Phiên bản
công bố có giới hạn

Tập 2

Tài liệu ôn thi FE

Tài liệu ôn thi cho kỳ thi kỹ sư Công nghệ thông tin cơ bản

Ôn tập cho phần thi buổi chiều

IPA
Information-Technology Promotion Agency, Japan

Tài liệu ôn thi FE Tập 2

Mục lục

ÔN TẬP PHẦN THI BUỔI CHIỀU

Ôn tập phần thi
Buổi chiều

Các câu hỏi kiểm tra buổi chiều thường nằm trong tám lĩnh vực sau đây: Phần cứng, phần mềm, cấu trúc dữ liệu và cơ sở dữ liệu, giao tiếp mạng, công nghệ xử lý thông tin, thuật toán, thiết kế chương trình, và phát triển chương trình. Ở đây cung cấp các câu hỏi và câu trả lời trong từng lĩnh vực. Toàn bộ các câu hỏi được là được sử dụng của bài thi các năm trước.

1 Phần cứng

[Phạm vi câu hỏi]
Biểu diễn số,
Biểu diễn ký tự,
Biểu diễn hình ảnh/âm thanh, Bộ xử lý,
Bộ nhớ, Thiết bị vào ra,
Sự thực thi,
Địa chỉ,
Xử lý vào ra,
Cấu hình hệ thống.

Câu hỏi 1

Q1. Đọc các mô tả sau về thực hiện một lệnh, và trả lời câu hỏi.

Một máy tính với bộ nhớ chính 65,536 words, trong đó 1 words bao gồm 16 bits. Nó có 4 thanh ghi đa năng (0 đến 3) và một thanh ghi chương trình ("PR" dưới đây). Định dạng của một lệnh (độ dài 2 words) như sau:

OP	R	X	I	D	adr

OP: Xác định mã lệnh 8 bits. Trong ví dụ này 3 mã lệnh được sử dụng là

- 20_{16}: Tải địa chỉ có hiệu lực vào thanh ghi đa năng được xác định bởi R
- 21_{16}: Tải nội dung từ được chỉ ra bởi địa chỉ có hiệu lực vào thanh ghi đa năng xác định bởi R
- FF_{16}: Kết thúc sự thực thi

R: Xác định số của thanh ghi đa năng (0 đến 3) bằng 2 bit.

X: Xác định số của thanh ghi chỉ số (1 đến 3) bằng 2 bit. Thanh ghi đa năng với số xác định được dùng như thanh ghi chỉ số. Tuy nhiên, nếu là thanh ghi số 0, không có sự thay đổi thanh ghi địa chỉ cơ sở nào được tạo

I: 1 bit đơn nhận giá trị "1" nếu định địa chỉ gián tiếp, ngược lại nhận giá trị "0"

D: 3 bit mở rộng luôn nhận giá trị 0

adr: Xác định một địa chỉ 16 bit.

Địa chỉ có hiệu lực của lệnh được tính như trong bảng sau.

Bảng. Mối quan hệ giữa X, I và địa chỉ có hiệu lực

X	I	Địa chỉ có hiệu lực
0	0	Adr
1 đến 3	0	adr + (X)
0	1	(adr)
1 đến 3	1	(adr + (X))

Chú ý ( ): Dấu ngoặc đơn được dùng để biểu thị nội dung được lưu trữ trong thanh ghi hoặc địa chỉ

Câu hỏi

Từ nhóm các câu trả lời dưới đây, chọn câu trả lời chính xác để điền vào chỗ trống ☐ trong các mô tả sau:

Khi nội dung của thanh ghi đa năng và bộ nhớ chính có giá trị như hình dưới đây (giá trị trong hệ 16), 0100_{16} được đặt cho PR và chương trình được thực thi. Trong ví dụ này lệnh tại địa chỉ 0100_{16} nạp nội dung 0113_{16} tại địa chỉ $011B_{16}$ (gạch dưới) cho thanh ghi đa năng 0.

Sau khi kết thúc thực thi, [A] được đặt cho thanh ghi đa năng 0, [B] cho thanh ghi đa năng 1, [C] cho thanh ghi đa năng 2 và [D] cho thanh ghi đa năng 3.

Thanh ghi đa năng 0: 0003 1: 0000 2: 0000 3: 0000

Thanh ghi chương trình PR: 0100

Bộ nhớ chính

Address	+ 0	+ 1	+ 2	+ 3	+ 4	+ 5	+ 6	+ 7
00F8	0000	0000	0000	0000	0000	0000	0000	0000
0100	2100	011B	20C0	0003	2170	0111	21B8	011A
0108	FF00	0000	0000	0000	0000	0000	0000	0000
0110	0000	0001	0002	0003	0004	0005	0006	0007
0118	0110	0111	0112	0113	0114	0115	0116	0117
0120	0118	0119	011A	011B	011C	011D	011E	011F

Hình. Giá trị của thanh ghi đa năng, PR và bộ nhớ chính

Nhóm câu trả lời:

a) 0000_{16} b) 0001_{16} c) 0002_{16}
d) 0003_{16} e) 0004_{16} f) 0005_{16}
g) 0006_{16} h) 0113_{16} i) 0115_{16}

Đáp án câu 1

Tìm giá trị của các thanh ghi đa năng trong lệnh thực thi

Câu trả lời đúng

A – h, B – e, C – f, D – d

Giải thích

Ngày nay, rất hiếm chương trình bằng ngôn ngữ assembly, có rất nhiều người gặp khó khăn trong việc tìm câu trả lời cho các câu hỏi về ngôn ngữ máy. Và vì thế rất nhiều câu hỏi dường như không có đầu mối như việc làm thế nào để đọc sơ đồ nội dung của bộ nhớ chính (được gọi là dump list)

Tuy nhiên bạn có thể tìm được câu trả lời chính xác nếu bạn hiểu nguyên tắc hoạt động của CPU. Lặp đi lặp lại việc nhận lệnh và thực hiện lệnh, và trong trường hợp này việc nhận lệnh với độ dài 2 từ (= 32 bit) tại một thời điểm và thực hiện nó, bắt đầu tại địa chỉ của thanh ghi chương trình (bộ đếm chương trình). Câu hỏi này có thể thuận lợi với những ai có hiểu biết về ngôn ngữ lập trình CASL, tuy nhiên câu hỏi và ví dụ đưa ra rất chi tiết vì thế bạn có thể lấy điểm bằng cách dò chương trình một cách cẩn thận, chú ý đến việc định địa chỉ gián tiếp

Có hai chế độ định địa chỉ - định địa chỉ chỉ số và định địa chỉ gián tiếp được dùng ở đây. Định địa chỉ chỉ số yêu cầu sự tính địa chỉ có hiệu lực bằng cách cộng nội dung của thanh ghi chỉ số với giá trị địa chỉ được chỉ định. Định địa chỉ gián tiếp yêu cầu lấy giá trị được lưu trong bộ nhớ chính được chỉ ra bởi trường địa chỉ trong lệnh là địa chỉ có hiệu lực.

Định dạng của một lệnh được chỉ ra dưới đây, bạn phải rất chú ý đến các thành phần R, X và I khi dò vết chương trình.

[Định dạng của một lệnh (Độ dài 2 từ nhớ)]

OP	R	X	I	D	Adr
8 bits	2	2	1	3	16 bits

- R (0 – 3 : số của thanh ghi đa năng)
- X (1 – 3 : số của thanh ghi chỉ số, 0 : không được thay đổi bởi thanh ghi chỉ số)
- I (1 : áp dụng định địa chỉ gián tiếp , 0 : định địa chỉ gián tiếp không được áp dụng)

Đầu tiên, địa chỉ bắt đầu của chương trình là 0100_{16}, địa chỉ mà được gán cho thanh ghi chương trình (PR), vì thế lệnh của chương trình được lưu trữ bắt đầu tại địa chỉ này. Nếu chúng ta gán nhãn mỗi lệnh 2 từ nhớ trong nội dung của bộ nhớ chính. Chúng ta có như sau:

[Nội dung của bộ nhớ chính] Giá trị số hệ 16

Address	+0	+1	+2	+3	+4	+5	+6	+7
00F8	(1) 0000	0000	(2) 0000	0000	(3) 0000	0000	(4) 0000	0000
0100	2100	011B	20C0	0003	2170	0111	21B8	011A
0108	(5) FF00	0000	0000	0000	0000	0000	0000	0000
0110	0000	0001	0002	0003	0004	0005	0006	0007
0118	0110	0111	0112	0113	0114	0115	0116	0117
0120	0118	0119	011A	011B	011C	011D	011E	011F

Chúng ta cùng kiểm tra nội dung của những lệnh này theo thứ tự. Để tính toán địa chỉ có hiệu lực, chúng ta cần tham chiếu đến bảng được cho trong câu hỏi (mối quan hệ X, I và địa chỉ có hiệu lực) và tính toán một cách cẩn thận.

(1) 2100 011B

- Đây là lệnh đầu tiên trong chương trình
- Biểu diễn bit của phần gạch chân 00: 0000 0000 → R : 00 (thanh ghi đa năng 0), X : 00, I : 0.
- Mã lệnh 21_{16} nội dung $(0113)_{16}$ của từ nhớ được chỉ ra bởi địa chỉ hiệu lực $(011B)_{16}$ được nạp vào thanh ghi đan năng 0, thanh ghi được chỉ ra bởi R
- Thanh ghi đa năng 0 0003_{16} → 0113_{16}

(2) 20C0 0003

- Biểu diễn bit của phần gạch chân C0: 1100 0000 → R: 11 (thanh ghi đa năng 3), X : 00, I : 0.
- Mã lệnh 20_{16} → Địa chỉ hiệu lực (0003_{16}) được nạp vào thanh ghi đa năng 3, thanh ghi được chỉ ra bởi R.
- Thanh ghi đa năng 3 0000_{16} → 0003_{16}

(3) 2170 0111

- Biểu diễn bit của phần gạch chân 70: 0111 0000 →R : 01(thanh ghi đa năng 1), X : 11, I : 0.
- Vì X:11 (tham khảo thanh ghi đa năng 3), xảy ra định địa chỉ, sử dụng thanh ghi đa năng 3 như thanh ghi chỉ số.
- Tính toán địa chỉ có hiệu lực
- adr + (X) = 0111_{16} + (thanh ghi đa năng3) = $0111_{16} + 0003_{16} = 0114_{16}$

  ↓

  Nội dung được lưu trữ trong thanh ghi đa năng 3
- Mã lệnh 21_{16} → Nội dung $(0004)_{16}$ của từ nhớ được chỉ ra bởi địa chỉ hiệu lực $(0114)_{16}$ được nạp vào thanh ghi đa năng 1 được chỉ ra bởi R
- Thanh ghi đa năng 1 0000_{16} → 0004_{16}

(4) 21B8 011A

- Biểu diễn bit của phần gạch chân B8: 1011 1000 → R : 10 (thanh ghi đa năng 2), X : 11, I : 1.
- Vì X:11 (tham chiếu đến thanh ghi đa năng 3), định địa chỉ xảy ra, Sử dụng thanh ghi đa năng 3 như thanh ghi chỉ số. Và vì I:1, định địa chỉ gián tiếp được thực hiện.
- Calculation of the effective address

(adr + (X)) = ($011A_{16}$ + (thanh ghi đa năng 3)) nội dung được lưu trữ trong thanh ghi đa năng 3
= ($011A_{16}$ + 0003_{16})
= ($011D_{16}$) ……nội dung được lưu trữ trong ô nhớ $011D_{16}$
= 0115_{16}

- Mã lệnh 21_{16} → Nội dung $(0005)_{16}$ của từ nhớ chỉ bởi địa chỉ hiệu lực $(0115)_{16}$ được nạp vào thanh ghi đa năng 2, thanh ghi được chỉ ra bởi R.
- Thanh ghi đa năng 2 0000_{16} → 0005_{16}

(5) FF00 0000

- Mã lệnh FF_{16} → Việc thực thi chương trình kết thúc.

Như đã thấy ở trên, **rất cần thiết rằng bạn phải dò vết kết quả thực thi của lệnh một cách cẩn thận; bằng cách làm như vậy bạn có thể tìm được câu trả lời chính xác.** Bảng sau đây tổng hợp sự thay đổi của giá trị thanh ghi trong quá trình thực thi lệnh.

Lệnh	A	B	C	D
	GR0	GR1	GR2	GR3
(Beginning)	0003	0000	0000	0000
(1) 2100 011B	0113	↓	↓	↓
(2) 20C0 0003	↓	↓	↓	0003
(3) 2170 0111	↓	0004	↓	↓
(4) 21B8 011A	↓	↓	0005	↓
(5) FF00 0000	↓	↓	↓	↓
(Câu trả lời)	h	E	f	d

2 Phần mềm

[Phạm vi câu hỏi]
Phần mềm hệ thống, phần mềm ứng dụng
Gói phần mềm, các hàm của hệ thống,
Ngôn ngữ lập trình,
Ngôn ngữ bộ xử lí, thực thi chương trình,…

Câu hỏi 1

Q1. Hãy đọc những mô tả sau về một bộ lập lịch, sau đó trả lời câu hỏi con.

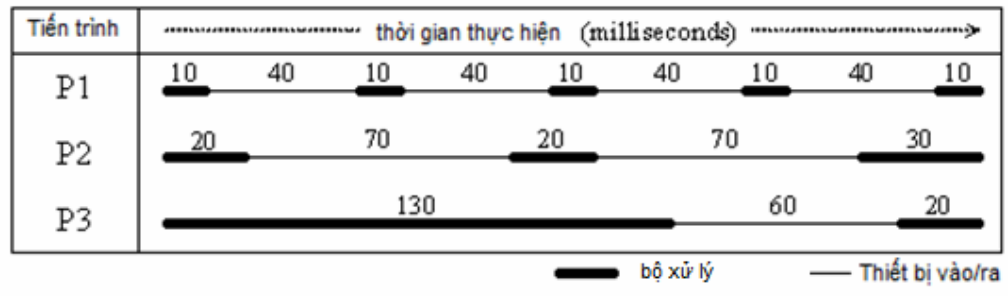
[Mô tả bộ lập lịch]

(1) Bộ lập lịch được thực hiện trên một bộ xử lí đơn sử dụng sự đa lập trình.
(2) Một lát thời gian (time slice) là 50 mili giây.
(3) Không có sự ưu tiên nào giữa các tiến trình, và sự lập lịch xoay vòng được sử dụng.
(4) Nếu có trao đổi vào/ra xuất hiện trong khi một tiến trình đang thực hiện, tiến trình tiếp theo sẽ được thực hiện. Cũng như thế, khi sự quay vòng tiến đến một tiến trình đang chờ đợi sự trao đổi vào/ra, mà tiến trình đó bị bỏ qua thì tiến trình tiếp theo được thực hiện.

Câu hỏi con

Từ nhóm câu trả lời ở dưới, hãy chọn các câu trả lời đúng để điền vào các ô trống trong mô tả sau.

Có ba tiến trình: P1, P2, P3. Hình vẽ chỉ ra trình tự trong đó bộ xử lí nào và sự trao đổi vào/ra nào được sử dụng, và khoảng thời gian chúng sử dụng khi mỗi tiến trình được thực hiện độc lập.



Hình vẽ. Trình tự và khoảng thời gian sử dụng bộ xử lí và thiết bị vào ra.

Bây giờ, giả sử rằng có ba tiến trình được thực thi dựa theo kiểu lập lịch vòng tròn theo thứ tự P1 → P2 → P3 → P1 → ... .Tổng chi phí gây bởi việc chuyển giữa các tiến trình được bỏ qua.

