีทดสอบตัวอย่างสำหรับโปรแกรมนี้ เมื่อโปรแกรมถูกนำมาทดสอบด้วยกรณีทดสอบตามวิธีการที่ถูกใช้ในบริษัท V แล้ว ผลลัพธ์การทดสอบพบว่า [A] ในกรณีทดสอบ 1 และ [B] ในกรณีทดสอบ 2 ในที่นี้ การประเมินทางลัดถูกนำมาใช้กับเงื่อนไขหลายกรณี

```
FUNCTION func(INT: x, INT: a, INT: b, INT: c, INT: d) {
   IF (x > 10) {
      func1();
      IF ((a < 10) or (b < 20)) {
         func2();
      } ELSE {
         func3();
      }
      IF ((c > 10) and (d > 10)) {
         func4();
      } ELSE {
         func5();
      }
   }
}
```

รูปที่ 1 โปรแกรมตัวอย่างที่จะทำการทดสอบ

ตารางที่ 1 กรณีทดสอบตัวอย่าง

	ข้อมูลทดสอบ				
ตัวแปร	x	a	b	c	d
กรณีทดสอบ 1	11	9	19	10	10
กรณีทดสอบ 2	11	10	20	11	11

กลุ่มคำตอบสำหรับ A และ B

a) b < 20 ไม่ได้รับการประเมิน
b) b < 20 และ c > 10 ไม่ได้รับการประเมิน
c) b < 20 และ d > 10 ไม่ได้รับการประเมิน
d) c > 10 ไม่ได้รับการประเมิน
e) c > 10 และ d > 10 ไม่ได้รับการประเมิน
f) d > 10 ไม่ได้รับการประเมิน
g) เงื่อนไขเดี่ยวทั้งหมดได้รับการประเมิน

คำถามย่อย 2

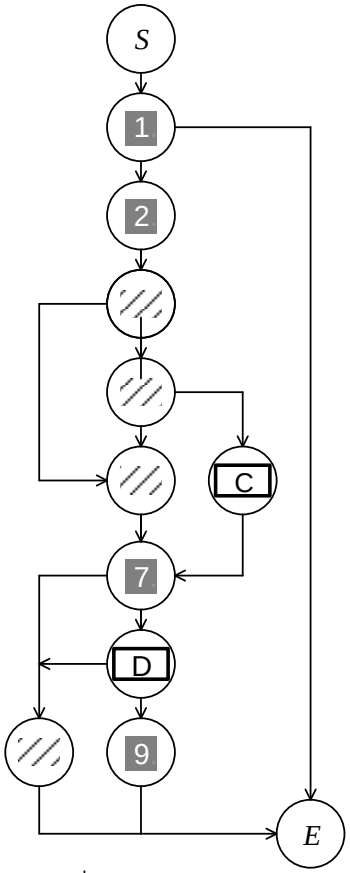
โครงสร้างควบคุมของโปรแกรมหนึ่ง ๆ นั้น สามารถแสดงให้เห็นได้ด้วยกราฟกระแสควบคุม (control flow graph) ในกราฟกระแสควบคุม กระบวนการต่าง ๆ จะถูกแบ่งออกเป็นคำสั่งตามลำดับ (serial instruction) คำสั่งวนซ้ำ (iteration instruction) และคำสั่งทางเลือก (branch instruction) แต่ละส่วนจะถูกนำไปใส่ในบล็อกกระบวนการ (process block) ซึ่งจากนี้ไปจะเรียกว่าโหนด (node) ที่ถูกเชื่อมต่อกันด้วยส่วนของเส้นตรงระบุทิศทาง ซึ่งจากนี้ไปจะเรียกว่าเส้นเชื่อม (edge) ตามลำดับของการประมวลผล ในที่นี้ เงื่อนไขหลายกรณีจะถูกแบ่งออกเป็นเงื่อนไขเดี่ยวตามลำดับ แล้วจึงนำไปใส่ในกราฟกระแสควบคุม

รูปที่ 2 ถูกจัดเตรียมขึ้นด้วยการกำหนดโหนดหมายเลข 1 ถึง 10 ให้กับโปรแกรมตัวอย่างในรูปที่ 1 รูปที่ 3 แสดงกราฟกระแสควบคุมของโปรแกรมนี้ โดยวงกลมแสดงถึงโหนดและลูกศรแสดงถึงเส้นเชื่อม หมายเลขโหนดในรูปที่ 3 ตรงกับหมายเลขโหนดในรูปที่ 2 โหนด S และ E ในรูปที่ 3 เป็นโหนดพิเศษที่แสดงถึงทางเข้าและทางออกของโปรแกรมตามลำดับ และไม่ได้ตรงกับกระบวนการใด ๆ ในโปรแกรมตัวอย่าง

จากกลุ่มคำตอบด้านล่าง ให้เลือกหมายเลขโหนดที่เหมาะสมเพื่อเติมลงในช่องว่าง ☐ แต่ละช่องในรูปที่ 3

```
FUNCTION func(INT: x, INT: a, INT: b,
                      INT: c, INT: d) {
   IF [1](x > 10) {
      [2]func1();
      IF ([3](a < 10) or [4](b < 20)) {
         [5]func2();
      } ELSE {
         [6]func3();
      }
      IF ([7](c > 10) and [8](d > 10)) {
         [9]func4();
      } ELSE {
         [10]func5();
      }
   }
}
```

รูปที่ 2 โปรแกรมตัวอย่างพร้อมหมายเลขโหนด



หมายเหตุ: ส่วนที่แรเงา ▨ ถูกซ่อนไว้

รูปที่ 3 กราฟกระแสควบคุม

กลุ่มคำตอบสำหรับ C และ D

a) 3 b) 4 c) 5

d) 6 e) 8 f) 10

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

สำหรับการทดสอบโปรแกรมที่แสดงในรูปที่ 1 ในกรณีของการครอบคลุมทางเลือกที่ใช้โดยบริษัท V จำนวนกรณีทดสอบขั้นต่ำที่ต้องใช้คือ [E]

นอกจากนี้ ยังมีวิธีสำหรับสร้างกรณีทดสอบด้วยการดึงเส้นทาง (path) ต่าง ๆ จากกราฟกระแสควบคุม จำนวนขั้นต่ำของเส้นทาง (P) ที่ครอบคลุมเส้นเชื่อมและโหนดทั้งหมดของกราฟกระแสควบคุมนั้น สามารถหาได้จากสมการต่อไปนี้:

$$P = \text{จำนวนของเส้นเชื่อม} - \text{จำนวนของโหนด} + 2$$

การทดสอบสำหรับกรณีทดสอบจำนวน P กรณีที่สอดคล้องกับเส้นทางต่าง ๆ ที่ได้มานี้ สามารถช่วยให้มั่นใจได้ว่าจะครอบคลุมมากกว่าการครอบคลุมทางเลือก

เมื่อพิจารณาจากกราฟกระแสควบคุมในรูปที่ 3 แล้ว ค่าของ P คือ [F]