(1) Khi ba tiến trình đó sử dụng ba thiết bị vào ra khác nhau lần lượt là α, β và γ, tiến trình đầu tiên kết thúc là [A] và thời gian kể từ lúc bắt đầu tiến trình P1 cho đến lúc kết thúc tất cả các tiến trình là [B]

(2) Khi tiến trình P1 sử dụng thiết bị vào ra α còn tiến trình P2, P3 đều sử dụng thiết bị vào ra β (một tiến trình phải đợi cho đến khi các tiến trình khác bắt đầu rồi mới sử dụng thiết bị vào ra đã kết thúc trước đó), các tiến trình [C] phải dùng một lượng thời gian nhiều hơn của trường hợp (1) và lượng thời gian kéo dài thêm là [D] mili giây.

Nhóm câu trả lời cho A và C:

a) P1	b) P1 và P2	c) P1 và P3
d) P2	e) P2 và P3	f) P3

Nhóm câu trả lời cho B:

a) 250	b) 260	c) 270	d) 280
e) 290	f) 300	g) 310	h) 320

Nhóm câu trả lời cho D:

a) 10	b) 20	c) 30	d) 40
e) 50	f) 60	g) 70	h) 80

Đáp án câu 1

Lập lịch

Đáp án đúng

A – d, B – f, C – f, D – c

Giải thích

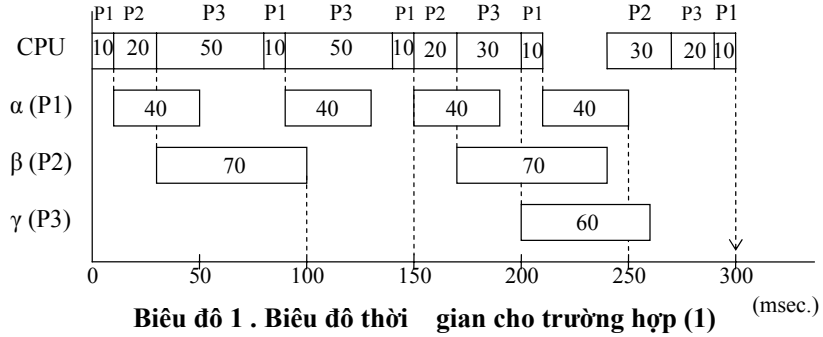
Câu hỏi này liên quan đến một bộ lập lịch thực hiện đa lập trình. Sự lập lịch xoay vòng được sử dụng, nó thực hiện chuyển giữa các tiến trình sau một khoảng thời gian xác định (gọi là thời gian lượng tử hoặc lát cắt thời gian) để phân bổ sự quyền thực hiện của CPU. Nếu một tiến trình vẫn đang chạy ở cuối của lượng tử, CPU sẽ chặn trước và trao quyền cho một tiến trình khác. Sau đây là một vài điểm quan trọng của bộ lập lịch:

(1) Sử dụng một bộ xử lí đơn.
(2) Một lát cắt thời gian là 50ms, nghĩa là thời gian lớn nhất sử dụng bộ xử lí là 50ms.
(3) Không có một sự ưu tiên nào giữa các tiến trình, và sự lập lịch xoay vòng được sử dụng.
(4) Nếu có một sự trao đổi vào/ra xuất hiện trong khi một tiến trình đang chạy, tiến trình tiếp theo sẽ được thực thi.
(5) Nếu vòng quay tiến đến một tiến trình đang đợi trao đổi vào/ra, mà tiến trình đó bị bỏ qua, thì tiến trình tiếp theo được thực thi.

Điểm mấu chốt để trả lời câu hỏi này là hiểu được nội dung của bộ lập lịch và tạo ra được một biểu đồ thời gian đúng và nhanh. Một điểm quan trọng để xem xét là: nếu sự xoay vòng tiến đến một tiến trình đang chờ một trao đổi vào ra và do đó tiến trình không được thực hiện, nó được bỏ qua, tuy nhiên, ngay cả khi tiến trình quay trở lại trạng thái thực hiện, nó cũng không được chạy ngay lập tức tại thời điểm đó. Tiến trình luôn được thực hiện trong một thứ tự nhất định. Điều này phải được ghi nhớ khi biểu đồ thời gian đã được chuẩn bị.

[Các ô trống A và B]

Thứ tự thực hiện của 3 tiến trình P1, P2, P3 là P1 → P2 → P3 → P1 →…. Dựa trên thứ tự và thời gian thực hiện của bộ xử lí và thiết bị vào ra được cho bởi sơ đồ trong câu hỏi, ta có thể tạo ra một sơ đồ thời gian như dưới đây:

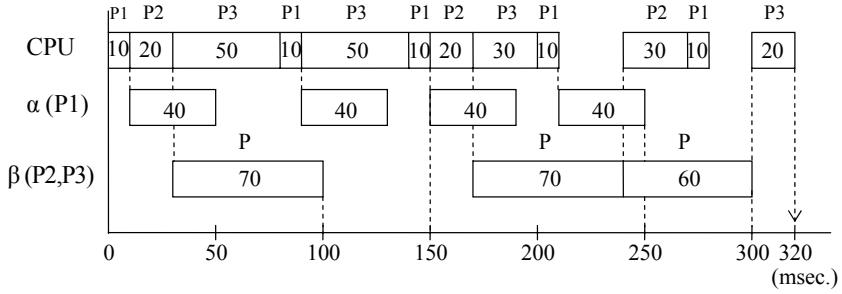


Biểu đồ 1 . Biểu đồ thời gian cho trường hợp (1)

Do đó, các tiến trình kết thúc theo thứ tự P2 (270) → P3 (290) → P1 (300), trong đó các số trong dấu ngoặc đơn chỉ ra thời điểm tại đó kết thúc. Do đó, tiến trình kết thúc đầu tiên là P2 (đáp án (d)), và để kết thúc tất cả các tiến trình cần 300ms (đáp án (d)).

[Các ô trống C và D]

Biểu đồ thời gian dưới đây là cho trường hợp trong đó P1 sử dụng thiết bị vào ra α trong khi P2 và P3 sử dụng cùng một thiết bị vào ra β.



Biểu đồ 2: Biểu đồ thời gian cho trường hợp (2)

Từ hình vẽ 2, ta có thể thấy rằng các tiến trình kết thúc theo thứ tự sau P2 (270) → P1 (280) → P3 (320). So sánh với trường hợp (1), tiến trình P3 chiếm nhiều thời gian nhất, và khoảng thời gian kéo dài thêm của tiến trình P3 là 320 – 290 = 30 milliseconds. Do đó, đáp án đúng cho C là (f), và đáp án đúng cho D là (c).

Tuy nhiên, cụm từ "khoảng thời gian kéo dài thêm" có thể được hiểu là toàn bộ khoảng thời gian so với trường hợp (1). Trong trường hợp đó, đáp án sẽ là 20 mili giây. Điều này sẽ làm thay đổi đáp án đúng cho D là "20 mili giây (b)". Do đó, có một sự mập mờ trong câu hỏi, nhưng ở đây chúng ta hiểu thuật ngữ "khoảng thời gian kéo dài thêm" theo nghĩa "khoảng thời gian kéo dài thêm cho tiến trình P3".

3 Cấu trúc dữ liệu và cơ sở dữ liệu

[Phạm vi của các câu hỏi]
Nền tảng về cấu trúc dữ liệu,
Các loại và đặc điểm của phương tiện lưu trữ
Các phương thức tổ chức file,
Các loại và các đặc điểm của cơ sở dữ liệu
Ngôn ngữ cơ sở dữ liệu,
Thao tác cơ sở dữ liệu sử dụng SQL,…

Câu hỏi 1

Q1. Đọc các mô tả dưới đây về cơ sở dữ liệu quan hệ. Sau đó trả lời các câu hỏi con 1 và 2.

Có hai bảng PRODUCT_TABLE và bảng WHOLESALER_TABLE được mô tả dưới đây. Tên các trường ở trên đầu bảng. Trường gạch chân được xác định là khóa chính.

PRODUCT_TABLE

PRODUCT_CODE	PRODUCT_NAME	UNIT_PRICE	WHOLESALER_CODE
S0001	Pencil	10	T0306
S0002	Eraser	50	T1020
⋮			

WHOLESALER_TABLE

WHOLESALER_CODE	WHOLESALER_NAME	PHONE_NO	REMARKS
T0001	Smith Company	03-1234-5678	
T0002	Regan Company	03-8765-4321	
⋮			

Câu hỏi con 1

Từ các nhóm câu trả lời phía dưới, chọn các câu trả lời đúng để điền vào các ô trống ☐ trong câu lệnh SQL để tạo ra bảng ORDER_TABLE.

ORDER_TABLE

ORDER_NO	ORDER_DATE	PRODUCT_CODE	QUANTITY
3001	2007-03-01	S0110	1000
3002	2007-03-02	S0054	2000
⋮			

```
CREATE TABLE ORDER_TABLE
    (ORDER_NO NUMERIC(4) NOT NULL,
    ORDER_DATE DATE NOT NULL,
    PRODUCT_CODE CHAR(5) NOT NULL,
    QUANTITY NUMERIC(5) NOT NULL,
    [A] KEY (ORDER_NO),
    [B] KEY (PRODUCT_CODE) [C] PRODUCT_TABLE)
```

Nhóm các câu trả lời:

a) FOREIGN b) INDEX c) MAIN
d) PRIMARY e) REFERENCES f) UNIQUE
g) USING

Câu hỏi con 2

Từ các nhóm câu trả lời phía dưới, chọn các câu trả lời đúng để điền vào các ô trống [ ] trong câu lệnh SQL để tạo ra khung nhìn Order_SLIP, có cùng số dòng như bảng Order, do vậy nó có thể được sử dụng để xác nhận nội dung của Order.

ORDER_SLIP

WHOLESALER_NAME	ORDER_DATE	PRODUCT_NAME	QUANTITY	AMOUNT
Thomson Company	2007-03-01	Notebook	100	10000
Holland Company	2007-03-02	Stick glue	200	20000
⋮				

```
CREATE VIEW ORDER_SLIP
  AS SELECT W.WHOLESALER_NAME, O.ORDER_DATE,
            P.PRODUCT_NAME, O.QUANTITY, [ D ]
  FROM ORDER_TABLE O, PRODUCT_TABLE P, WHOLESALER_TABLE W [ E ]
```

Nhóm các câu trả lời cho D:

a) `P.UNIT_PRICE * O.QUANTITY AS AMOUNT`
b) `SUM(P.UNIT_PRICE * O.QUANTITY) AS AMOUNT`
c) `AMOUNT IS P.UNIT_PRICE * O.QUANTITY`
d) `AMOUNT IS SUM(P.UNIT_PRICE * O.QUANTITY)`

Nhóm các câu trả lời cho E:

a)
```
IN O.PRODUCT_CODE = P.PRODUCT_CODE
 AND P.WHOLESALER_CODE = W.WHOLESALER_CODE
```
b)
```
IN O.PRODUCT_CODE = P.PRODUCT_CODE
 OR P.WHOLESALER_CODE = W.WHOLESALER_CODE
```
c)
```
WHERE O.PRODUCT_CODE = P.PRODUCT_CODE
 AND P.WHOLESALER_CODE = W.WHOLESALER_CODE
```
d)
```
WHERE O.PRODUCT_CODE = P.PRODUCT_CODE
 OR P.WHOLESALER_CODE = W.WHOLESALER_CODE
```

Đáp án câu 1

Câu lệnh SQL để định nghĩa bảng

Đáp án đúng

[Câu hỏi con 1] A – d, B – a, C – e

[Câu hỏi con 2] D – a, E – c

Giải thích

[Câu hỏi con 1]

Nếu bạn chưa bao giờ nhìn thấy một câu lệnh CREATE TABLE, lúc đầu bạn có thể mất tinh thần, nhưng bạn có thể trả lời chắc chắn câu hỏi nếu bạn xem xem xét nó một cách bình tĩnh. Vì trường Order_No trong bảng ORDER_TABLE được gạch chân, do đó đây là khóa chính. Vì vậy ô trống A sẽ định nghĩa khóa chính, câu trả lời là (d) PRIMARY. Bạn có thể nghĩa câu trả lời là (f) UNIQUE, nhưng không có một khai báo nào như vậy để đề cập đến một khóa unique (duy nhất), và nó cũng rất hiếm gặp trong SQL ngày nay. Ô trống B đề cập đến PRODUCT_CODE, khóa chính của bảng PRODUCT_TABLE, do vậy chúng ta có thể nhận ra đây là một ràng buộc tham chiếu. Đối với một ràng buộc tham chiếu định nghĩa một khóa ngoài, câu trả lời đúng là (a) FOREIGN. Với ô trống C, vì khóa chính của bảng PRODUCT_TABLE đã được tham chiếu, do vậy câu trả lời đúng là (e) REFERENCES.

Chú ý ràng các trường như ORDER_NO được khai báo là NOT NULL do vậy các giá trị của chúng không được phép là NULL.

[Câu hỏi con 2]

Một khung nhìn là một bảng trừu tượng được tạo ra tạm thời khi tham chiếu. Khung nhìn ORDER_SLIP có các trường bao gồm WHOLESALER_NAME từ bảng WHOLESALER, ORDER_DATE từ bảng ORDER_TABLE và PRODUCT_NAME từ bảng PRODUCT, do vậy nó không thể được định nghĩa trừ khi ba bảng này đã được kết nối. Khai báo FROM đề cập cụ thể 3 bảng này, WHERE định nghĩa các điều kiện kết nối. Ô trống E khai báo các điều kiện kết nối do vậy câu trả lời đúng là (c) với điều kiện AND. Ô trống D đề cập đến trường Amount trong khung nhìn do vậy câu trả lời đúng là (a) unit price*quantity = amount. Trong câu trả lời (a) có AS AMOUNT định nghĩa trường này trong khung nhìn là “AMOUNT”. Trừ khi tên của trường được khai báo với từ khóa AS, tên sẽ không thể được xác định bởi SQL chuẩn. Trong Oracle, một hệ quản trị cơ sở dữ liệu quan hệ thương mại, công thức tính toán bản thân nó đã trở thành tên khi thiếu đi một tên xác định, đây là đặc điểm riêng của Oracle.

4 Mạng truyền thông

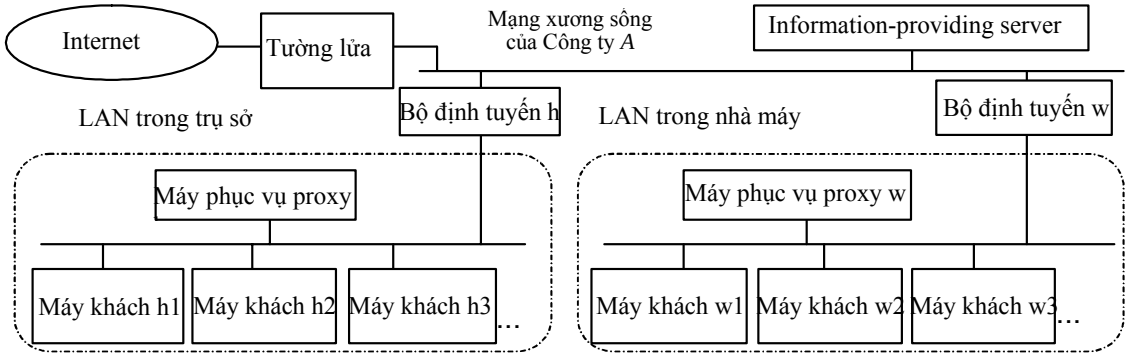
[Phạm vi câu hỏi]
Truyền dữ liệu
Điều khiển truyền, TCP/IP, LAN, WAN
Internet, Thư điện tử, Web,…

Câu hỏi 1

Q1. Đọc đoạn mô tả dưới đây về mạng truyền thông và trả lời 3 câu hỏi 1, 2 và 3.

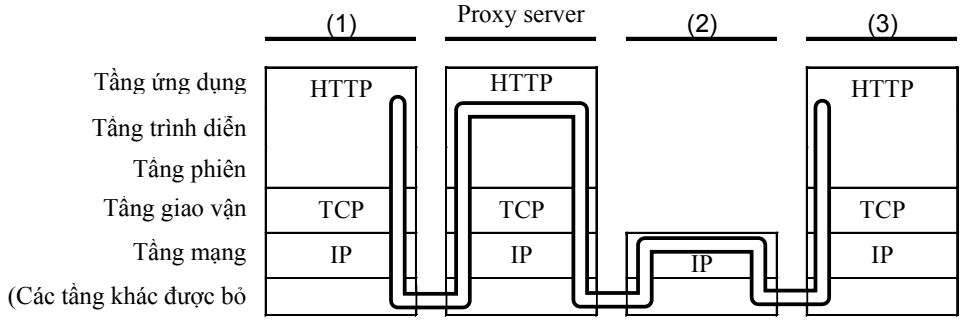
Công ty A dự định cung cấp dịch vụ thông tin nội bộ bằng cách thiết đặt một máy phục vụ cung cấp thông tin mà trụ sở và nhà máy có thể sử dụng một cách thường xuyên.
Như trong Hình 1, mạng LAN trong trụ sở và trong nhà máy được kết nối thông các bộ định tuyến tới mạng xương sống của công ty A, và mạng xương sống được kết nối thông qua tường lửa ra ngoài mạng Internet. TCP và IP được sử dụng lần lượt tại tầng giao vận và tầng mạng.

Tại trụ sở và nhà máy, mỗi máy khách sử dụng một chương trình hiển thị gọi là trình duyệt. Trình duyệt cho phép máy khách giao tiếp với vô số các máy phục vụ trên Internet với giao thức ở tầng ứng dụng gọi là HTTP. Mọi máy tính trong mạng đều được thiết lập theo cách giao tiếp với bên ngoài qua những bộ định tuyến luôn qua các máy phục vụ proxy.



Hình. 1 Mạng nội bộ tại công ty *A*

Hình 2 chỉ ra đường truyền thông cho các dịch vụ thông tin. Nếu HTTP, TCP, và IP được sử dụng, tầng ứng dụng nhận trực tiếp các dịch vụ từ tầng giao vận.



Hình. 2 Đường truyền thông cho các dịch vụ thông tin

Câu hỏi con 1

Trong hình 1, máy khách h1 phải được cho phép chuyển qua bộ định tuyến h nếu nó muốn giao tiếp với máy phục vụ cung cấp thông tin. Để được cho phép, một thiết lập từ nguồn tới đích phải chính xác. Từ nhóm câu trả lời dưới, chọn hai thiết lập chính xác từ nguồn tới đích.

Nhóm câu trả lời

	Nguồn	Đích
a)	Máy khách h1	Máy phục vụ cung cấp thông tin
b)	Máy khách h1	Máy phục vụ proxy h
c)	Máy khách h1	Tường lửa
d)	Máy phục vụ cung cấp thông tin	Máy khách h1
e)	Máy phục vụ cung cấp thông tin	máy phục vụ proxy h
f)	Máy phục vụ proxy h	Máy khách h1
g)	Máy phục vụ proxy h	Máy phục vụ cung cấp thông tin
h)	Máy phục vụ proxy h	Tường lửa
i)	Tường lửa	Máy khách h1
j)	Tường lửa	Máy phục vụ proxy h

Câu hỏi 2

Trong hình 2, một sự kết hợp nào đó của những thiết bị khác nhau được sử dụng trong mạng máy tính. Từ nhóm câu trả lời dưới đây, hãy chọn một kết hợp chính xác được sử dụng trong Hình 2.

Nhóm câu trả lời

	(1)	(2)	(3)
a)	Máy khách	Máy phục vụ cung cấp thông tin	Bộ định tuyến
b)	Máy khách	Bộ định tuyến	Máy phục vụ cung cấp thông tin
c)	Máy phục vụ cung cấp thông tin	Máy khách	Bộ định tuyến
d)	Máy phục vụ cung cấp thông tin	Bộ định tuyến	Máy khách
e)	Bộ định tuyến	Máy khách	Máy phục vụ cung cấp thông tin
f)	Bộ định tuyến	Máy phục vụ cung cấp thông tin	Máy khách

Câu hỏi con 3

Có một vấn đề được mô tả dưới đây. Từ nhóm câu trả lời dưới đây, hãy chọn nguyên nhân có thể gây ra vấn đề này.

Tất cả các máy khách kết nối tới mạng LAN điều hành và tới mạng LAN nhà máy có thể giao tiếp với máy phục vụ cung cấp thông tin và những máy phục vụ trên Internet.

Một máy khách được thêm vào mạng LAN tại nhà máy. Máy khách mới có thể giao tiếp với máy phục vụ cung cấp thông tin hoặc máy phục vụ trên Internet.

Tất cả các máy khách bao gồm cả các máy khách mới thêm vào trong mạng LAN nhà máy sử dụng cùng thiết bị, cùng phần mềm và cùng thiết lập trình duyệt.

Các câu trả lời

a) Một địa chỉ IP sai được thiết lập trên máy phục vụ cung cấp thông tin gây ra việc điều khiển truy cập sai.

b) Một địa chỉ IP sai được thiết lập trên máy phục vụ proxy "w" gây ra việc điều khiển truy cập sai.

c) Một địa chỉ IP sai được thiết lập trên bộ định tuyến "w" gây ra việc điều khiển truy cập sai

d) Thiết lập trên bộ định tuyến "w" sai làm cho quá trình truyền thông sử dụng giao thức HTTP bị vô hiệu hóa

Đáp án câu 1

Mạng truyền thông

Đáp án đúng

[Câu hỏi con 1] e, g

[Câu hỏi con 2] b

[Câu hỏi con 3] b

Giải thích

Đây là câu hỏi liên quan tới mạng truyền thông (LAN) được kết nối Internet. Có các mạng: mạng intranet – hệ thống mạng cục bộ của công ty sử dụng công nghệ Internet, và extranet – kết nối các công ty đối tác kinh doanh và khách hàng. Trong cả hai trường hợp, nếu mạng được nối với Internet, điều quan trọng là làm thế nào để giải quyết những vấn đề liên quan tới an toàn bảo mật.

Để ngăn ngừa sự truy cập trái phép và để bảo đảm sự an toàn bảo mật, một tường lửa (hệ thống phòng thủ) được xây dựng. Chức năng cơ bản của tường lửa là kiểm soát luồng dữ liệu. Nhìn chung, để đạt được mục đích đó cần thiết lập một bộ định tuyến hoặc một máy phục vụ proxy.

[Câu hỏi con 1]

Một bộ định tuyến có các chức năng tìm đường (routing) và lọc dữ liệu. Tìm đường (routing) là quá trình chọn đường theo đó dữ liệu gửi từ nút nguồn tới nút đích trong mạng. Thông tin địa chỉ (một bảng địa chỉ) của những nút mạng được nối với mạng cục bộ được lưu trong bộ định tuyến, và thông tin này được sử dụng khi tìm đường. Lọc là một chức năng điều khiển luồng dữ liệu trong mạng.
Câu hỏi con hỏi về “bộ lọc của bộ định tuyến h”, vì thế bạn chỉ có thể quan sát quan hệ truy cập bên trong giữa những máy phục vụ cung cấp thông tin và các máy khách trong mạng LAN tại trụ sở chính. Mỗi máy khách đều được quản lý bởi một máy phục vụ proxy, và chúng "luôn luôn đi qua qua máy phục vụ proxy", vì vậy bạn có thể coi nguồn của truyền thông từ máy khách h1 như máy phục vụ proxy h. Lúc này, đích là máy phục vụ cung cấp thông tin. Truyền thông từ máy phục vụ cung cấp thông tin chỉ là ngược lại nên câu trả lời đúng là (e) và (g).

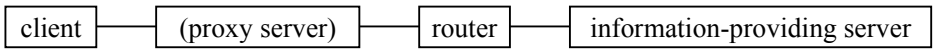
Tường lửa trong nhóm câu trả lời chỉ cho quyền truy cập truyền thông tới mạng Internet. “Máy phục vụ proxy”, chứ không phải là “Máy khách h1”, giao tiếp với “Bộ định tuyến h”. Vì thế, tất cả các lựa chọn khác đều sai.

[Câu hỏi 2]

Dịch vụ cung cấp thông tin nội bộ của công ty xuất hiện giữa máy khách và máy phục vụ cung cấp thông tin thông qua máy phục vụ proxy. Tập trung vào các đường truyền thông này, chúng ta thấy 4 thiết bị liên quan là máy khách, máy phục vụ proxy, bộ định tuyến và máy phục vụ cung cấp thông tin.

Vị trí của các máy phục vụ proxy đã được biết (được gắn nhãn) và chúng được kết nối với các máy khách và các bộ định tuyến. Vì bộ định tuyến được nối với máy phục vụ proxy và máy phục vụ cung cấp thông tin nên (2) trong Hình 2 không thể là máy khách, nếu là máy khách sẽ dẫn đến mâu thuẫn. Một bộ định tuyến là một đơn vị chỉ có chức năng trên lớp mạng vì thế (2) là bộ định tuyến .

Do đó, chúng ta có



Và câu trả lời đúng là (b).

[Câu hỏi con 3]

Theo đề bài của câu hỏi, các thông tin sau được cung cấp:

(1) Mọi máy khách đều có thể giao tiếp với máy phục vụ cung cấp thông tin và máy phục vụ trên Internet.
(2) Các máy khách mới được thêm vào mạng LAN tại nhà máy có khả năng giao tiếp trực tiếp với các máy khách khác trong mạng LAN tại nhà máy.

Các thông tin ở trên chỉ ra vấn đề ko phải với mạng LAN hay với máy khách đã thêm vào. Giao tiếp từ một máy khách tới máy phục vụ cung cấp thông tin hoặc ác máy phục vụ khác trên Internet luôn luôn đi qua máy phục vụ proxy. Máy khách không giao tiếp trực tiếp với bộ định tuyến hoặc máy phục vụ ngoại trừ máy phục vụ proxy; thay vào đó, máy phục vụ proxy tạo ra các giao tiếp này với tư cách là máy khách.

Vì thế, máy phục vụ proxy được dùng để kiểm soát truy cập sử dụng địa chỉ IP được thiết lập trong máy phục vụ proxy. Nếu truy cập được tạo ra bị từ chối bởi thiết lập đó thì máy phục vụ proxy sẽ từ chối các yêu cầu, vì thế kiểm soát truy cập bởi sự thiết lập địa chỉ IP trong máy phục vụ proxy w là sai. Do đó, câu trả lời đúng là (b). Giao tiếp giữa máy khách mới được thêm vào và các máy khách đang tồn tại được thiết lập bên trong cùng mạng LAN, vì thế chúng giao tiếp trực tiếp với nhau mà không cần thông qua máy phục vụ proxy. Điều này giải thích tại sao giao tiếp trên cùng mạng LAN là có thể ngay cả khi máy phục vụ proxy w có các thiết lập sai. Đối với các lựa chọn khác trong nhóm câu trả lời, dưới đây là một vài nhận xét:

a) Nếu kiểm soát truy cập tới máy phục vụ cung cấp thông tin là sai thì vẫn có thể truy cập ra ngoài.
c) Trong việc truyền thông thông qua bộ định tuyến w, giao tiếp giữa tất cả các máy khách bên trong mạng LAN tại công ty thông qua máy phục vụ proxy w, nên khi thêm mới một máy khách thì không cần phải thiết lập địa chỉ IP tại bộ định tuyến w.
d) Nếu truyền thông bằng giao thức HTTP bị vô hiệu hóa thì không có giao tiếp nào giữa các máy khách khác tới các máy phục vụ trên Internet hoặc máy phục vụ cung cấp thông tin.

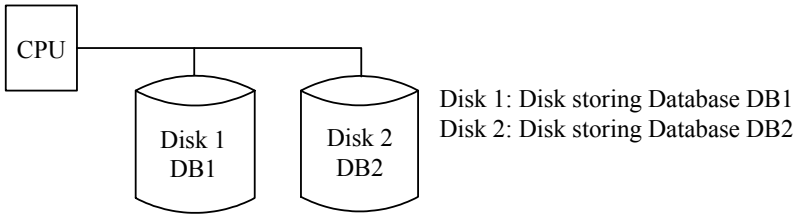
5 Công nghệ xử lý thông tin

[Phạm vi các câu hỏi]
Hiệu năng hệ thống, Tính tin cậy,
Quản lý rủi ro, Tính an toàn, Chuẩn hóa,
Các hoạt động nghiên cứu...

Câu hỏi 1

Q1. Đọc mô tả sau về hiệu năng xử lý giao dịch và trả lời câu hỏi.

Trong hình dưới đây, có một máy phục vụ gồm một bộ vi xử lý và hai đĩa cứng. Cơ sở dữ liệu DB1 và DB2 nằm trên hai đĩa khác nhau. Trong trường hợp máy phục vụ này, các giao dịch khác nhau có thể thực hiện độc lập trên các đĩa. Bảng 1 đưa ra thời gian trung bình sử dụng tài nguyên hệ thống trong mỗi giao dịch dựa trên kết quả phân tích xử lý giao dịch.



Hình. Thiết kế máy phục vụ

Bảng 1 Thời gian sử dụng tài nguyên hệ thống trung bình trong mỗi giao dịch

Tài nguyên hệ thống	Thời gian sử dụng trung bình trong một giao dịch
CPU	20 ms
Disk 1	80 ms
Disk 2	40 ms

Số lượng giao dịch máy phục vụ xử lý trên một giây tại thời điểm bất kỳ gọi là TPS (Transcactions Per Second) của máy phục vụ đó. Khi hoạt động, TPS của máy phục vụ , tỉ lệ sử dụng của một tài nguyên, thời gian sử dụng trung bình của tài nguyên hệ thống trong mỗi giao dịch (viết tắt là thời gian sử dụng trung bình) liên hệ với nhau như sau::

Tỉ lệ sử dụng = TPS × Thời gian sử dụng trung bình …….. [1]

Thí dụ, TPS của máy phục vụ được tính theo công thức sau khi thời gian sử dụng trung bình của một tài nguyên hệ thống là 50ms và tỉ lệ sử dụng của tài nguyên đó là 80%.

$$\text{TPS} = \frac{0.8}{0.05} = 16$$

Câu hỏi

Từ các nhóm đáp án dưới đây, chọn đáp án đúng nhất để điền vào chỗ trống [ ] trong các mô tả sau.

(1) Khi giới hạn trên của tỉ lệ sử dụng mỗi tài nguyên hệ thống trong hình đạt 40%, giới hạn trên của TPS hệ thống là [A].

(2) Khi tiếp tục giả định giới hạn trên của tỉ lệ sử dụng mỗi tài nguyên hệ thống là 40%, để tăng giới hạn trên TPS hệ thống lên 4 lần thì cần giảm thời gian truy cập của đĩa đi ít nhất [B]. Giả sử tài nguyên hệ thống tăng lên n lần thì thời gian truy cập tài nguyên hệ thống trong xử lý giao dịch cũng giảm đi còn 1/n.