เพื่อลดจำนวนจุดบกพร่องที่ยังไม่ได้รับการแก้ไขและยังหลงเหลือในโปรแกรมของบริษัท บริษัท V ตัดสินใจที่จะทดสอบโปรแกรมต่าง ๆ ด้วยกรณีทดสอบที่ได้จากกราฟกระแสควบคุม

กลุ่มคำตอบสำหรับ E และ F

a) 2 b) 3 c) 4

d) 5 e) 6 f) 7

คำถาม **Q6** เป็น **คำถามบังคับ**

Q6. อ่านคำอธิบายเกี่ยวกับโปรแกรมที่ใช้อัลกอริทึมบิตแทป (Bitap algorithm) เพื่อการค้นหาสตริงต่อไปนี้ แล้วตอบคำถามย่อย 1 ถึง 3

[คำอธิบายโปรแกรม]
ฟังก์ชัน `BitapMatch` เป็นโปรแกรมที่ใช้อัลกอริทึมบิตแทป (Bitap algorithm) เพื่อค้นหาสตริง อัลกอริทึมบิตแทปมีลักษณะเฉพาะอยู่ที่การใช้ลำดับของบิตที่ถูกกำหนดขึ้นสำหรับอักขระแต่ละตัว โดยเฉพาะในการเปรียบเทียบระหว่างสตริงที่จะถูกนำมาค้นหา ซึ่งจากนี้ไปจะเรียกสตริงเป้าหมาย (target string) กับสตริงที่ใช้ค้นหา (search string)
ในคำถามนี้ ชนิดข้อมูลไบนารี 16 บิต (16-bit Binary) ถูกแสดงโดยตัดเลขศูนย์ที่นำหน้าออกไป ตัวอย่างเช่น สำหรับค่า 0000000000010101 เมื่อนำเลขศูนย์นำหน้าออกไปแล้ว ค่านี้จะถูกแสดงเป็น `''10101''B`

(1) ฟังก์ชัน `BitapMatch` ใช้อาร์เรย์อักขระสองตัว: `Text[]` ใช้จัดเก็บสตริงเป้าหมาย และ `Pat[]` ใช้จัดเก็บสตริงค้นหา ดัชนีของอาร์เรย์ทั้งสองนี้เริ่มต้นที่ 1
อักขระลำดับที่ `i` ของ `Text[]` ถูกแทนด้วย `Text[i]` และอักขระลำดับที่ `i` ของ `Pat[]` ถูกแทนด้วย `Pat[i]` ทั้งสตริงเป้าหมายและสตริงค้นหาประกอบด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่เท่านั้น และทั้งสองมีความยาวสูงสุดเป็น 16 ตัวอักษร
รูปที่ 1 แสดงตัวอย่างของการจัดเก็บเมื่อสตริงเป้าหมาย `Text[]` คือ “AACBBAACABABAB” และสตริงค้นหา `Pat[i]` คือ “ACABAB”

ลำดับที่ของสมาชิก	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Text[]	A	A	C	B	B	A	A	C	A	B	A	B	A	B

ลำดับที่ของสมาชิก	1	2	3	4	5	6
Pat[]	A	C	A	B	A	B

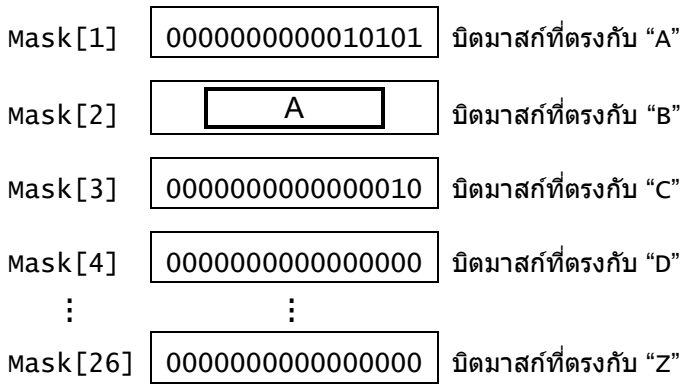
รูปที่ 1 ตัวอย่างของการจัดเก็บสตริงเป้าหมายและสตริงค้นหา

(2) ฟังก์ชัน BitapMatch เรียกใช้ฟังก์ชัน `GenerateBitMask`
สำหรับแต่ละตัวอักษรตั้งแต่ “A” ถึง “Z” ฟังก์ชัน `GenerateBitMask` จะสร้างลำดับบิต ซึ่งจากนี้ไปจะเรียกบิตมาสก์ (bit mask) ที่ตรงกับสตริงค้นหา และจัดเก็บไว้ในอาร์เรย์ชนิดไบนารี 16 บิต `Mask[]` ที่มีสมาชิก (element) 26 ตัว
ใน `Mask[1]` จัดเก็บบิตมาสก์ที่ตรงกับอักขระ “A” ใน `Mask[2]` จัดเก็บบิตมาสก์ที่ตรงกับอักขระ “B” และเป็นไปตามนี้เรื่อย ๆ ดังนั้น ใน `Mask[1]` จนถึง `Mask[26]` ก็จะเก็บบิตมาสก์ที่ตรงกับอักขระตั้งแต่ “A” ถึง “Z”

ฟังก์ชัน GenerateBitMask เริ่มต้นด้วยการกำหนดค่าให้สมาชิกแต่ละตัวของ Mask[] เป็น "0"B จากนั้นสำหรับแต่ละ i ที่มีค่าเท่ากับหรือมากกว่า 1 และเท่ากับหรือน้อยกว่าจำนวนของตัวอักษรใน Pat[] ก็จะกำหนดค่า 1 ให้กับบิตที่ตำแหน่ง i โดยเริ่มจากตำแหน่งต่ำที่สุดของค่าที่ถูกจัดเก็บอยู่ใน Mask[Index(Pat[i])] ซึ่งก็คือสมาชิกของ Mask[] ที่สอดคล้องกับ Pat[i]

ฟังก์ชัน Index คืนค่าเป็นจำนวนเต็ม n (1 ≦ n ≦ 26) เมื่อถูกเรียกใช้โดยมีอาร์กิวเมนต์เป็นตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ลำดับที่ n

(3) ในตัวอย่าง ซึ่ง Pat[] คือ "ACABAB" ดังแสดงในรูปที่ 1 นั้น เมื่อเรียกใช้ฟังก์ชัน GenerateBitMask แล้ว จะได้ Mask[] เป็นดังที่แสดงในรูปที่ 2



รูปที่ 2 ค่าของ Mask[] ที่ตรงกับ Pat[] ที่แสดงในรูปที่ 1

(4) ข้อกำหนดของอาร์กิวเมนต์ต่าง ๆ และการคืนค่าของฟังก์ชัน GenerateBitMask เป็นดังแสดงในตารางที่ 1

ตารางที่ 1 ข้อกำหนดของอาร์กิวเมนต์และการคืนค่าของฟังก์ชัน GenerateBitMask

อาร์กิวเมนต์ / การคืนค่า	ชนิดข้อมูล	อินพุต / เอาต์พุต	คำอธิบาย
Pat[]	CHAR	Input	อาร์เรย์หนึ่งมิติที่จัดเก็บสตริงค้นหา
Mask[]	16-bit Binary	Output	อาร์เรย์หนึ่งมิติที่จัดเก็บบิตมาสต์ที่ตรงกับตัวอักษรตั้งแต่ "A" ถึง "Z"
Return value	INT	Output	จำนวนของตัวอักษรในสตริงค้นหา