(3) Khi TPS của máy phục vụ là 10 thì thời gian đáp ứng trung bình trong xử lý một giao dịch xấp xỉ [C] lần so với thời gian xử lý giao dịch đó. Ở đây, giả sử rằng thời gian đáp đáp ứng trung bình cho mỗi giao dịch bằng tổng thời gian đáp ứng trung bình cho mỗi tài nguyên hệ thống, và thời gian đáp ứng trung bình cho mỗi tài nguyên hệ thống được tính theo công thức sau đây.

$$\text{Thời gian đáp ứng trung bình} = \frac{\text{Thời gian sử dụng trung bình}}{1 - \text{Tỉ lệ sử dụng}} \quad \text{…….. [2]}$$

(4) Theo công thức [1] và [2], có thể thấy khi TPS của máy phục vụ tăng lên, tỉ lệ sử dụng tài nguyên hệ thống cũng tăng và [D].

Nhóm đáp án cho A:

a) 0.005 b) 0.01 c) 0.02
d) 5 e) 10 f) 20

Nhóm đáp án cho B:

a) tăng số lượng Disk 1 lên bốn chiếc.
b) tăng số lượng Disk 2 lên bốn chiếc.
c) tăng số lượng Disk 1 lên hai và Disk 2 lên hai.
d) tăng số lượng Disk 1 lên hai và Disk 2 lên bốn.
e) tăng số lượng Disk 1 lên bốn và Disk 2 lên hai.

Nhóm đáp án cho C:

a) 0.48 b) 0.95 c) 1.66
d) 2.86 e) 3.51

Nhóm đáp án cho D:

a) thời gian đáp ứng trung bình tăng.
b) thời gian đáp ứng trung bình giảm.
c) thời gian sử dụng trung bình tăng.
d) thời gian sử dụng trung bình giảm.

Đáp án câu 1

Hiệu năng xử lý giao dịch

Đáp án đúng

A – d, B – e, C – e, D – a

Giải thích

Chủ đề của câu hỏi này là về hiệu năng tính toán trong xử lý giao dịch, nhưng nói chung, những tính toán về tỉ lệ sử dụng và thời gian đáp ứng trung bình đều nằm trong lý thuyết hàng đợi. Tuy vậy, các công thức tính, các thủ tục đã được mô tả cụ thể trong câu hỏi do vậy cũng không cần kiến thức về lý thuyết hàng đợi để trả lời câu hỏi này. Mục đích câu hỏi này là để xác định bạn có thể thực hiện tính toán chính xác dựa trên những giải thích của câu hỏi không.

(1) Liên quan đến TPS, số lượng giao dịch trong một giây, câu hỏi đưa ra thí dụ một TPS = 0.8 / 0.05 = 16 với giả thiết về thời gian sử dụng trung bình là 50 ms và tỉ lệ sử dụng là 80%. Bây giờ, giả sử rằng tỉ lệ sử dụng là 40%, và thời gian sử dụng trung bình cho mỗi thiết bị được cho trong bảng 1, ta có thể tính TPS cho mỗi thiết bị sử dụng giá trị trên và có kết quả sau.

Nguồn tài nguyên hệ thống	TPS
CPU	0.4 / 0.02=20
Disk 1	0.4 / 0.08=5
Disk 2	0.4 / 0.04=10

Nếu một quá trình có thể thực hiện song song với nhiều đơn vị xử lý, hiệu năng xử lý của toàn hệ thống có thể bị ảnh hưởng xấu đi bởi đơn vị xử lý có hiệu năng kém nhất. Những đơn vị xử lý như vậy gọi là cổ chai. Trong trường hợp này, Đĩa 1 là cổ chai, và giới hạn trên cho TPS của máy phục vụ là đáp án (d) 5..

(2) Để tăng gấp bốn lần giới hạn trên cho TPS của máy phục vụ, nghĩa là tăng lên 20. TPS của CPU là 20, của các đĩa 1 và 2 là 5 và 10, đều dưới ngưỡng 20. Do vậy cần cải thiện hiệu năng của các đĩa. Chiến lược cải thiện hiệu năng là tăng số lượng tài nguyên hệ thống (đĩa cứng). Hiệu năng của mỗi đơn vị đĩa tỉ lệ thuận với số lượng đĩa, bởi giả định rằng nếu số lượng tài nguyên tăng lên n lần thì thời gian truy cập giả đi còn 1/n. Một vài ý trong câu hỏi có thể rắc rối, nhưng chúng ta có thể giả định rằng thời gian truy cập bằng với thời gian sử dụng trung bình (Thời gian truy cập = tỉ lệ sử dụng / TPS).

Trước hết, để tăng TPS của Disk 1 lên 20 trong khi giữ giới hạn trên 40% tỉ lệ sử dụng của nó, thời gian truy cập cần là 0.02 giây (= 0.4 / 20). Thời gian truy cập này bằng ¼ thời gian truy cập hiện tại – 0.08 giây, do vậy số lượng đĩa phải tăng lên gấp 4. Tương tự với Disk 2, thời gian truy cập hiện tại là 0.04 giây và cần giảm xuống 0.02, do vậy cần tăng gấp đôi số lượng Disk 2. Vậy đáp án đúng là (e), tăng số lượng Disk 1 lên bốn và số Disk 2 lên hai.

(3) Thời gian xử lý giao dịch là toàn bộ thời gian (tổng thời gian) cần thiết cho mỗi quá trình con trong giao dịch đó, không bao gồm thời gian đợi. Nói cách khác, thời gian đáp ứng trung bình bao gồm tổng thời gian xử lý giao dịch và thời gian đợi. Nội dung của quá trình xử lý giao dịch trong câu hỏi này không được chỉ ra, nhưng câu hỏi chỉ ra rằng "thời gian đáp ứng trung bình xử lý giao dịch là tổng thời gian đáp ứng trung bình cho mỗi tài nguyên hệ thống". Do vậy , có thể giả định thời gian xử lý giao dịch là tổng thời gian sử dụng trung bình của mỗi tài nguyên hệ thống. Như vậy, theo bảng 1, thời gian giao dịch là 20 + 80 + 40 = 140 ms. Thời gian đáp ứng trung bình cần được tính cho mỗi tài nguyên hệ thống, sử dụng công thức [1] và [2].

CPU: Tỉ lệ sử dụng = $10 \times 0.02 = 0.2$,
Thời gian đáp ứng trung bình = $0.02 / (1 - 0.2) = 0.025 = 25$ ms.
Đĩa 1: Tỉ lệ sử dụng = $10 \times 0.08 = 0.8$,
Thời gian đáp ứng trung bình = $0.08 / (1 - 0.8) = 0.4 = 400$ ms.
Đĩa 2: Tỉ lệ sử dụng = $10 \times 0.04 = 0.4$,
Thời gian đáp ứng trung bình = $0.04 / (1 - 0.4) = 0.067 = 67$ ms.

Thời gian đáp ứng trung bình cho cả máy phục vụ là 25 + 400 + 67 = 492 ms. Câu hỏi là thời gian này lớn gấp bao nhiêu lần thời gian xử lý giao dịch, đáp án là 492 / 140 = 3.514, tương ứng với phương án (e).

(4) Từ công thức [1], Tỉ lệ sử dụng = TPS × Thời gian sử dụng trung bình, có thể thấy khi TPS tăng, tỉ lệ sử dụng cũng tăng. Công thức [2] như sau:

$$\text{Thời gian đáp ứng trung bình} = \frac{\text{Thời gian sử dụng trung bình}}{1 - \text{Tỉ lệ sử dụng}}$$

Từ công thức này chúng ta thấy, khi tỉ lệ sử dụng tăng lên, thời gian đáp ứng trung bình cũng tăng. Do đó, câu trả lời đúng là (a). Nếu các công thức trên chưa đủ thuyết phục bạn, hãy xem xét tình huống cụ thể sau : tỉ lệ sử dụng tăng lên từ 0.2 lên 0.4. Thời gian sử dụng trung bình có thể bất kỳ, ở đây ta chọn là 1. Nếu tỉ lệ sử dụng là 0.2, thời gian đáp ứng trung bình là 1 / (1 - 0.2) = 1.25. Nếu tỉ lệ sử dụng tăng lên 4, thời gian đáp ứng trung bình là 1 / (1 - 0.4) = 1.67.

[Hàng đợi]

Trả lời câu hỏi này không nhất thiết phải có những kiến thức về hàng đợi. Thực tế, câu hỏi trên đã gợi ý sử dụng lý thuyết hàng đợi để trả lời. Tuy đây là một bài tập, nhưng nó đưa ra ý tưởng khá hay để học lý thuyết hàng đợi. Phần sau đây sẽ giải thích thêm để bạn đọc tham khảo.

Chắc hẳn ai cũng đã từng chờ đợi ở chợ, siêu thị hay ngân hàng. Một khái niệm tương tự cũng được áp dụng cho xử lý máy tính. Người ta đánh giá hiệu năng của máy tính bằng hai phương diện: thời gian xử lý theo phương diện máy tính và thời gian đáp ứng theo phương diện người dùng. Lý thuyết hàng đợi áp dụng cho phương diện sau: thời gian đáp ứng.

Nhắc lại rằng, thời gian đáp ứng là thời gian từ khi yêu cầu được gửi đến khi hệ thống phản hồi yêu cầu đó. Trong thế giới thực, tại quầy thu ngân trong ngân hàng, thời gian đáp ứng được tính từ lúc bạn xếp hàng đợi đến lúc kết thúc công việc. Thời gian này bao gồm thời gian hệ thống phục vụ những người đứng trước bạn và thời gian hệ thống phục vụ bạn.

Thời gian đáp ứng = thời gian đợi
+ thời gian bạn được phục vụ.

Thời gian bạn phải đợi những người khác gọi là thời gian đợi, và thời gian đáp ứng là tổng thời gian bạn phải đợi người khác và thời gian hệ thống phục vụ chính bạn. Thời gian đợi phụ thuộc vào chiều dài của hàng đợi và thời gian cần thiết phục vụ mỗi người. Thời gian phục vụ mỗi người có thể xác định bằng các thông số như kỹ năng…Thực tế, một câu hỏi về lý thuyết hàng đợi thường ước lượng trước được chiều dài hàng đợi.

Xem xét các hàng đợi thanh toán ở siêu thị. Khi nào chúng dài ra ? Chỉ khi có đông người đến mua sắm. Nhưng ngay cả khi có hai hàng đợi với cùng số lượng người và được thành lập cùng một lúc, thì người thu tiền chưa có kinh nghiệm sẽ mất nhiều thời gian hơn, hàng đợi sẽ dài hơn. Nói cách khác, thời gian đợi của hàng đợi được xác định bằng số lượng khách hàng và kỹ năng của người thu tiền. Thuật ngữ có thể diễn tả được cả hai thông tin trên là tỉ lệ sử dụng. Tỉ lệ sử dụng biểu diễn mức độ, khả năng sử dụng được của hệ thống, do vậy giá trị của nó được tính từ kỹ năng của người thu tiền và số lượng khách hàng.

Dựa trên ý tưởng trên, nếu thời gian sử dụng của CPU là 20 ms thì nó có thể xử lý 50 giao dịch / giây. Nếu có 10 giao dịch, tỉ lệ sử dụng là 10/50 = 0.2. Trong câu hỏi này, số lượng giao dịch trên một giây (TPS) được nhân với thời gian sử dụng để tính tỉ lệ sử dụng. Giá trị TPS × (thời gian sử dụng) cho ta biết trong một giây tài nguyên được sử dụng bao lâu phần trăm. Đó cũng chính là tỉ lệ sử dụng. Tuy nhiên, hãy cẩn thận với đơn vị, đơn vị thời gian cho giao dịch là 1 giây, vì vậy thời gian sử dụng cũng phải tính theo giây. Nếu tất cả được chuyển đổi sang mili giây, chúng ta có TPS × (thời gian sử dụng) = 10 × 20 = 200 không hợp lý lắm với ý nghĩa của tỉ lệ sử dụng (không lớn hơn 1).

Trong các mô hình chung được sử dụng, chiều dài của hàng đợi được tính bằng (tỉ lệ sử dụng) / (1- tỉ lệ sử dụng). Vì vậy phương trình này có thể sử dụng để tìm chiều dài hàng đợi. Thời gian xử lý cả hàng đợi này là thời gian đợi. Do vậy có thể tính thời gian đợi = chiều dài hàng đợi × thời gian xử lý(thời gian sử dụng). Trong thí dụ trên, thời gian sử dụng là 20 mili giây và tỉ lệ sử dụng là 0.2. Thay số vào phương trình ta có chiều dài hàng đợi là 0.2 / (1 – 0.2) = ¼. Bạn có thể băn khoăn giá trị này không phải là số nguyên, nhưng có thể hiểu là cứ bốn người thì sẽ có một người phải đợi. Vậy thời gian đợi là ¼ thời gian xử lý, ¼ × 20 mili giây = 5 mili giây. Cộng với thời gian xử lý cho một giao dịch là 20 mili giây ta có thời gian đáp ứng là 25 mili giây. Hơn nữa, công thức [2] cũng được giới thiệu trong câu hỏi.

$$\text{Thời gian đáp ứng trung bình} = \frac{\text{Thời gian sử dụng trung bình}}{1 - \text{Tỉ lệ sử dụng}}$$

Nếu thay thời gian sử dụng trung bình (20 mili giây) và tỉ lệ sử dụng là 0.2 vào công thức, ta cũng có 20 / (1- 0.2) = 25 mili giây. Thời gian đáp ứng là tổng thời gian xử lý hàng đợi và thời gian xử lý một giao dịch. Ta có thể tính chiều dài hàng đợi như sau:

$$\text{Chiều dài hàng đợi} = \frac{\text{Tỉ lệ sử dụng}}{1 - \text{Tỉ lệ sử dụng}}$$

Ta có số lượng xử lý trong khoảng thời gian đáp ứng như sau::

$$\text{Số lượng xử lý trong khoảng thời gian đáp ứng} = \frac{\text{Tỉ lệ sử dụng}}{(1 - \text{Tỉ lệ sử dụng})} + 1$$

$$= \frac{\text{Tỉ lệ sử dụng}}{1 - \text{Tỉ lệ sử dụng}} + \frac{(1 - \text{Tỉ lệ sử dụng})}{1 - \text{Tỉ lệ sử dụng}}$$

$$= \frac{\text{Tỉ lệ sử dụng} + (1 - \text{Tỉ lệ sử dụng})}{1 - \text{Tỉ lệ sử dụng}} = \frac{1}{1 - \text{Tỉ lệ sử dụng}}$$

Và để tính thời gian đáp ứng, ta nhân số này với thời gian xử lý (thời gian sử dụng cho một giao dịch)

$$\frac{\text{Thời gian sử dụng}}{1 - \text{Tỉ lệ sử dụng}}$$

Tại sao lại có thời gian đợi ngay cả khi tỉ lệ sử dụng nhỏ hơn 1. Thí dụ, với Đĩa 1 trong câu hỏi này, tỉ lệ sử dụng là 0.8 khi TPS là 10. Nếu chúng ta tính thời gian của hàng đợi, ta sẽ có 0.8 / (1 – 0.8) = 4. Thời gian truy cập đĩa là 80 mili giây với TPS là 10, thí dụ cứ 100 mili giây mới có một giao dịch, do vậy đĩa sẽ có 20 mili giây rảnh rỗi sau mỗi lần truy câp và bạn thường bỏ qua thời gian đợi này. Đây là dòng suy nghĩ đúng. Ở đây, giả định rằng thời gian truy cập và giao dịch xảy ra đều đặn, do vậy sẽ không có thời gian đợi. Tuy vậy mô hình của hàng đợi dựa trên giả thuyết thời gian truy cập cũng như các giao dịch xảy ra ngẫu nhiên. Nếu chúng là ngẫu nhiên hoàn toàn thì sẽ không thể thực hiện một tính toán nào, vì vậy giả định rằng ít nhất chúng ta biết được giá trị "trung bình". Để ý rằng tất cả các giá trị trong câu hỏi đều mang nghĩa "trung bình". Đây là một từ khóa, thời gian sử dụng là thời gian sử dụng trung bình, thời gian đáp ứng là thời gian đáp ứng trung bình. Trong phần giải thích trên, từ "trung bình" bị bỏ qua, nhưng nói một cách chính xác: tất cả phải có chữ "trung bình". Trong câu hỏi này, tất cả các mô hình hàng đợi, số lượng giao dịch trong một đơn vị thời gian được giả định có phân bố Poisson, và thời gian xử lý một giao dịch có phân bố hàm mũ.