(5) ในโปรแกรมนี้ มีตัวดำเนินการ (operator) ที่ถูกใช้งานสามชนิด "|", "&" และ "<<":

- $a \mid b$ หาผลบวกทางตรรกะ (OR) ขนาด 16 บิต สำหรับแต่ละคู่ของบิตที่อยู่ในตำแหน่งเดียวกัน ของข้อมูล 16-bit binary สองจำนวน a และ b
- $a \,\&\, b$ หาผลคูณทางตรรกะ (AND) ขนาด 16 บิต สำหรับแต่ละคู่ของบิตที่อยู่ในตำแหน่งเดียวกัน ของข้อมูล 16-bit binary สองจำนวน a และ b
- $a << b$ ทำการเลื่อนบิตข้อมูล 16-bit binary a ไปทางซ้ายแบบตรรกะเป็นจำนวน b บิต

[โปรแกรม 1]

```
FUNCTION: GenerateBitMask(CHAR: Pat[], 16-bit Binary: Mask[]) {
   INT: i, PatLen

   PatLen ← number of characters in Pat[];
   FOR (i ← 1; i ≤ 26; i ← i + 1) {
      Mask[i] ← [ B ];     /* Initialize */
   }
   FOR (i ← 1; i ≤ PatLen; i ← i + 1) {
      Mask[Index(Pat[i])] ← ( [ C ] ) | Mask[Index(Pat[i])];
   }
   return (PatLen);
}
```

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายโปรแกรมและโปรแกรม 1

กลุ่มคำตอบสำหรับ A

a) 0000000000000101
b) 0000000000101000
c) 0001010000000000
d) 1010000000000000

กลุ่มคำตอบสำหรับ B

a) "0"B
b) "1"B
c) "1"B << (PatLen - 1)
d) "1"B << PatLen
e) "1111111111111111"B

กลุ่มคำตอบสำหรับ C

a) "1"B
b) "1"B << (PatLen - 1)
c) "1"B << (i - 1)
d) "1"B << PatLen
e) "1"B << i

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้

[คำอธิบายของฟังก์ชัน BitapMatch]

(1) ฟังก์ชันนี้รับค่า Text[] และ Pat[] และทำการค้นหาสตริงที่เหมือนกับ Pat[] จากลำดับสมาชิกที่น้อยที่สุดของ Text[] หากพบว่ามีส่วนที่เหมือนกันก็จะคืนค่าเป็นลำดับสมาชิก (element number) ของ Text[] ที่ตรงกับอักขระตัวแรกของสตริงที่เหมือนกัน แต่หากไม่พบส่วนที่เหมือนกัน จะคืนค่าเป็น -1

(2) ในตัวอย่างที่แสดงในรูปที่ 1 สตริงที่อยู่ระหว่าง Text[7] และ Text[12] เหมือนกับ Pat[] ดังนั้น จึงคืนค่า 7

(3) ตารางที่ 2 แสดงข้อกำหนดของอาร์กิวเมนต์และการคืนค่าของฟังก์ชัน BitapMatch

ตารางที่ 2 ข้อกำหนดของอาร์กิวเมนต์และการคืนค่าของฟังก์ชัน BitapMatch

อาร์กิวเมนต์ / การคืนค่า	ชนิดข้อมูล	อินพุต / เอาต์พุต	คำอธิบาย
Text[]	CHAR	Input	อาร์เรย์หนึ่งมิติที่จัดเก็บสตริงเป้าหมาย
Pat[]	CHAR	Input	อาร์เรย์หนึ่งมิติที่จัดเก็บสตริงค้นหา
Return value	INT	Output	ถ้าพบสตริงค้นหาในสตริงเป้าหมาย ให้คืนค่าเป็นลำดับสมาชิกของ Text[] ที่ตรงกับอักขระตัวแรกของสตริงที่เหมือนกัน แต่หากไม่พบ ให้คืนค่า -1

[โปรแกรม 2]

```
FUNCTION: BitapMatch(CHAR: Text[], CHAR: Pat[]) {
   16-bit Binary: Goal, Status, Mask[26]
   INT: i, TextLen, PatLen

   TextLen ← number of characters in Text[];
   PatLen ← GenerateBitMask(Pat[], Mask[]);
   Status ← "0"B;
   Goal ← "1"B << (PatLen - 1);
   FOR (i ← 1; i ≤ TextLen; i ← i + 1) {
      Status ← (Status << 1) | "1"B;                /*** α ***/
      Status ← Status & Mask[Index(Text[i])];       /*** β ***/
      IF ((Status & Goal) ≠ "0"B) {
         return (i - PatLen + 1)
       }
   }
   return (-1)
}
```

ฟังก์ชัน BitapMatch ถูกเรียกใช้ กำหนดให้ค่าต่าง ๆ ที่แสดงในรูปที่ 1 ถูกจัดเก็บอยู่ใน Text[] และ Pat[]
รูปที่ 3 แสดงการเปลี่ยนแปลงค่าของ i, Mask[Index(Text[i])] และ Status โดยทันทีหลังจากการทำงานของบรรทัดที่ระบุคอมเมนต์ /*** β ***/ (จากนี้ไปจะเรียก บรรทัด β) ในโปรแกรม 2

ตัวอย่างเช่น ค่าของ Status ทันทีที่ประมวลผลบรรทัด β แล้วเสร็จ เมื่อ i เป็น 1 คือ "1"B ดังนั้น ค่าของ Status ทันทีที่ประมวลผลบรรทัด α แล้วเสร็จ เมื่อ i เป็น 2 คือ "11"B ซึ่งก็คือ ผลรวมทางตรรกะในระดับบิต (bitwise logical sum) ของ "10"B (ค่าที่ได้จากการเลื่อนบิตของ "1"B ไปทางซ้ายทางตรรกะ 1 บิต) กับ "1"B จากนั้น เมื่อค่าของ Mask[Index(Text[2])] คือ "10101"B แล้ว ค่าของ "10101"B ทันทีที่ประมวลผลบรรทัด β แล้วเสร็จ เมื่อ i เป็น 2 คือ [D]

ในลักษณะเดียวกันนั้น เมื่อค่าของ Status ทันทีที่ประมวลผลบรรทัด β แล้วเสร็จ เมื่อ i เป็น 8 คือ "10"B แล้ว ค่าของ Mask[Index(Text[9])] ทันทีที่ประมวลผลบรรทัด α แล้วเสร็จ เมื่อ i เป็น 9 คือ [E] ดังนั้น ค่าของ Status ทันทีที่ประมวลผลบรรทัด β แล้วเสร็จ คือ [F]