Câu hỏi 2

Q2. Đọc mô tả sau đây về lập lịch và trả lời câu hỏi.

Một nhà thầu được chọn để hoàn thành một dự án trong thời gian nhanh nhất và với chi phí thấp nhất. Dự án này có thể được chia làm tám công việc: từ A đến H, và mỗi công việc được gán một thứ tự thực hiện, như bảng 1 dưới đây. Một công việc sẽ không thể bắt đầu nếu nó yêu cầu một công việc khác trước đó mà chưa hoàn tất.

Bảng 2 đưa ra số lượng ngày công và chi phi nếu giao cho nhà thầu X hay Y. Nhà thầu X có thể hoàn tất công việc với chi phí ít hơn nhà thầu Y nhưng lại cần nhiều ngày công hơn.

Bảng 1 :

Danh sách công việc

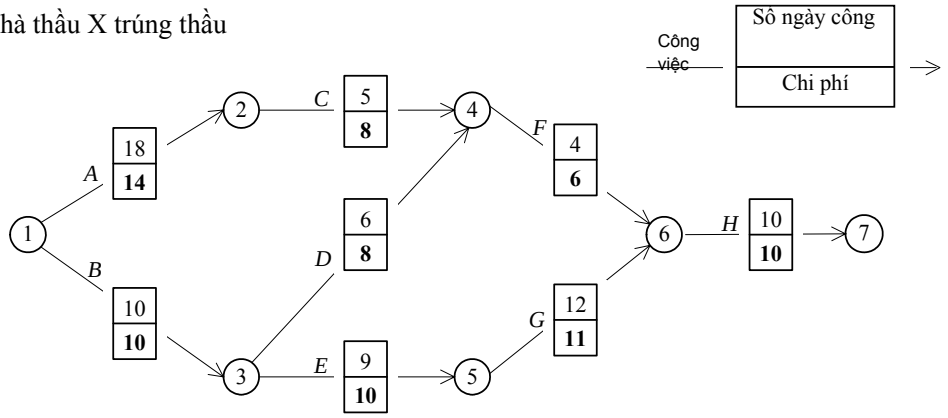
Công việc	Công việc yêu cầu hoàn thành
A	–
B	–
C	*A*
D	*B*
E	*B*
F	*C*, *D*
G	*E*
H	*F*, *G*

Bảng 2: Số lượng ngày công và chi phí theo các nhà thầu

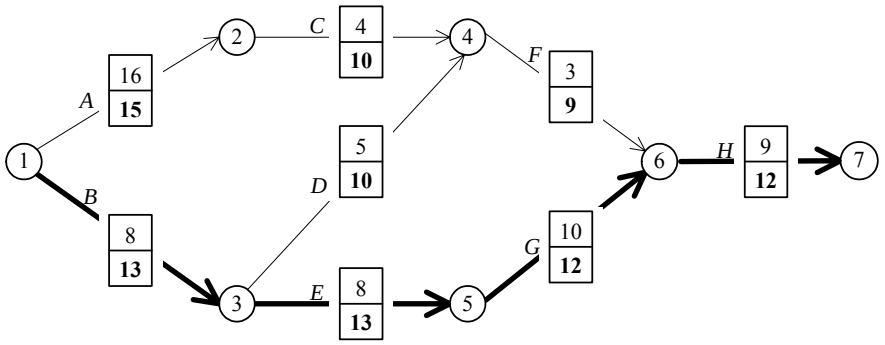
Công việc	Nhà thầu *X*		Nhà thầu *Y*	
	Số lượng ngày công	Chi phí	Số lượng ngày công	Chi phí
A	18	14	16	15
B	10	10	8	13
C	5	8	4	10
D	6	8	5	10
E	9	10	8	13
F	4	6	3	9
G	12	11	10	12
H	10	10	9	12

Trang tiếp theo mô tả cách thức các nhà thầu hoàn tất công toàn bộ công việc.

Nếu nhà thầu X trúng thầu



Nếu nhà thầu Y trúng thầu



Hình: Biểu đồ dự án

Câu hỏi

Từ các nhóm câu trả lời dưới đây, chọn câu trả lời đúng nhất điền vào chỗ trống [ ] chu phù hợp với các mô tả sau.

(1) Khi nhà thầu X trúng thầu, chi phí hoàn tất dự án là [A] và số lượng ngày công cần thiết là [B].

(2) Khi nhà thầu Y trúng thầu , chi phí hoàn tất dự án là 94 và số ngày công cần thiết là 35. Đường đi ngắn nhất là đường tô đậm BEGH trong biểu đồ trên. Nếu nhà thầy X trúng thấu để thực hiện một phần của dự án, số lượng ngày công vẫn như trên: 35 ngày, nhưng chi phí có thể giảm xuống, cụ thể, chi phí thấp nhất sẽ là [C].

Nhóm câu trả lời

a) 35	b) 41	c) 63	d) 74	e) 77
f) 86	g) 87	h) 88	i) 89	j) 90

Đáp án câu 2

Lập lịch

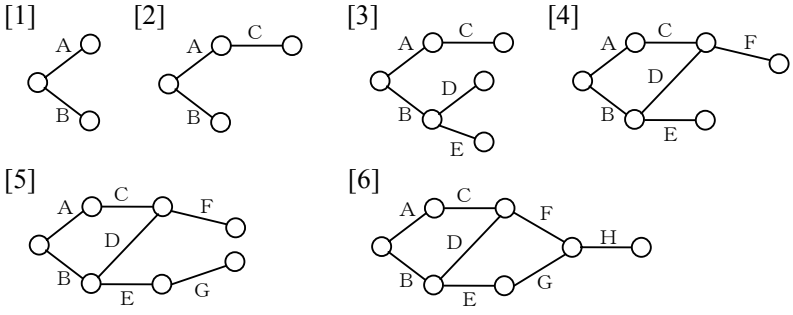
Đáp án đúng

A – e, B – b, C – g

Giải thích

Đây là câu hỏi liên quan đến việc lựa chọn nhà thâu sao cho dự án có thể hoàn thành với số ngày công tối thiểu và chi phí thấp nhất có thể.
Dự án này có thể chia làm tám công việc đánh dấu từ A đến H và chúng có yêu cầu về thứ tự thực hiện, như đã được chỉ ra trong bảng 1. Các hình dưới đây mô tả mỗi liên hệ này, từ đó xây dựng ra biểu đồ PERT.

[1] *A* và *B* không có yêu cầu công việc trước đó, do vậy có thể thực hiện song song.
[2] *C* cần *A* phải được thực hiện trước, vì vậy *C* phải thực hiện sau *A*.
[3] *D* và *E* cần B thực hiện trước, vì vậy chúng cần phải thực hiện sau *B*.
[4] *F* cần *C* và *D* thực hiện trước, vì vậy *F* phải được thực hiện sau *C* và *D*.
[5] *G* cần *E* thực hiện trước, vì vậy *G* phải thực hiện sau *E*.
[6] *H* cần *D* và *G* thực hiện trước, vì vậy *H* phải thực hiện sau *F* và *G*.



A: Chi phí thuê nhà thầu X thực hiện tất cả các công việc đơn giản là tổng tất cả chi phí trong bảng 2. Đó là (e): 14 + 10 + 8 + 8 + 10 + 6 + 11 + 10 = 77.

B: Số lượng ngày công cần thiết để hoàn tất có thể tính theo biểu đồ PERT. Số lượng công việc khá ít do vậy sẽ không mất thời gian để xem xét tất cả các đường đi có thể trong biểu đồ.

- $A \rightarrow C \rightarrow E \rightarrow H$ (37 ngày)
- $B \rightarrow D \rightarrow F \rightarrow H$ (30 ngày)
- $B \rightarrow E \rightarrow G \rightarrow H$ (41 ngày)

Do vậy số lượng ngày công cần thiết là 31 + 10 = 41 ngày (b). Đường đi tối ưu là $B \rightarrow E \rightarrow G \rightarrow H$.

C: Chi phí cần thiết để thuê nhà thầu Y thực hiện tất cả công việc là tổng các chi phí của Y trong bảng 2: 15 + 13 + 10 + 10 + 13 + 9 + 12 + 12 = 94, và số ngày cần thiết là 35. Mục tiêu bây giờ là để giữ nguyên 35 ngày này nhưng giảm chi phi bằng việc thuê X thực hiện một phần dự án.

Từ câu hỏi và từ bảng 2, dễ thấy rằng mỗi công việc, nhà thầu X mất nhiều thời gian hơn nhưng chi phí thì ít hơn nhà thầu Y. Từ điều kiện này, có thể giảm chi phí nếu ta thuê X thực hiện những công việc có dư ra một chút thời gian trong dự án nhưng không ảnh hưởng đến các công việc sau đó.

Đường đi tối ưu là B – E – G - H, tất cả mất 35 ngày, ta cần gán những công việc không nằm trên đường đi này : A, C, D, và F cho nhà thầu X. Ở đây, tất cả những công việc cho tới nút [6] phải được hoàn tất trong 35 – (số ngày cần thiết cho H) = 35 – 9 = 26 (ngày). Hơn nữa, bởi số lượng ngày dư ra và chi phí tùy thuộc và các công việc, ta ưu tiên vào những công việc có tỉ lệ giữa chi phí/ngày công cao. Bảng dưới đây so sánh các công việc A,C,D và F giữa nhà thầu X và Y:

Công việc	# số ngày	Chi phí	Chi phí /Ngày
A	+2	-1	-0.5
C	+1	-2	-2
D	+1	-2	-2
F	+1	-3	-3

Trước hết giả sử cho nhà thầu X làm công việc F. Số lượng ngày tăng lên 1 nhưng chi phi giảm đi 3.

Nếu F được gán cho X, các công việc cho đến nút [4] phải kết thúc trong vòng 26 –(số ngày để X hoàn thành F) = 26 – 4 = 22 (ngày). Ở đây, nếu ta cũng thuê nhà thầu X thực hiện công việc C và D, mỗi công việc sẽ bị tăng lên 1 ngày, nhưng chúng ta vẫn có thể hoàn tất trong vòng 22 ngày, cùng lúc đó giảm chi phí đi 4 (mỗi công việc giảm đi 2).

Cuối cùng xem xét công việc A. Công việc C đã được gán cho nhà thầu X, tất cả các công việc đến nut [2] phải hoàn tất trong vòng 22- (số ngày X hoàn thành C) = 22 – 5 = 17 ngày. Tuy vậy, nếu công việc A được gán cho nhà thầu X, số ngày cần thiết đến điểm đó sẽ là 18, vì vậy không thể gán A cho X.

Như vậy, chi phí tối thiểu, khi nhà thầu X được thuê thực hiện các công việc C , D và F là
(chi phí thuê Y là tất) – (chi phí giảm được khi thuê X)
= 94 – (3 + 2 + 2)
= 94 – 7 = 87.
Đáp án đúng là (g).

Câu hỏi 3

Q3. Đọc mô tả sau đây về biểu diễn số dấu phảy động, sau đó trả lời các câu hỏi con từ 1 đến 3.

Biểu diễn một số dấu phảy động 32-bit như hình 1 dưới đây.

Ý nghĩa của các bit như sau:

(1) Bit 0: Dấu

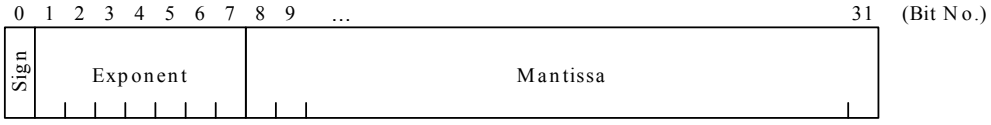
Bit này chưa dấu của phần định trịnh. "0" biểu diễn giá trị dương, và "1" biểu diễn giá trị âm.

(2) Các bit 1 đến 7: Phần mũ.

Các bit này biểu diễn phần mũ theo hệ nhị phân. Giá trị âm được biểu diễn theo số bù hai.

(3) Các bit 8 đến 31: Phần định trị.

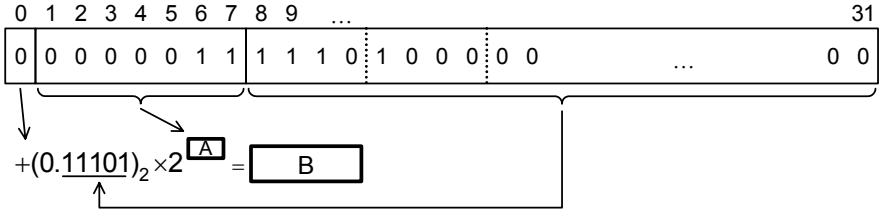
Những bit này biểu diễn giá trị tuyệt đối của phần định trị. Bit 8 là bit đầu tiên trong phần lẻ của số dấu phảy động.



Hình .1 Định dạng biểu diễn số dấu phảy động

Câu hỏi con 1

Hình 2 minh họa phương pháp chuyển đổi giá trị nhị phân trong biểu diễn số dấu phảy động sang giá trị thập phân. Chọn giá trị thập phân điền vào các chỗ trống [A] và [B] từ các nhóm câu trả lời dưới. Ở đây, "$(0.11101)_2$" trong hình 2 là một số nhị phân.



Hình. 2 Phương pháp chuyển đổi số nhị phân trong biểu diễn dấu phẩy động sang số thập phân.

Nhóm câu trả lời cho A:

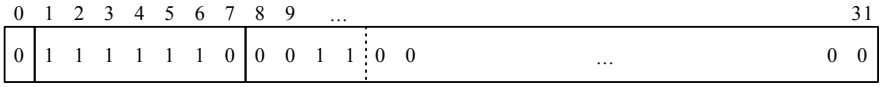
a) 0 b) 1 c) 2 d) 3 e) 4

Nhóm câu trả lời cho B:

a) 0.725 b) 0.74 c) 7.25
d) 7.4 e) 72.5 f) 74

Câu hỏi con 2

Từ nhóm câu trả lời dưới đây, chọn ra biểu diễn thập phân đúng cho số nhị phân trong dạng dấu phảy động sau.