ตารางที่ 3 การเปลี่ยนแปลงค่าของ i, Mask[Index(Text[i])] และ Status ทันทีที่ประมวลผลบรรทัด β ในโปรแกรม 2 แล้วเสร็จ สำหรับข้อมูลดังแสดงในรูปที่ 1

i	1	2	…	8	9	…
Mask[Index(Text[i])]	"10101"B	"10101"B	…	"10"B	[E]	…
Status	"1"B	[D]	…	"10"B	[F]	…

กลุ่มคำตอบสำหรับ D ถึง F

a) "0"B b) "1"B c) "10"B d) "11"B
e) "100"B f) "101"B g) "10101"B

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในคำอธิบายต่อไปนี้ ซึ่งเกี่ยวกับส่วนขยายของฟังก์ชัน GenerateBitMask ในที่นี้ ให้ถือว่า [B] ในโปรแกรม 3 เป็นคำตอบที่ถูกต้องสำหรับ [B] ในคำถามย่อย 1

เพื่อให้สามารถใช้คุณสมบัติของนิพจน์ปรกติ (regular expression) ดังแสดงในตารางที่ 4 ในสตริงค้นหาได้ ฟังก์ชัน GenerateBitMask จึงถูกต่อยอดเพื่อสร้างฟังก์ชัน GenerateBitMaskRegex

ตารางที่ 4 นิพจน์ปรกติ (Regular Expression)

สัญลักษณ์	คำอธิบาย
[]	ใช้แทนอักขระที่ตรงกับอักขระต่าง ๆ ที่ระบุไว้ใน [] ตัวอย่างเช่น "A[XYZ]B" สามารถใช้แทน "AXB", "AYB" และ "AZB" ได้

[โปรแกรม 3]

```
FUNCTION: GenerateBitMaskRegex(CHAR: Pat[], 16-bit Binary: Mask[]) {
   INT: i, OriginalPatLen, PatLen, Mode

   OriginalPatLen ← number of characters in Pat[];
   PatLen ← 0;
   Mode ← 0;
   FOR (i ← 1; i ≤ 26; i ← i + 1) {
      Mask[i] ← [ B ];     /* Initialize */
   }
   FOR (i ← 1; i ≤ OriginalPatLen; i ← i + 1) {
      IF (Pat[i] = "[") {
         Mode ← 1;
         PatLen ← PatLen + 1;
      }
      ELSE {
         IF (Pat[i] = "]") {
            Mode ← 0;
         }
         ELSE {
            IF (Mode = 0) {
               PatLen ← PatLen + 1;
            }
            Mask[Index(Pat[i])] ← Mask[Index(Pat[i])] |
                                  ("1"B << (PatLen - 1));
         }
      }
   }
   return (PatLen);
}
```

กำหนดให้ “AC[BA]A[ABC]A” ถูกจัดเก็บอยู่ใน Pat[] และฟังก์ชัน GenerateBitMaskRegex ถูกเรียกใช้ ในกรณีนี้ บิตมาสก์ Mask[1] ที่ตรงกับ “A” คือ "111101"B และค่าที่คืนมาจากฟังก์ชัน GenerateBitMaskRegex คือ [G] นอกจากนี้ การใช้ [] ซ้อนกันในสตริงที่จัดเก็บใน Pat[] ยังไม่สามารถทำได้ แต่ถ้า “AC[B[AB]AC]A” ซึ่งเป็นรูปแบบที่ไม่ถูกต้อง ถูกนำไปใส่ใน Pat[] แล้วเรียกใช้ฟังก์ชัน GenerateBitMaskRegex จะทำให้ Mask[1] กลายเป็น [H]

กลุ่มคำตอบสำหรับ G

a) 4 b) 6 c) 9 d) 13

กลุ่มคำตอบสำหรับ H

a) "1001101"B
b) "1010100001"B
c) "1011001"B
d) "101111"B
e) "110011"B
f) "111101"B

สำหรับข้อสอบข้อที่ Q7 และ Q8 ให้เลือกทำหนึ่งในสองข้อ
จากนั้น ให้ระบายทับในวงกลม Ⓢ ในกระดาษคำตอบสำหรับข้อที่เลือกทำ
หากเลือกทั้งสองข้อ จะตรวจให้คะแนนเฉพาะข้อแรกเท่านั้น

Q7. **อ่านคำอธิบายโปรแกรมภาษาซีต่อไปนี้ แล้วตอบคำถามย่อย 1 ถึง 3**

กีฬาต่อสู้มักมีการกำหนดระดับเพื่อหลีกเลี่ยงการจับคู่ที่ไม่เหมาะสมและเพื่อความปลอดภัย ตัวอย่างเช่น หนึ่งในกฎต่าง ๆ ของกีฬามวยปล้ำชายฟรีสไตล์ที่แบ่งเป็น 6 ระดับตามน้ำหนัก: 57, 65, 74, 86, 97 และ 125 กิโลกรัม (กก.) โดยนักมวยปล้ำสามารถร่วมแข่งขันได้ในระดับน้ำหนักเดียวกันหรือมากกว่าน้ำหนักของตนได้ตามกฎที่กำหนดไว้ ในคำถามนี้ กำหนดให้นักมวยปล้ำแต่ละรายเข้าแข่งขันในระดับน้ำหนักต่ำที่สุดที่เป็นไปได้ ตัวอย่างเช่น นักมวยปล้ำที่มีน้ำหนัก 58 กก. จะได้รับอนุญาตให้เข้าแข่งขันในระดับใดก็ได้ ยกเว้นระดับ 57 กก. แต่เขาควรเข้าแข่งขันในระดับ 65 กก.

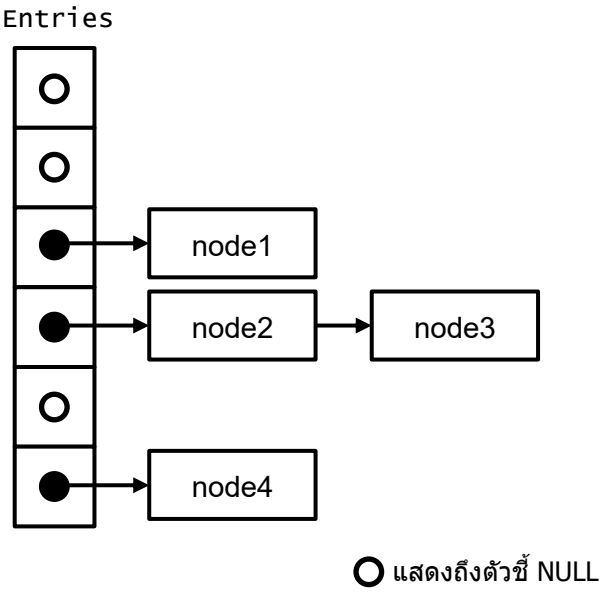
[คำอธิบายโปรแกรม]
โปรแกรมนี้จะรับคำขอเข้าร่วมการแข่งขันจากนักมวยปล้ำ จัดระดับนักมวยปล้ำตามน้ำหนักของนักมวยปล้ำรายนั้น ๆ และแสดงผลคำขอทั้งหมดที่ได้รับการตอบรับ