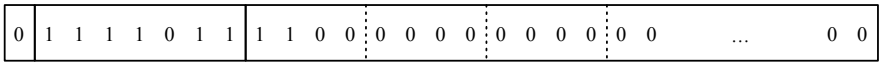
Nhóm câu trả lời:

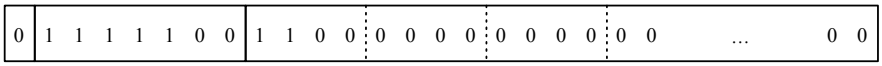
a) 3×2^{-6} b) 3×2^{-4} c) 3×2^{-1} d) 3×2^{0}
e) 3×2^{1} f) 3×2^{2} g) 3×2^{122}

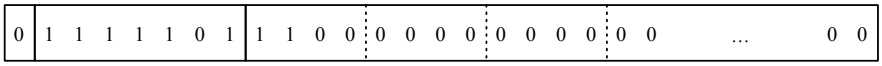
Câu hỏi con 3

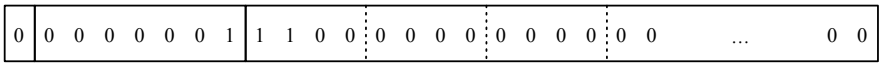
Chọn dạng biểu diễn “đã chuẩn” của số dấu phảy động trong câu hỏi 2 từ nhóm câu trả lời dưới đây. Ở đây, "chuẩn hóa" nghĩa là biến đổi phần mũ và phần định trị để bit trái nhất của phần định trị là “1”.

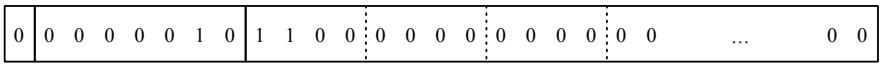
Nhóm câu trả lời:

a) 

b) 

c) 

d) 

e) 

Đáp án câu 3

Biểu diễn số dấu phảy động

Đáp án đúng

[Câu hỏi con 1] A – d, B – c

[Câu hỏi con 2] a

[Câu hỏi con 3] b

Giải thích

Đây là câu hỏi về biểu diễn dấu phảy động. Phần lớn các phần thi buổi sáng có câu hỏi về biểu diễn dấu phảy động. Tuy vậy, hầu hết người thi có vẻ yếu về phần này. Phần lớn câu hỏi về số dấu phảy động được đưa ra trong các kỳ thi FE có thể trả lời nếu hiểu rõ về việc chuyển đổi hệ cơ số và số dấu phảy động, do vậy bạn phải nắm được những khái niệm này.

Dấu phảy động là một cách biểu diễn giá trị số học bằng bậc của chúng (một số xấp xỉ kích thước của số đó, kiểu như "hàng nghìn", "hàng triệu") và các chữ số đầu của phần định trị. Trong biểu diễn dấu phảy tĩnh, số chữ số cần biểu diễn (số bit) phụ thuộc vào bậc của giá trị số học, nhưng bên trong máy tính xử lý tất cả các số đều theo chiều dài cố định, do vậy khoảng giá trị số học được biểu diễn bị giới hạn. Mặt khác, khi cần xử lý một khoảng rộng số học, không phải tất cả các chữ số đều quan trọng như nhau, thường các chữ số đầu của phần định trị quan trọng hơn. Do đó người ta chấp nhận sai số để biểu diễn được khoảng số học rộng hơn. Số dấu phảy động là một phương pháp biểu diễn số học đạt được yêu cầu trên, có khả năng biểu diễn một khoảng rộng giá trị số học (với những giá trị tuyệt đối rất lớn và rất nhỏ) bằng một số cố định chữ số (số bit).

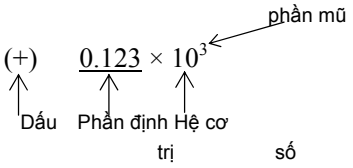
Thí dụ : số thập phân (+)123

Khi di chuyển dấu phảy phần thập phân, ta có :

$$
\begin{aligned}
&: \\
&= 1230 \times 10^{-1} \\
(123)_{10} &= 123 \times 10^{0} \\
&= 12.3 \times 10^{1} \\
&: \\
&= \boxed{0.123 \times 10^{3}} \quad \ldots\ldots [1] \\
&= 0.0123 \times 10^{4} \\
&:
\end{aligned}
$$

Như trên, bằng việc thay đổi vị trí dấu phảy thập phân, một giá trị số học có thể biểu diễn bằng vô số cách. Để thống nhất một chuẩn, người ta thiết lập ra một luật sử dụng để biểu diễn có dạng như [1]. Quá trình này gọi là chuẩn hóa. Chuẩn hóa là việc chuyển đổi để có nhiều chữ số có nghĩa bằng việc chuyển chữ số trái cùng khác 0 về chữ số đầu tiên sau dấu phảy. Căn cứ vào hệ cơ số nào sử dụng (trong trường hợp này là 10), ta có thể xác định giá trị số học bởi dấu, phần định trị và phần mũ của số đó.

Trong thí dụ trên:



Bên trong máy tính, tất cả phần dấu (0 với số dương, 1 với số âm), phần định trị, và phần mũ đều được biểu diễn trong hệ nhị phân. Phần mũ có hai định dạng biểu diễn:

(1) Định dạng dấu phảy cố định (số bù hai)
(2) Biểu diễn dịch (biểu diễn vượt)

Định dạng thứ nhất không được sử dụng thường xuyên, nhưng các câu hỏi liên quan đến định dạng này vẫn có thể xuất hiện trong bài thi. Định dạng thứ hai được sử dụng thực tế trong máy tính. Định dạng này đã được giải thích kỹ trong câu hỏi.
Việc chuyển đổi được thực hiện như sau:

(1) Số thập phân được chuyển sang hệ nhị phân bằng phép chuyển cơ số..
(2) Chuẩn hóa số nhị phân sao cho chữ số đầu tiên khác không là chữ số đầu tiên sau dấu phảy.
(3) Biểu diễn phần mũ theo số bù hai.
(4) Dấu, số mũ và phần định trị lần lượt được đặt vào đúng vị trí trong biểu diễn sau cùng.

Với số thập phân trong thí dụ trên, để ý rằng

$(123)_{10} = (1111011)_2$

Và dạng biểu diễn dấu phảy động của nó là:

$(+)\ (1111011)_2 \times 2^0 = (+)(0.1111011)_2 \times 2^7$

[Câu hỏi con 1]

Bởi vì phần mũ được biểu diễn dưới dạng số bù 2, $(0000011)_2$ = "+3". Do đó, giá trị biểu diễn bởi hình 2 là

$$(+)\ (0.11101)_2 \times 2^3$$
$$= (111.01)_2 \times 2^0$$
$$= 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$
$$= (7.25)_{10}$$

Đáp án đúng cho chỗ trống A là (d), đáp án đúng cho chỗ trống B là (c).

[Câu hỏi con 2]

Từ biểu diễn bit $(1111110)_2$ của phần mũ, ta thấy bit đầu tiên là 1, do đó giá trị phần mũ là âm. Đây là số bù hai 7 bit (định dạng dấu phảy tĩnh), do đó giá trị là "-2". Giá trị được cho trong câu hỏi con này là "$(+)\ (0.0011)_2 \times 2^{-2}$", có thể viết lại như sau.

$(+)\ (0.0011)_2 \times 2^{-2}$

$= (+)\ ((11)_2 \times 2^{-4}) \times 2^{-2}$

$= (+)\ (11)_2 \times 2^{-6}$

$= (+)\ (3)_{10} \times 2^{-6}$

=> Đáp án đúng là (a).

[Câu hỏi con 3]

Như đã giải thích ở trên, chuẩn hóa là thao tác dịch chuyển chữ số khác không đầu tiên bên trái đến vị trí đầu tiên của phần thập phân. Do đó, số đã cho trong câu hỏi con 2 có thể biểu diễn lại như sau.

$(+)\ (0.0011)_2 \times 2^{-2}$

$= (+)\ ((0.11)_2 \times 2^{-2}) \times 2^{-2}$

$= (+)\ (0.11)_2 \times 2^{-4}$

Đáp án đúng sẽ là số nào có phần mũ biểu diễn trong dạng bù hai của "-4". "+4" là $(00000100)_2$, số bù 2 của nó là $(1111100)_2$. Vậy đáp án đúng là (b). Phần mũ trong các lựa chọn khác lần lượt là (a) -5, (c) -3, (d) +1, và (e) +2.

6 Thuật toán

[Phạm vi câu hỏi]
sắp xếp, tìm kiếm, xử lý chuỗi ký tự, xử lý file, biểu đồ, đồ thị, tính toán số, ….

Câu hỏi 1

Q1. Đọc mô tả chương trình và chương trình sau đây, sau đó trả lời các câu hỏi con.

[Mô tả chương trình]

Hàm `RadixConv` là một chương trình chuyển đổi một xâu ký tự số trong hệ `M` (2 <= `M` <= 16) thành một xâu ký tự số trong hệ `N` (2 <= `N` <= 16).

(1) Một xâu ký tự hệ M bao gồm M ký tự số không có ký tự trắng. Trong trường hợp các hệ từ 11-16, các ký tự "A" ~ "F" được sử dụng để ký hiệu các giá trị từ 10 ~ 15.

(2) Hàm `RadixConv` đầu tiên chuyển một xâu ký tự số hệ M thành một số nguyên (integer) sau đó sẽ chuyển sang một xâu ký tự số hệ N. Hàm `MToInt` chuyển một xâu số hệ M thành một số nguyên và hàm `IntToN` chuyển số nguyên thành một xâu số hệ N.

(3) Hàm `MToInt` và hàm `IntToN` sử dụng các hàm sau:

(i) Hàm `ToInt` chuyển một ký tự số `P` ("0", "1", …, or "F") thành một số nguyên.

(ii) Hàm `ToStr` chuyển một số nguyên `Q` (0 <= `Q` <= 15) thành một ký tự số ("0", "1", …, "F")

(iii) Hàm định sẵn `Length` trả về chiều dài của một chuỗi.

(iv) Hàm định sẵn `Substr` lấy ra một phần của chuỗi.

(4) Các bảng từ 1 đến 5 dưới đây đưa ra danh sách các đối số và giá trị trả về của các hàm.

Bảng 1 `RadixConv`

Đối số/ Giá trị trả về	Kiểu dữ liệu	Ý nghĩa
`Frdx`	Integer	Cơ số của chuỗi số trước khi chuyển đổi (2<= `Frdx` <= 16)
`Fnum`	Char	Chuỗi số trước khi chuyển
`Trdx`	Integer	Cơ số của chuỗi số sau khi chuyển (2 <= `Trdx` <= 16)
Giá trị trả về	Char	Chuỗi số sau khi chuyển trong hệ `Trdx`

Bảng 2 `MToInt`

Đối số/ Giá trị trả về	Kiểu dữ liệu	Ý nghĩa
`Rdx`	Integer	Cơ số của chuỗi số trước khi chuyển đổi (2 <= `Rdx` <= 16)
`Num`	Char	Chuỗi số trước khi chuyển
Giá trị trả về	Integer	Số nguyên sau khi chuyển

Bảng 3 `IntToN`

Đối số/ Giá trị trả về	Kiểu dữ liệu	Ý nghĩa
`Val`	Integer	Số nguyên trước khi chuyển
`Rdx`	Integer	Cơ số của chuỗi số sau khi chuyển đổi (2 <= `Rdx` <= 16)
Giá trị trả về	Char	Chuỗi số sau khi chuyển về hệ cơ số `Rdx`

Bảng 4 `ToInt`

Đối số/ Giá trị trả về	Kiểu dữ liệu	Ý nghĩa
`P`	Char	Các ký tự số trước khi chuyển đổi ("0", "1", …, or "F")
Giá trị trả về	Integer	Số nguyên sau khi chuyển

Bảng 5 `ToStr`

Đối số/ Giá trị trả về	Kiểu dữ liệu	Ý nghĩa
`Q`	Integer	Số nguyên trước khi chuyển đổi (0 <= `Q` <= 15)
Giá trị trả về	Char	Ký tự số sau khi chuyển đổi.

[Chương trình]

```
O char_type: RadixConv (int_type: Frdx, char_type: Fnum, int_type: Trdx)
• return IntToN(MToInt(Frdx, Fnum), Trdx)
                                    /* Lấy giá trị của IntToN là giá trị trả về của hàm */

O int_type: MToInt(int_type: Rdx, char_type: Num)
O int_type: Idx, Val
• Val ← 0
■ Idx: 1, Idx <= Length(Num), 1  /* Hàm Length trả về độ dài của chuỗi Num */
  · Val ← [ A ] + ToInt(Substr(Num, Idx, 1))
■   /* Hàm Substr trả về Idx(>= 1) ký tự bắt đầu từ vị trí Num */
• return Val                    /* Lấy Val là giá trị trả về của hàm */

O char_type: IntToN(int_type: Val, int_type: Rdx)
O int_type: Quo                 /* Quotient: Thương số */
O int_type: Rem                 /* Remainder: Số dư */
O char_type: Tmp
• [ B ]
• Tmp ← ""
■ Quo >= Rdx
  • Rem ← Quo % Rdx
  • Tmp ← ToStr(Rem) + Tmp/* + là toán tử nối chuỗi */
  • [ C ]
■
• [ D ]
• return Tmp                    /* Lấy Tmp là giá trị trả về của hàm */

O int_type: ToInt(char_type: P)
O int_type: Idx
O char_type: Code[16]           /* Chỉ số bắt đầu từ 0 */
/* Code chứa các giá trị khởi tạo "0" , "1" , "2" , "3" , "4" , "5" , "6" , "7" ,    */
/* "8" , "9" , "A" , "B" , "C" , "D" , "E" , "F"  theo thứ tự này                    */
/* Giá trị các ký tự được tăng dần                                                    */
• Idx ← 0
■ [ E ]                         /* So sánh chuỗi */
  • Idx ← Idx + 1
■
• return Idx                    /* Lấy Idx là giá trị trả về của hàm */

O char_type: ToStr(int_type: Q)
O char_type: Code[16]           /* Chỉ số bắt đầu từ 0 */
/* Code chứa các giá trị khởi tạo  "0" , "1" , "2" , "3" , "4" , "5" , "6" , "7" ,   */
/* "8" , "9" , "A" , "B" , "C" , "D" , "E" , "F"  theo thứ tự này                    */
/* Giá trị các ký tự tăng dần theo thứ tự này                                         */
• return Code[Q]                /* Lấy Code[Q] là giá trị trả về của hàm */
```

Câu hỏi con

Từ các nhóm đáp án phía dưới, chọn đáp án đúng để điền vào các ô trống ☐ trong chương trình trên.

Nhóm đáp án cho A:

a) `Rdx`
b) `Val`
c) `Val * Rdx`
d) `Val / Rdx`

Nhóm đáp án cho B, C và D:

a) `Quo ← Quo / Rdx`
b) `Quo ← Quo / Rem`
c) `Quo ← Rdx`
d) `Quo ← Rem / Rdx`
e) `Quo ← Val`
f) `Rem ← Rdx`
g) `Rem ← Val`
h) `Tmp ← ToStr(Quo) + Tmp`
i) `Tmp ← ToStr(Rem) + Tmp`

Nhóm đáp án cho E:

a) `P < Code[Idx]`
b) `P > Code[Idx]`
c) `P <= Code[Idx]`
d) `P >= Code[Idx]`

Đáp án câu 1

Chương trình chuyển đổi cơ số (Giả ngôn ngữ)

Đáp án đúng

A – c, B – e, C – a, D – h, E – b

Chú thích

Trong [Chương trình], có 5 hàm con được đưa ra tuần tự. Dạng câu hỏi này khác so với các câu hỏi trước, nên cần được chú ý. Tuy nhiên, điều quan trọng là tiếp cận chương trình bằng cách chia nó thành các phần nhỏ. Cụ thể cách tiếp cận với dạng câu hỏi này như sau.