(1) โปรแกรมจะรับค่าจำนวนของนักมวยปล้ำ จากนั้นจะรับชื่อและน้ำหนักของนักมวยปล้ำจาก `stdin` โดยจำนวนและน้ำหนักของนักมวยปล้ำเป็นตัวเลขฐานสิบจำนวนไม่เกิน 3 หลัก และชื่อจะประกอบไปด้วยอักขระตัวอักษรยาวไม่เกิน 50 ตัวอักษร กำหนดให้ข้อมูลนำเข้าทั้งหมดเป็นไปตามข้อกำหนดเหล่านี้ ตารางที่ 1 แสดงตัวอย่างของข้อมูลนักมวยปล้ำ 4 คน

ตารางที่ 1 ตัวอย่างของข้อมูลนักมวยปล้ำ 4 คน

ชื่อ	น้ำหนัก (กก.)
Anthony	85
Bob	68
Charles	77
Daniel	103

(2) โครงสร้างข้อมูลสำหรับจัดเก็บข้อมูลของนักมวยปล้ำเป็นอาร์เรย์ของลิงก์ลิสต์ชื่อ `Entries` ในที่นี้ `Entries[0]`, `Entries[1]`, ..., `Entries[5]` จะสอดคล้องกับระดับน้ำหนัก 57, 65, ..., 125 กก. ตามลำดับ สมาชิก (element) แต่ละตัวของ `Entries` ชี้ไปยังลิงก์ลิสต์ที่เก็บข้อมูลของนักมวยปล้ำทั้งหมดที่เข้าแข่งขันในระดับน้ำหนักนั้น ๆ หรือ `NULL` หากไม่มีนักมวยปล้ำรายใดเข้าแข่งขันในระดับน้ำหนักนั้น
รูปที่ 1 แสดงโครงสร้างข้อมูลสำหรับจัดเก็บข้อมูลของนักมวยปล้ำ



รูปที่ 1 โครงสร้างข้อมูลสำหรับจัดเก็บข้อมูลของนักมวยปล้ำ

(3) โหนด (node) ของลิงก์ลิสต์ใช้แทนข้อมูลของนักมวยปล้ำแต่ละราย ซึ่งเป็น `struct` ที่มีชื่อว่า `Wrestler` และมีสามฟิลด์: `name`, `weight` และ `next`

```
struct Wrestler {
    char name[51];
    int weight;
    struct Wrestler *next;
};
typedef struct Wrestler Wrestler_t;
```

สองฟิลด์แรกจัดเก็บชื่อและน้ำหนักของนักมวยปล้ำ ฟิลด์ `next` เป็นตัวชี้ (pointer) ไปยังโหนดที่จัดเก็บข้อมูลของนักมวยปล้ำรายถัดไป หรือเป็น `NULL` หากไม่มีนักมวยปล้ำรายอื่นอีกในระดับน้ำหนักนั้น

(4) โปรแกรมทำงานดังต่อไปนี้:

ขั้นที่ 1 กำหนดค่าตั้งต้นให้ `Entries` และรับจำนวนนักมวยปล้ำจาก `stdin`

ขั้นที่ 2 ทำซ้ำตั้งแต่ (i) ถึง (iii) สำหรับนักมวยปล้ำแต่ละราย

(i) รับชื่อและน้ำหนักของนักมวยปล้ำจาก `stdin`

(ii) สร้างโหนดเพื่อจัดเก็บข้อมูลของนักมวยปล้ำ

(iii) ค้นหาลิงก์ลิสต์ที่เหมาะสมสำหรับโหนดที่สร้างขึ้นใน (ii) จาก `Entries` แล้วเพิ่มโหนดลงไปด้านท้ายของลิสต์ หากไม่พบระดับน้ำหนักที่ระบุ ให้รับข้อมูลน้ำหนักสำหรับโหนดนั้นใหม่ แล้วจึงค้นหาลิงก์ลิสต์ที่เหมาะสมอีกครั้ง

ขั้นที่ 3 แสดงข้อมูลในลิงก์ลิสต์ หากระดับน้ำหนักใดไม่มีโหนด ให้แสดง “`<empty>`” แทนข้อมูลของนักมวยปล้ำ

(5) เมื่อเรียกใช้โปรแกรมโดยป้อนข้อมูลตัวอย่างในตารางที่ 1 โปรแกรมจะแสดงผลลัพธ์ดังนี้:

```
How many wrestlers want to participate: 4
Enter the name of the wrestler: Anthony
Enter the weight for Anthony: 85
Enter the name of the wrestler: Bob
Enter the weight for Bob: 68
Enter the name of the wrestler: Charles
Enter the weight for Charles: 77
Enter the name of the wrestler: Daniel
Enter the weight for Daniel: 103
Result:
 57 kg class: <empty>
 65 kg class: <empty>
 74 kg class: [Bob 68]
 86 kg class: [Anthony 85] [Charles 77]
 97 kg class: <empty>
125 kg class: [Daniel 103]
```

(6) ตารางที่ 2 แสดงฟังก์ชันต่าง ๆ ที่ถูกใช้ในโปรแกรม

ตารางที่ 2 ฟังก์ชันต่าง ๆ ที่ถูกใช้ในโปรแกรม

ชื่อฟังก์ชัน	คำอธิบาย
`void Initialize(void)`	กำหนดค่าตั้งต้นให้ Entries
`Wrestler_t *AcceptEntry(void)`	รับข้อมูลการเข้าแข่งขันจากนักมวยปล้ำ
`int FindWeightClass(int)`	ค้นหาระดับน้ำหนักที่เหมาะสมกับน้ำหนักของนักมวยปล้ำที่ระบุ หากไม่พบ ให้คืนค่า -1
`void Insert(Wrestler_t *, int)`	เพิ่มโหนดที่ระบุลงในลิงก์ลิสต์ตามดัชนี (index) ที่กำหนด
`void Display(void)`	แสดงข้อมูลของนักมวยปล้ำทั้งหมดที่ลงทะเบียนอยู่ในลิงก์ลิสต์
`void Deallocate(void)`	ล้างข้อมูลในความจำที่ได้จองไว้แบบไดนามิก
`void *malloc(size_t)`	(ฟังก์ชันไลบรารีมาตรฐาน) จัดสรรหน่วยความจำตามขนาดที่ระบุแบบไดนามิก และคืนค่าเป็นตัวชี้ที่ชี้ไปยังหน่วยความจำนั้น
`Void free(void *)`	(ฟังก์ชันไลบรารีมาตรฐาน) คืนหน่วยความจำที่ได้รับการจัดสรรไว้ด้วย malloc

(7) ตารางที่ 3 แสดงตัวแปรต่าง ๆ ที่ถูกใช้ในโปรแกรม

ตารางที่ 3 ตัวแปรต่าง ๆ ที่ถูกใช้ในโปรแกรม

ชื่อตัวแปร	คำอธิบาย
`Wrestler_t *Entries[NUM_CLASSES]`	อาร์เรย์ของลิงก์ลิสต์ซึ่งสมาชิก (element) แต่ละตัวเป็นตัวแทนของระดับน้ำหนัก 57, 65, 74, 86, 97 และ 125 กก. ตามลำดับ
`const int UpperLimits[NUM_CLASSES]`	อาร์เรย์ที่ใช้จัดเก็บขอบจำกัดบน (upper limit) สำหรับแต่ละระดับน้ำหนัก