Mọi chương trình, không quan tâm nó lớn như thế nào, đều được tạo ra bởi sự kết hợp của các hàm cần thiết nhỏ hơn để hoàn thành chức năng của toàn bộ hàm. Một chương trình có chức năng càng phức tạp thì càng cần chia thành các hàm con, từ các hàm con tạo nên một chương trình lớn. Tuy nhiên, nếu bạn chỉ tập trung vào các hàm con thì bạn chỉ cần suy nghĩ về nó mà thôi không quan tâm tới kích thước toàn bộ chương trình. Thông thường, khi nghĩ về một câu hỏi phức tạp, gợi ý là bắt đầu bằng việc bỏ đi nguyên nhân gây ra sự phức tạp. Khi kích thước là nguyên nhân thì bạn nên chia thành các phần nhỏ hơn. Khi một hàm là phức tạp, bạn nên nghĩ tới các hàm nhỏ hơn cần thiết kết hợp với nhau để hoàn thành chức năng của hàm đó. Tuy nhiên chúng ta đã được học trong mục liên quan tới tính chất cố kết và kết nối giữa các module, thiết kế chương trình là cân nhắc lợi hại khi chia thành các module (Các hàm nhỏ hơn) có tính độc lập cao so với nhau. Trong các câu hỏi thi có các chương trình gần như ngược lại với nguyên tắc thiết kế, bạn không phải lo lắng nhiều về điều đó.

Đầu tiên, cần phải hiểu chức năng của toàn bộ chương trình. Tại điểm bắt đầu của [Mô tả chương trình], chức năng đã được mô tả rõ ràng như sau: "Hàm `RadixConv` là một chương trình chuyển chuỗi số hệ M thành chuỗi số hệ N". Chúng ta nên chắc chắn nhớ chức năng này trong đầu khi trả lời các câu hỏi. Nhưng đồng thời cẩn thận đừng quá tập trung vào toàn bộ chương trình.

Tiếp theo, chúng ta cố gắng hiểu tổng quát các hàm nhỏ. Mỗi câu hỏi luôn luôn có gợi ý, bạn phải tìm ra nó. Nếu một chương trình được phân chia rõ ràng thành các thủ tục hoặc các hàm nhỏ hơn, gợi ý thường ở trong các tham số hoặc giá trị trả về. Hãy nhớ chức năng của toàn bộ chương trình, sau đây chúng ta sẽ xem xét tổng quát mỗi hàm.

- **`RadixConv`** (Bảng 1)

  Chính hàm này là mục đích của toàn bộ chương trình lớn. Chúng ta hãy xem cách tổ chức và các thành phần liên quan. Các tham số là cơ số của chuỗi ký tự trước khi chuyển (Frdx), chuỗi số trước khi chuyển (Fnum) và cơ số của chuỗi số sau khi chuyển (Trdx). Giá trị trả về được chuyển về dạng chuỗi số. Chúng ta thấy rằng phần này là chuyển một chuỗi số (Fnum) trong hệ cơ số M (Giá trị của Frdx) thành chuỗi trong hệ cơ số N và trả về giá trị chuỗi số là kết quả của việc chuyển đổi. Đây chính là mục đích chính của toàn bộ chương trình.

- **`MToInt`** (Bảng 2)

  Các tham số gồm có cơ số của chuỗi số trước khi chuyển (Rdx) và chuỗi số trước khi chuyển (Num); giá trị trả về là số nguyên đã chuyển. Do đó đây là hàm để chuyển một xâu số (Num) thành một số nguyên (Giá trị số). Do kiểu dữ liệu của các tham số là xác định, chuỗi số là thuộc kiểu kí tự. Thông thường, các chuỗi số không thể được sử dụng cho tính toán, vì vậy các phép tính toán cần chuyển các chuỗi số thành các giá trị số tương ứng. Mặc khác, tên của hàm "MToInt" cũng gợi ý "Chuyển từ M thành số nguyên", nên có thể đoán được chức năng của hàm từ tên của nó.

- **`IntToN`** (Bảng 3)

  Các tham số gồm có số nguyên trước khi chuyển (Val), cơ số của chuỗi số sau khi chuyển (`Rdx`), và giá trị trả về là chuỗi số sau khi chuyển. Dựa trên các thông tin này thì đây là hàm chuyển số nguyên trước khi chuyển (Val) thành một chuỗi số trong hệ cơ số N. Tên "IntToN" có thể hiểu nghĩa là "Chuyển từ số nguyên thành N".

- **`ToInt`** (Bảng 4)

  Tham số là một ký tự số (`P`) trước khi chuyển, và giá trị trả về là số nguyên sau khi chuyển. Ký tự số (`P`) được chuyển thành một số nguyên. Chú ý rằng ký tự số được coi như là một ký tự (Chuỗi).

- **`ToStr`** (Bảng 5)

  Tham số là số nguyên (`Q`) trước khi chuyển, và giá trị trả về là một ký tự số đã chuyển. Hàm chuyển một số nguyên (`Q`) thành một ký tự số. Bạn có thể lo lắng bởi việc không có cơ số trong chuyển đổi. Tuy nhiên, dù bạn sử dụng cơ số nào đi nữa, giá trị tương ứng với ký tự số là giống nhau. Cơ số được sử dụng để quyết định phạm vi biểu diễn của một giá trị số mà tham số Q nắm giữ. Ví dụ, trong hệ cơ số 2, có các giá trị 0 hoặc 1; trong hệ cơ số 16, có phạm vi từ 0 tới 15. Trong các trường hợp khác, các giá trị số giống nhau tương ứng với một số.

  Khi kiểm tra các hàm này, bạn có lẽ phân vân về cơ số sử dụng cho số nguyên trước và sau khi chuyển. Tuy nhiên, các giá trị số là giá trị không phụ thuộc vào cơ số. Khi chúng ta nói "hệ cơ số n", chúng ta nói tới hệ cơ số mà trong đó ta biểu diễn các giá trị số (Nếu bạn phải nghĩ về cơ số trong trường hợp này, chọn 2 bởi vì mọi thứ trong máy tính được biểu diễn trong hệ nhị phân). Khi chuyển một chuỗi số trong hệ 7 sang hệ 9, chúng ta chuyển tạm thời sang hệ 10; tuy nhiên, chúng ta không chuyển thực tế sang hệ 10 mà là chuyển thành một giá trị số. Hệ đếm dễ dàng nhất cho chúng ta là hệ cơ số 10, vì thế chúng ta chuyển số sang hệ cơ số 10 để hiểu giá trị của nó.

Ô trống A:

Hàm "`MToInt`" là hàm chuyển một chuỗi số trong hệ cơ số M thành một số nguyên. Từ dòng cuối "`return Val`" chúng ta nhìn thấy giá trị số nguyên sau khi chuyển đổi được lưu trữ trong Val. Bản thân hàm là một vòng lặp của quá trình có điều kiện là "Idx <= Length(Num)". Giá trị khởi tạo của Idx là 1; "Length (`Num`)" chỉ ra độ dài của chuỗi số Num, và được tăng dần lên 1 đơn vị, do đó quá trình lặp có số lần lặp bằng số các ký tự. Hàm "Substr(Num, Idx, 1)" ở trong vòng lặp dùng để "Lấy ra một ký tự tại vị trí thứ `Idx` (Tính từ đầu chuỗi) của chuỗi Num". Chương trình lấy một ký tự tại một thời điểm từ đầu của chuỗi số `Num` và cộng giá trị vào biến `Val`, biến chứa kết quả của quá trình chuyển đổi.

Việc chuyển đổi một số hệ M thành một số giá trị nguyên tương đương với việc chuyển sang

hệ 10. Nhớ lại cách thực hiện chuyển đổi. Ví dụ, xem xét một số nhị phân 101. Tính toán chuyển đổi $1\times2^2(=4) + 0\times2^1(=0) + 1\times2^0(=1)$, cho giá trị 5 (trong hệ 10). Nói cách khác, chúng ta lấy giá trị số của mỗi ký tự số nhân với trọng số tương ứng với vị trí của bit đó và tính ra kết quả. Nhưng nhóm trả lời lại có vẻ như không chứa cách này. Chú ý là $1\times2^2 + 0\times2^1 + 1\times2^0 = ((1\times2)+0)\times2+1$, vì thế cách sau cùng là cách được dùng ở đây. "2" tương ứng với `M` trong hệ cơ số `M`, vì thế chúng ta có thể nhân `Val` với `Rdx` (Biến chứa giá trị `M`) và cộng giá trị số tại mỗi vị trí các bit. Bởi vậy, đáp án đúng là (c). Nhớ lại là hàm `ToInt` dùng để chuyển một ký tự số thành một số nguyên (giá trị).

Nếu bạn chú ý là $1\times2^2 + 0\times2^1 + 1\times2^0 = ((1\times2) + 0)\times2+1$, câu trả lời sẽ không khó để tìm ra. Thậm chí nếu bạn không chú ý điều đó, bạn cũng không cần phải bỏ qua việc tìm ra câu trả lời. Sử dụng thực tế là đáp án đúng nằm trong nhóm các đáp án. Khi tham số `Rdx` = 2 và `Num` = "101" được dùng cho hàm này, có thể biết `Val` là 5. Khi gán mỗi lựa chọn trả lời vào trong ô trống và kiểm tra chương trình, đáp án nào cho kết quả đúng là đáp án đúng. Trong ví dụ, `Num` có 3 ký tự số nên quá trình thực hiện 3 lần như sau: ([①] ~ [③]).

a) : ① `Val` = 2 + 1 = 3 → ② `Val` = 2 + 0 = 2 →③ `Val` = 2 + 1 = **3**

b) : ① `Val` = 0 + 1 = 1→② `Val` = 1 + 0 = 1 →③ `Val` = 1 + 1 = **2**

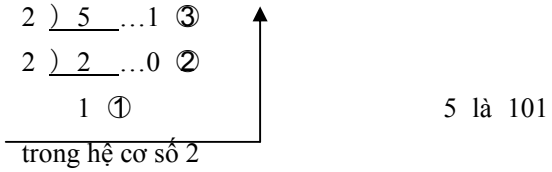
c) : ① `Val` = 0 × 2 + 1 = 1 →② `Val` = 1 × 2 + 0 = 2 →③ `Val` = 2 × 2 + 1 = **5**

d) : ① `Val` = 0 / 2 + 1 = 1 →② `Val` = 1 / 2 + 0 = 0.5 →③ `Val` = 0.5 / 2 + 1 = **1.25**

Khi bạn biết giá trị của kết quả đúng, như trong trường hợp này, thử mỗi đáp án để tìm ra kết quả là một phương pháp hiệu quả. Vì việc thao tác $1\times2^2 + 0\times2^1 + 1\times2^0 = ((1\times2) + 0)\times2+1$ là một cách hiệu quả để giảm khối lượng tính toán, và nó xuất hiện trong cả bài kiểm tra buổi sáng lẫn buổi chiều.

Các ô trống B - D:

Nhớ lại là hàm `IntToN` dùng để chuyển một số nguyên thành chuỗi số trong hệ N. Hàm này dài hơn một chút so với hàm `MToInt`, và do đó có nhiều ô trống hơn, nhưng đừng lo lắng về điều đó. Hãy tự tin vào bản thân. Đầu tiên, chú ý giá trị trả về là chuỗi số đã chuyển, nội dung của nó thấy trong biến `Tmp`. Xét về mặt cấu trúc, quá trình lặp là phần chính, và chúng ta nhìn thấy là chương trình nối các ký tự (số) trong biến `Tmp` trong khi số dư thu được ở mỗi bước. Giữ điều điều đó trong đầu, nhớ cách thực hiện chuyển cơ số. Ví dụ, cách để chuyển giá trị số 5 sang hệ nhị phân, số được lặp đi lặp lại việc chia cho 2 trong khi lưu trữ số dư tại mỗi lần chia.



Không nên chỉ nhìn vào chương trình viết bằng giả ngôn ngữ. Nó rất quan trọng khi bạn nhớ các ví dụ như ví dụ trên.

Với ô trống B, bạn có thể tìm ra đó có thể là sự khởi tạo biến, xem xét vị trí tương đối của nó với câu sau (`Tmp` ← ""). Trong khởi tạo biến, điều quan trọng là xác định biến cần được khởi

tạo cho mục sau đó. Điều kiện của vòng lặp là "Quo >= Rdx", và Rdx là một tham số, do đó giá trị của nó phải có rồi. Nhưng "Quo" (thương số) là một biến được định nghĩa trong hàm này, vì thế khởi tạo giá trị của nó trước khi so sánh là cần thiết. Vậy đây là câu lệnh khởi tạo cho biến Quo.

Tiếp đó, xem xét biến Quo. Tại thời điểm ban đầu của vòng lặp, chúng ta thấy "Rem ← Quo % Rdx"; số dư thu được khi chia Quo cho Rdx chứa trong biến Rem. ("%" là toán tử Môđun, nhưng bạn có thể tìm ra nếu bạn không biết điều đó bởi vì tên Rem có nghĩa là số dư). Chú ý điều này và ví dụ trong hình 1, bạn có thể thấy đây có thể là một phần trong việc thực hiện chuỗi biến đổi 5 → 2. Nếu là trường hợp này, lời chú thích "thương số" cho biến cũng đáng chú ý. Giá trị khởi tạo của nó là số nguyên trước khi chuyển, do đó đáp án đúng là (e) Quo ← Val. Chú ý là tham số Val, là số nguyên trước khi chuyển, không được sử dụng ở chỗ nào khác. Tham số này là giá trị cần thiết nhưng không tìm thấy ở đâu trong hàm. Điều đó có nghĩa là giá trị của nó được truyền cho một biến khác. Đó chính là một dấu hiệu cho bạn chọn đáp án đúng.

Với ô trống C, xem xét nội dung của vòng lặp. Đầu tiên, "Rem ← Quo % Rdx" gán giá trị số dư cho biến Rem, kết quả của phép chia cho Rdx là "N" trong hệ cơ số N. Câu lệnh tiếp theo "Tmp ← ToStr(Rem) + Tmp" nối số dư vào biến Tmp sau khi chuyển thành một số. Bây giờ Tmp là một biến chứa chuỗi số kết quả. Trong lần lặp tiếp theo, giá trị của Quo cần được thay đổi nên phần tương ứng với sự thay đổi này phải được thực hiện ở đây. Trong ví dụ ở hình 1, đó là nơi diễn ra "5 → 2". Trong vòng lặp tiếp theo, giá trị của Quo phải là thương số thu được khi lấy số chia cho Rdx, nhưng thương số này bị bỏ qua. Thậm chí nếu bạn không tìm thấy nơi nào mà giá trị thương số này được lưu lại. Do đó, cần thiết để tìm thương số ở đây và thay giá trị vào biến Quo. Vì thế câu trả lừoi đúng là (a) Quo ← Quo / Rdx.

Với ô trống D, phần còn lại của vòng lặp phải được thực hiện. Sử dụng ví dụ trên, biến Tmp bây giờ chứa chuỗi số tương ứng với thủ tục ② ③ của hình 1, (i.e., "01") cho tới thời điểm hiện tại. Vì thế kết quả của bước cuối cùng ① phải được nối vào và đây là thương số không phải là số dư. Mặt khác, điều kiện dừng của vòng lặp chia là thương số (1) nhỏ hơn số chia (2). Điều này cũng phù hợp với điều kiện lặp của hàm này. Do đó, chúng ta thấy rằng tại điểm này cần thiết để chuyển thương số thành một số và nối vào trước trong chuỗi ký tự. Đáp án đúng là (h) Tmp ← ToStr(Quo) + Tmp. Ngoài ra còn có dấu hiệu khác cho ô trống này ở cấu trúc chương tình. Điều duy nhất xảy sau ô trống là đưa ra giá trị trả về. Vì thế không quan trọng giá trị biến nào thay đổi đều không ảnh hưởng tới giá trị trả về trừ biến Tmp. Điều này gợi ý là dòng này liên quan tới biến Tmp, từ đó giới hạn các lựa chọn cho đáp án đúng còn (h) và (i). Sau đó, bạn phải tìm cách xác định đáp án đúng.