[โปรแกรม]

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NUM_CLASSES 6

struct Wrestler {
    char name[51];
    int weight;
    struct Wrestler *next;
};
typedef struct Wrestler Wrestler_t;

Wrestler_t *Entries[NUM_CLASSES];
const int UpperLimits[NUM_CLASSES] = {57, 65, 74, 86, 97, 125};

void Initialize(void);
Wrestler_t *AcceptEntry(void);
int FindWeightClass(int);
void Insert(Wrestler_t *, int);
void Display(void);
void Deallocate(void);

int main() {
    Wrestler_t *wrestler;
    int n, i, index;

    Initialize();
    printf("How many wrestlers want to participate: ");
    scanf("%3d", &n);
```

```
    for (i = 0; i < n; i++) {
        wrestler = AcceptEntry();
        index = FindWeightClass(wrestler->weight);
        while (    A    ) {
            printf("%s's weight exceeds %d kg. Re-enter the weight: ",
                   wrestler->name, UpperLimits[NUM_CLASSES - 1]);
            scanf("%3d", &wrestler->weight);
            index = FindWeightClass(wrestler->weight);
        }
        Insert(wrestler, index);
    }
    Display();
    Deallocate();
    return 0;
}

void Initialize(void) {
    int i;
    for (i = 0; i < NUM_CLASSES; i++) {
        Entries[i] = NULL;
    }
}

Wrestler_t *AcceptEntry(void) {
    Wrestler_t *wrestler = (Wrestler_t *)malloc(    B    );
    printf("Enter the name of the wrestler: ");
    scanf("%50s", wrestler->name);
    printf("Enter the weight for %s: ", wrestler->name);
    scanf("%3d", &wrestler->weight);
    wrestler->next = NULL;
    return wrestler;
}

int FindWeightClass(int weight) {
    int i;
    for (i = 0; i < NUM_CLASSES; i++) {
        if (    C    ) {
            return i;
        }
    }
    return -1;
}
```

```
void Insert(Wrestler_t *wrestler, int i) {
    Wrestler_t *curr = Entries[i];
    if (curr == NULL) {                          /*** α ***/
        [ D ] = wrestler;
    } else {
        while (curr->next != NULL) {             /*** β ***/
            curr = curr->next;
        }
        [ E ] = wrestler;                        /*** γ ***/
    }
}

void Display(void) {
    Wrestler_t *curr;
    int i;
    printf("Result:\n");
    for (i = 0; i < NUM_CLASSES; i++) {
        curr = Entries[i];
        printf("%3d kg class:", UpperLimits[i]);
        if (curr == NULL) {
            printf("<empty>\n");
            continue;
        }
        while (curr != NULL) {
            printf("[%s %d] ", curr->name, curr->weight);
            curr = curr->next;
        }
        printf("\n");
    }
}

void Deallocate(void) {
    int i;
    Wrestler_t [ F ];
    for (i = 0; i < NUM_CLASSES; i++) {
        curr = Entries[i];
        while (curr != NULL) {
            next = curr->next;
            free(curr);
            curr = next;
        }
    }
}
```

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในโปรแกรมข้างต้น

กลุ่มคำตอบสำหรับ A

a) `index != -1`
b) `index != 0`
c) `index == -1`
d) `index == 0`

กลุ่มคำตอบสำหรับ B

a) `sizeof(Wrestler_t *)`
b) `sizeof(Wrestler_t)`
c) `Wrestler_t *`
d) `Wrestler_t`

กลุ่มคำตอบสำหรับ C

a) `UpperLimits[i] < weight`
b) `UpperLimits[i] <= weight`
c) `UpperLimits[i] > weight`
d) `UpperLimits[i] >= weight`

กลุ่มคำตอบสำหรับ D และ E

a) `Entries[i + 1]`
b) `Entries[i - 1]`
c) `Entries[i]`
d) `curr`
e) `curr->next`
f) `curr->next->next`

กลุ่มคำตอบสำหรับ F

a) `&curr, &next`
b) `&curr, next`
c) `*curr, *next`
d) `*curr, next`
e) `curr, next`

คำถามย่อย 2

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [G] ในคำอธิบายต่อไปนี้

โปรแกรมถูกแก้ไขดังต่อไปนี้ เพื่อเรียงลำดับโหนดต่าง ๆ ในลิงก์ลิสต์ตามน้ำหนักจากน้อยไปหามาก แทนการเรียงตามลำดับการลงทะเบียน

(1) แทนที่บรรทัดที่ระบุด้วย `/*** α ***/` ด้วยสองบรรทัดต่อไปนี้

```
if (curr == NULL || curr->weight >= wrestler->weight) {
    wrestler->next = curr;        /*** δ ***/
```

(2) แทนที่บรรทัดที่ระบุด้วย /*** β ***/ ด้วยสองบรรทัดต่อไปนี้

```
while ((curr->next != NULL) &&
       ( G )) {
```

(3) แทรกบรรทัดต่อไปนี้เข้าไปเป็นบรรทัดก่อนหน้าของบรรทัดที่ระบุด้วย /*** γ ***/

```
wrestler->next = curr->next;
```

กลุ่มคำตอบสำหรับ G

a) `curr->next->weight < wrestler->weight`
b) `curr->next->weight >= wrestler->weight`
c) `curr->weight < wrestler->weight`
d) `curr->weight >= wrestler->weight`

คำถามย่อย 3

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [H] ในคำอธิบายต่อไปนี้

เมื่อโปรแกรมที่ได้แก้ไขตามคำถามย่อย 2 ถูกเรียกใช้ และได้รับข้อมูลนักมวยปล้ำ 5 คนดังแสดงในตาราง 4 แล้ว บรรทัดที่ระบุด้วย /*** δ ***/ จะถูกเรียกใช้ [H] ครั้ง

ตารางที่ 4 ข้อมูลของนักมวยปล้ำห้าคน

ชื่อ	น้ำหนัก (กก.)
Eric	88
Faraday	72
Gregory	91
Harold	97
Issacs	68

กลุ่มคำตอบสำหรับ H

a) 2 b) 3 c) 4 d) 5

สำหรับข้อสอบข้อที่ **Q7** และ **Q8 ให้เลือกทำเพียงหนึ่ง**ในสองข้อ

Q8. อ่านคำอธิบายโปรแกรมภาษาจาวาต่อไปนี้ แล้วตอบคำถามย่อย 1 และ 2
คำอธิบาย API ที่ถูกใช้ในโปรแกรมภาษาจาวา สามารถดูได้จากด้านท้ายของข้อสอบฉบับนี้

[คำอธิบายโปรแกรม]
โปรแกรมนี้มีวัตถุประสงค์เพื่อจัดการ "สิ่งที่ต้องทำ" (จากนี้จะเรียกว่า ToDo)

(1) คลาส `ToDo` ใช้แทน ToDo หนึ่งรายการ หัวข้อเรื่อง (subject) กำหนดการ (deadline) และระดับความสำคัญ (priority) ถูกกำหนดโดยคอนสตรัคเตอร์ กำหนดการเป็นสตริงอักขระที่อาจประกอบไปด้วย 8 อักขระตัวเลขที่ใช้แทน ปี, เดือน, วัน หรือประกอบไปด้วย 12 อักขระตัวเลขที่ใช้แทน ปี, เดือน, วัน, ชั่วโมง และ นาที (จากนี้จะเรียก วันและเวลา) ตัวอย่างเช่น 22 ตุลาคม 2023 จะถูกแทนด้วยสตริง "20231022" และ 22 ตุลาคม 2023, 1:00 p.m. จะถูกแทนด้วยสตริง "202310221300" ในที่นี้ กำหนดให้ไม่มีข้อผิดพลาดใด ๆ ในการกำหนดวันและเวลา
คลาสนี้มีเมธอดที่ใช้เพื่อรับหัวข้อเรื่อง, กำหนดการ และระดับความสำคัญ มีเมธอดเพื่อกำหนด (set) และรับ (get) ค่าสถานะ (state) และฟิลด์ `id` เพื่อใช้ระบุ ToDo นั้น ๆ
`Priority` เป็น enumeration ที่ใช้แทนระดับความสำคัญของ ToDo โดยมีค่าเป็น `LOW` (ต่ำ), `MIDDLE` (กลาง) และ `HIGH` (สูง) เรียงลำดับความสำคัญจากน้อยไปหามาก
`State` เป็น enumeration ที่ใช้แทนสถานะของ ToDo โดยมีค่าเป็น `NOT_YET_STARTED` (ยังไม่เริ่ม), `STARTED` (เริ่มแล้ว) และ `DONE` (เสร็จสิ้นแล้ว)

(2) คลาส `ToDoList` เก็บรายการของ ToDo
คลาสนี้ช่วยรับรองได้ว่าในรายการจะไม่มี ToDo ที่มีฟิลด์ `id` เป็นค่าเดียวกัน
คลาสนี้มีเมธอด `add` เพื่อเพิ่ม ToDo หนึ่งรายการ เมธอด `update` เพื่ออัพเดต ToDo และเมธอด `select` เพื่อคืนค่ารายการของ ToDo ที่ตรงกับเงื่อนไขที่กำหนด
หาก ToDo ที่อยู่ในรายการอยู่แล้ว ถูกใช้เป็นอาร์กิวเมนต์ของเมธอด `add` หรือหาก ToDo ที่ไม่ได้อยู่ในรายการ ถูกใช้เป็นอาร์กิวเมนต์ของเมธอด `update` เมธอดเหล่านี้ก็จะไม่ดำเนินการใด ๆ
ในเมธอด `select` นั้น อาจมีเงื่อนไขตั้งแต่ศูนย์เงื่อนไขหรือมากกว่าถูกกำหนดเป็นอาร์กิวเมนต์ หากได้กำหนดเงื่อนไขตั้งแต่หนึ่งเงื่อนไขขึ้นไป เมธอดนี้จะคืนค่าเป็นรายการของ ToDo ที่ตรงกับเงื่อนไขทั้งหมดนั้น แต่หากไม่ได้ระบุเงื่อนไขใด ๆ เมธอดนี้จะคืนค่าเป็นรายการทั้งหมดที่มี

(3) อินเทอร์เฟส `Condition` เป็นอินเทอร์เฟสเชิงฟังก์ชัน (functional interface) ที่ใช้แทนเงื่อนไขต่าง ๆ เพื่อเลือก ToDo ขึ้นมา เมธอด `test` จะคืนค่าเป็น `true` เมื่อเป็นไปตามเงื่อนไขนั้น

(4) คลาส `ToDoListTester` เป็นคลาสที่ถูกใช้เพื่อการทดสอบ

[โปรแกรม 1]

```
import java.util.UUID;

public class ToDo {
   public enum Priority { LOW, MIDDLE, HIGH }
   public enum State { NOT_YET_STARTED, STARTED, DONE }

   // Regular expression that matches a string of 8 or 12 numeric characters
   private static final String DEADLINE_PATTERN = "\\d{8}|\\d{12}";

   private final String id;
   private String subject;
   private String deadline;
   private Priority priority;
   private State state;

   private ToDo(String subject, String deadline, Priority priority,
                String id, State state) {
      if (!deadline.matches(DEADLINE_PATTERN)) {
         throw new IllegalArgumentException();
      }
      this.subject = subject;
      this.deadline = deadline;
      this.priority = priority;
      this.id = id;
      this.state = state;
   }

   public ToDo(String subject, String deadline, Priority priority) {
      this(subject, deadline, priority,
           UUID.randomUUID().toString(), State.NOT_YET_STARTED);
   }

   public ToDo(ToDo todo) {
      this(todo.subject, todo.deadline,
           todo.priority, todo.id, todo.state);
   }

   public String getSubject() { return subject; }
   public String getDeadline() { return deadline; }
   public Priority getPriority() { return priority; }
   public State getState() { return state; }
   public void setState(State state) { this.state = state; }
   public int hashCode() { return id.hashCode(); }
```

```
    public boolean equals(Object o) {
        return o instanceof ToDo &&  A  ;
    }

    public String toString() {
        return String.format(
                    "Subject: %s, Deadline: %s, Priority: %s, State: %s",
                    subject, deadline, priority, state);
    }
}
```

[โปรแกรม 2]

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class ToDoList {
    private List<ToDo> todoList = new ArrayList<>();

    public void add(ToDo todo) {
        if (  B  ) {
            todoList.add(new ToDo(todo));
        }
    }

    public void update(ToDo todo) {
        int index = todoList.indexOf(todo);
        if (index  C  ) {
            todoList.set(index, todo);
        }
    }

    public List<ToDo> select(Condition... conditions) {
        return todoList.stream().
            filter(t -> Stream.of(conditions).  D  ).
            // Generate a list from the stream
            collect(Collectors.toList());
    }
}
```

[โปรแกรม 3]

```
import java.util.function.Predicate;

public interface Condition [ E ] Predicate<ToDo> {}
```

[โปรแกรม 4]

```
public class ToDoListTester {
   public static void main(String[] args) {
      ToDoList list = new ToDoList();
      list.add(new ToDo("Send the e-mail",
                        "202310231500", ToDo.Priority.HIGH));
      list.add(new ToDo("Reserve a hotel room",
                        "20231025", ToDo.Priority.LOW));
      list.add(new ToDo("Purchase tickets",
                        "20231030", ToDo.Priority.MIDDLE));
      list.add(new ToDo("Create the report",
                        "20231028", ToDo.Priority.HIGH));
      list.add(new ToDo("Set up the meeting",
                        "202311201400", ToDo.Priority.HIGH));
      list.update(new ToDo("Purchase a PC",
                           "20231121", ToDo.Priority.HIGH));
      list.select().forEach(todo -> {
            todo.setState(ToDo.State.STARTED);
            list.update(todo);
         });
      Condition condition1 =
         todo -> todo.getDeadline().compareTo("20231101") < 0;
      Condition condition2 =
         todo -> todo.getPriority().equals(ToDo.Priority.HIGH);
      list.select(condition1, condition2).
         forEach(System.out::println);
   }
}
```

คำถามย่อย 1

จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในโปรแกรมข้างต้น

กลุ่มคำตอบสำหรับ A

a) `((ToDo) o).id.equals(id)`
b) `(ToDo) o.id.equals(id)`
c) `id.equals(id)`
d) `o.id.equals(id)`

กลุ่มคำตอบสำหรับ B

a) `!todoList.contains(todo)`
b) `!todoList.isEmpty()`
c) `todoList.contains(todo)`
d) `todoList.isEmpty()`

กลุ่มคำตอบสำหรับ C

a) `!= -1`
b) `< todoList.size()`
c) `== -1`
d) `>= todoList.size()`

กลุ่มคำตอบสำหรับ D

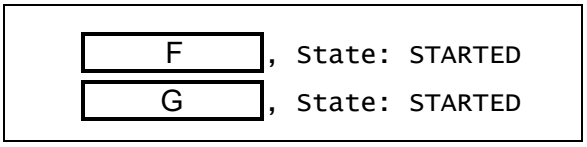
a) `allMatch(c -> c.test(t))`
b) `allMatch(c -> t.test(c))`
c) `anyMatch(c -> c.test(t))`
d) `anyMatch(c -> t.test(c))`

กลุ่มคำตอบสำหรับ E

a) `extends`
b) `implements`
c) `super`
d) `throws`

คำถามย่อย 2

รูปที่ 1 แสดงผลการประมวลผลของโปรแกรม 4 จากกลุ่มคำตอบด้านล่าง ให้เลือกคำตอบที่ถูกต้องเพื่อเติมลงในช่องว่าง [ ] แต่ละช่องในรูปที่ 1



รูปที่ 1 ผลลัพธ์การประมวลผลของโปรแกรม 4

กลุ่มคำตอบสำหรับ F และ G

a) `Subject: Create the report, Deadline: 20231028, Priority: HIGH`
b) `Subject: Purchase a PC, Deadline: 20231121, Priority: HIGH`
c) `Subject: Purchase tickets, Deadline: 20231030, Priority: MIDDLE`
d) `Subject: Reserve a hotel room, Deadline: 20231025, Priority: LOW`
e) `Subject: Send the e-mail, Deadline: 202310231500, Priority: HIGH`
f) `Subject: Set up the meeting, Deadline: 202311201400, Priority: HIGH`

■ **คำอธิบายของ API ที่ถูกใช้ในโปรแกรมภาษาจาวา**

`java.lang`
`public final class String`
คลาส `String` ใช้แสดงถึงสตริงอักขระ

เมธอด
`public boolean matches(String regex)` พิจารณาว่าสตริงนี้จับคู่ได้กับนิพจน์ปรกติ (regular expression) ที่ระบุหรือไม่ ตัวอย่างเช่น นิพจน์ปรกติ `"\\d{8}|\\d{12}"` จะจับคู่ได้กับสตริงของตัวเลขที่มีความยาว 8 หรือ 12 ตัวอักษร พารามิเตอร์: `regex` – นิพจน์ปรกติ การคืนค่า: `true` หากสตริงนี้จับคู่ได้กับนิพจน์ปรกติที่กำหนด ไม่เช่นนั้นคืนค่าเป็น `false`
`public int compareTo(String str)` เปรียบเทียบสตริงนี้กับสตริงที่กำหนดตามลำดับพจนานุกรม (lexicographically) พารามิเตอร์: `str` – สตริงที่กำหนด การคืนค่า: 0 หากสตริงนี้มีค่าเท่ากันกับสตริงที่กำหนด ค่าที่น้อยกว่า 0 หากสตริงนี้มีค่าตามลำดับพจนานุกรมน้อยกว่าสตริงที่กำหนด ค่าที่มากกว่า 0 หากสตริงนี้มีค่าตามลำดับพจนานุกรมมากกว่าสตริงที่กำหนด

`java.util`
`public final class UUID`
คลาส `UUID` ใช้แสดงถึงตัวระบุ universally unique identifier (UUID) ซึ่งเป็นค่าขนาด 128 บิต

เมธอด
`public static UUID randomUUID()` สุ่มสร้างตัวระบุ universally unique identifier ขึ้นมาหนึ่งจำนวน การคืนค่า: ตัวระบุ unique identifier ที่ถูกสุ่มสร้างขึ้นมา

java.util.function public interface Predicate<T> อินเทอร์เฟสเชิงฟังก์ชัน (functional interface) ที่ใช้แสดงถึงเพรดิเคตหรือภาคแสดง (ฟังก์ชันที่คืนค่าเป็นบูลีน) ของหนึ่งอาร์กิวเมนต์
เมธอด
public boolean test(T t) Evaluates this predicate on the given argument. **พารามิเตอร์:** t – อาร์กิวเมนต์ที่เป็นอินพุต **การคืนค่า:** true หากอาร์กิวเมนต์ที่เป็นอินพุตนั้นตรงกับเพรดิเคต ไม่เช่นนั้นคืนค่าเป็น false

java.util.stream public interface Stream<T> อินเทอร์เฟสที่รองรับการดำเนินการในรูปแบบฟังก์ชัน (functional-style operation) กับสตรีมหรือลำดับของเอลิเมนต์ (stream of element)
เมธอด
public boolean allMatch(Predicate<? super T> predicate) คืนค่าว่าเอลิเมนต์ทั้งหมดของสตรีมนี้ตรงกับเพรดิเคตที่ระบุหรือไม่ **พารามิเตอร์:** predicate – ตัวฟังก์ชัน **การคืนค่า:** true หากเอลิเมนต์ทั้งหมดของสตรีมนี้ตรงกับเพรดิเคตที่ระบุ ไม่เช่นนั้นคืนค่าเป็น false
public boolean anyMatch(Predicate<? super T> predicate) คืนค่าว่ามีเอลิเมนต์ใดของสตรีมนี้ตรงกับเพรดิเคตที่ระบุหรือไม่ **พารามิเตอร์:** predicate – ตัวฟังก์ชัน **การคืนค่า:** true หากมีเอลิเมนต์ใดของสตรีมนี้ตรงกับเพรดิเคตที่ระบุ ไม่เช่นนั้นคืนค่าเป็น false
public Stream filter(Predicate<? super T> predicate) คืนค่าสตรีมที่ประกอบไปด้วยเอลิเมนต์ของสตรีมนี้ที่ตรงกับเพรดิเคตที่ระบุ **พารามิเตอร์:** predicate – ตัวฟังก์ชัน **การคืนค่า:** สตรีมที่ประกอบไปด้วยเอลิเมนต์ที่ตรงกับเพรดิเคตที่ระบุ

Company names and product names appearing in the test questions are trademarks or registered trademarks of their respective companies. Note that the ® and ™ symbols are not used within the text.