Ô trống E:

Nhớ lại hàm ToInt chuyển tham số của nó, một ký tự số, thành một số nguyên. Từ các lựa chọn trả lời và tên của biến Idx, bạn có thể đã tìm ra một vài bảng tìm kiếm có liên quan. Vì giá trị trả về là Idx, bạn nên biết rằng chỉ số (Idx) được trả về khi nào mà bảng được tìm kiếm tuần tự và điều kiện tìm kiếm được thỏa mãn. Nhóm đáp án được chia ra bởi điều kiện cho việc tiếp tục vòng lặp: Trong khi tham số P nhỏ hơn (a, c) hoặc lớn hơn (b, d). Đầu tiên, chúng ta quyết định xem điều kiện nào. Bảng "Code" đang tìm kiếm có các ký tự "0" tới "F" theo thự tự tăng dần của mã ký tự. Giá trị khởi tạo Idx = 0, nên quá trình tìm kiếm bắt đầu từ thành phần đầu tiên. Mặt khác, trong quá trình tìm kiếm rất khó khăn để giữ điều kiện khi P nhỏ hơn. Bây giờ chúng ta biết rằng toán tử đúng là > hoặc >=, chúng ta chỉ phải quyết định nếu có = hay không; điều đó cho chúng ta câu trả lời đúng. Ví dụ, khi các tham số là "0", chúng ta có Idx = 0. Nhưng khi điều

kiện là (d) "`P >= Code[Idx]`", phép so sánh đầu tiên đã thỏa mãn điều kiện vì thể `Idx` được cộng vào, kết quả là `Idx` = 1. Bởi vậy câu trả lời đúng sẽ không có dấu =, và do đó đáp án đúng là (b) `P > Code[Idx]`.

Tại đầu câu hỏi này, chúng ta nhìn bao quát tất cả các hàm; tuy nhiên, trong bài thi thực tế, bạn nên nhớ lấy các ô trống và theo dòng chương trình khi cần thiết phải thực hiện.

Bạn có thể thấy là câu hỏi này chứa nhiều ví dụ nhắc bạn nhớ đến các tính chất của chương trình trong ngôn ngữ C, ví dụ sử dụng giá trị trả về của hàm mà không thay thế chúng vào các biến. Các thuật ngữ như "`ToInt`" và "`ToStr`" thì rõ ràng như C. Xu hướng này sẽ còn tiếp tục được sử dụng, vì thế mọi người nên có các kiến thức cơ bản về ngôn ngữ C. Tuy nhiên, bạn không bắt buộc biết rõ về cú pháp hoặc có khả năng viết chương trình với ngôn ngữ C.

Câu hỏi 2

Q2. Đọc mô tả chương trình, giải thích cú pháp cho giả ngôn ngữ, và bản thân chương trình dưới đây, sau đó trả lời các câu hỏi con 1 và 2.

[Mô tả chương trình]

Đây là chương trình con `QuickSort`, chương trình sắp xếp các số nguyên từ `A[Min]` đến `A[Max]` (0 <= `Min` < `Max`) trong mảng một chiều `A`.

(1) Thủ tục sắp xếp như sau.

(i) Chương trình tìm kiếm tuần tự một phần tử trong mảng từ `A[Min+1]` đến `A[Max]` , phần tử này có giá trị khác so với giá trị của `A[Min]`, lấy phần tử tìm thấy đầu tiên thỏa mãn so sánh với A[`Min`], và chọn bất cứ giá trị nào lớn hơn làm giá trị tham chiếu (`Pivot`). Nếu tất cả các phần tử của mảng là giống nhau, quá trình sắp xếp kết thúc. Chương trình con `FindPivot` được sử dụng để chọn giá trị tham chiếu.

(ii) Các phần tử được sắp xếp lại sao cho tất cả các phần tử nhỏ hơn `pivot` là `A[Min]`, …, `A[i-1]` (`Min` < `i` <= `Max`) và tất cả các phần tử lớn hơn hoặc bằng giá trị `pivot` là `A[i]`, …, `A[Max]`. Chương trình con `Arrange` thực hiện quá trình này.

(iii) Việc sắp xếp lại các phần tử của (`A[Min]`, …, `A[i-1]` và `A[i]`, …, `A[Max]`) được xem như là hai mảng mới và sắp xếp bằng cách áp dụng đệ quy `QuickSort`.

(2) Chi tiết tham số cho các chương trình con được đưa ra trong các bảng dưới đây.

Bảng 1 Tham số hàm `QuickSort`

Tên biến	Đầu vào/Đầu ra	Ý nghĩa
`A`	Vào/Ra	Mảng một chiều cần sắp xếp
`Min`	Vào	Chỉ số phần tử đầu tiên trong phạm vi sắp xếp
`Max`	Vào	Chỉ số phần tử cuối cùng trong phạm vi sắp xếp

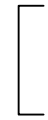
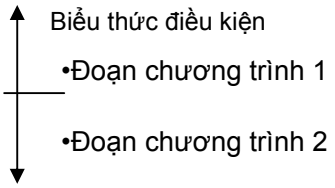
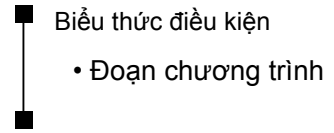
Bảng 2 Tham số hàm `FindPivot`

Tên biến	Đầu vào/Đầu ra	Ý nghĩa
`A`	Vào	Mảng một chiều cần sắp xếp
`Min`	Vào	Chỉ số phần tử đầu tiên trong phạm vi sắp xếp
`Max`	Vào	Chỉ số phần tử cuối cùng trong phạm vi sắp xếp
`Ret`	Ra	Trả về chỉ số của phần tử chứa giá trị pivot. Tuy nhiên trả về "`-1`" nếu các phần tử `A [Min]`, …, `A Max]` có cùng giá trị.

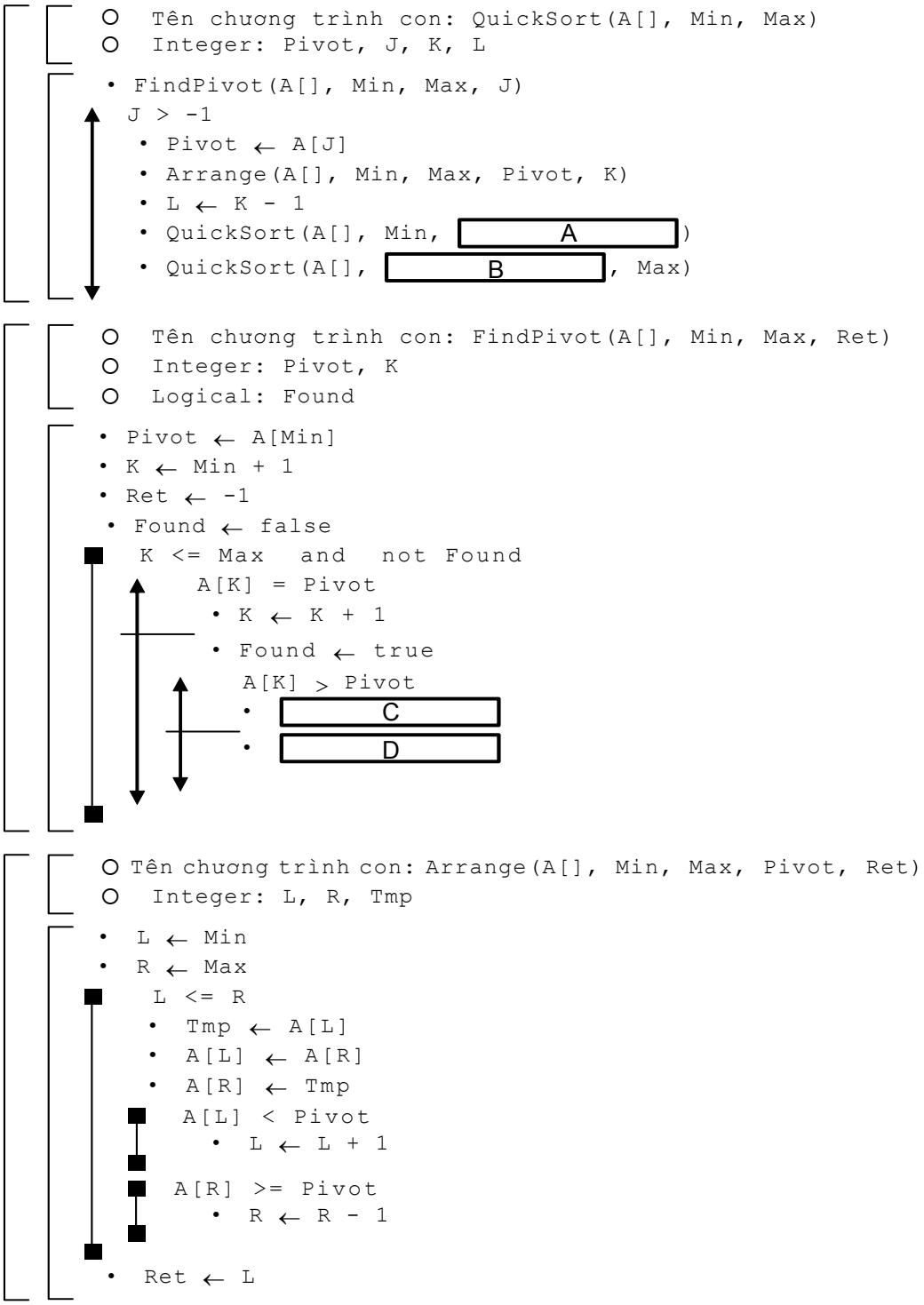
Bảng 3 Tham số hàm `Arrange`

Tên biến	Đầu vào/Đầu ra	Ý nghĩa
`A`	Vào/Ra	Mảng một chiều cần sắp xếp
`Min`	Vào	Chỉ số phần tử đầu tiên trong phạm vi sắp xếp
`Max`	Vào	Chỉ số phần tử cuối cùng trong phạm vi sắp xếp
`Pivot`	Vào	Giá trị tham chiếu
`Ret`	Ra	Sắp xếp lại các phần tử sao cho giá trị của `A[Min]`, …, `A[i-1]` nhỏ hơn `Pivot` và giá trị của `A[i]`, … `A[Max]` là lớn hơn hoặc bằng `Pivot`, và trả về giá trị của `i`.

[Giải thích cú pháp giả ngôn ngữ]

Cú pháp	Giải thích
	Một vùng liên tiếp nơi khai báo và thực hiện được miêu tả.
○	Khai báo tên, kiểu và các thuộc tính khác của thủ tục và biến …
• Biến ← Biểu thức	Gán giá trị của biểu thức vào biến.
{Chú thích}	Ghi chú thích.
	Thể hiện lựa chọn. Khi biểu thức điều kiện là đúng (true), đoạn chương trình 1 được thực hiện, khi là sai (false) thì đoạn chương trình 2 được thực hiện.
	Thể hiện lặp với điều kiện kết thúc ở trên đầu. Khi biểu thức điều kiện là đúng (true) thì đoạn chương trình được thực hiện.

[Chương trình]



Câu hỏi con 1

Từ các nhóm đáp án dưới đây, chọn đáp ánđúng để điền vào các ô trống ☐ trong chương trình ở trên.

Nhóm đáp án cho A và B:

a) `K` b) `L` c) `Max` d) `Min`

Nhóm đáp án cho C và D:

a) `Ret ← A[K]` b) `Ret ← A[Max]` c) `Ret ← A[Min]`
d) `Ret ← K` e) `Ret ← Max` f) `Ret ← Min`

Câu hỏi con 2

Nội dung của các tham số đã được tổng hợp trong bảng 4 sau khi sử dụng `QuickSort` để sắp xếp các phần tử từ `A[0]` tới `A[9]` của mảng số nguyên một chiều. Từ các nhóm đáp án dưới đây, chọn đáp án đúng để điền vào ô trống ☐ trong bảng sau đây.

Bảng 4 Số lần gọi và nội dung các tham số của hàm `QuickSort`

Số lần gọi	A	Min	Max
vòng đầu tiên	A[0] … A[9]: 3 5 8 4 0 6 9 1 2 7	0	9
vòng thứ 2	3 2 1 4 0 6 9 8 5 7	0	4
vòng thứ 3	E	F	G
⋮	⋮	⋮	⋮
vòng thứ n	0 1 2 3 4 5 6 7 8 9	9	9

Nhóm đáp án cho E:

a) 0 1 2 4 3 6 9 8 5 7

b) 0 2 1 3 4 6 9 8 5 7

c) 0 2 1 4 3 6 9 8 5 7

d) 0 4 1 2 3 6 9 8 5 7

Nhóm đáp án cho F và G:

a) 0 b) 1 c) 2
d) 3 e) 4 f) 5

Đáp án câu 2

Sắp xếp nhanh (Giả ngôn ngữ)

Đáp án đúng

[Câu hỏi con 1] A – b, B – a, C – d, D – f

[Câu hỏi con 2] E – c, F – a, G – c

Giải thích

Sắp xếp nhanh (Quick sort) xuất hiện trong bài kiểm tra buổi sáng, khi đó nó được định nghĩa theo cách sau: "Đây là một cách sắp xếp dữ liệu trong đó toàn bộ tập dữ liệu đầu tiên được chia thành hai nhóm, một nhóm nhỏ hơn giá trị đang xét và một nhóm lớn hơn hoặc bằng giá trị này. Quá trình phân nhóm tiếp tục được áp dụng cho mỗi nhóm cho tới khi toàn bộ dữ liệu được sắp xếp. Phương pháp này hiệu quả khi số lượng phép so sánh và chuyển đổi là nhỏ, tuy nhiên nhiều người vẫn có cảm giác là quá trình đệ quy tự nhiên gây khó khăn cho việc quản lý. Ở nhiều kỳ thi, khi họ nhìn thấy câu hỏi này trong bài thi, họ tự nghĩ, "Ôi không! Tôi biết rằng tôi nên học thuật toán quick sort". Ghi nhớ, mặc dù các thí sinh bị bẫy bởi các câu hỏi loại này khi họ suy nghĩ theo cách như vậy. Nhưng các câu hỏi thi được thiết kế với ý định để cho thí sinh dự thi với các kiến thức cơ bản thông thường có thể trả lời, vì thế bạn cần giữ được bình tĩnh khi trả lời các câu hỏi này.

[Câu hỏi con 1]

Đây là câu hỏi điền vào chỗ trống, nhưng bất cứ khi nào một thủ tục cụ thể được miêu tả trong phần [Mô tả chương trình] giống như câu hỏi này, tốt nhất là đầu tiên nhìn tổng quát chương trình và xem xét câu hỏi phù hợp với [Mô tả chương trình]. Giống như chủ đề câu hỏi "Quick Sort", bất cứ khi nào một câu hỏi liên quan tới một thuật toán mà không ai có thể dễ dàng giả quyết, câu hỏi có xu hướng xuất hiện theo cách này.
Dựa trên [Mô tả chương trình], thủ tục sắp xếp bởi chương trình con `QuickSort` là như sau:

(i) Sử dụng chương trình con `FindPivot` để tìm ra giá trị tham chiếu (`Pivot`).
(ii) Sử dụng chương trình con `Arrange` để tái sắp xếp các phần tử trong mảng (`A[Min],…, A[i - 1]` nhỏ hơn `Pivot` và `A[i],…, A[Max]` lớn hơn hoặc bằng giá trị `Pivot`).
(iii) Xem xét `A[Min],…, A[i - 1]` và `A[i],…, A[Max]` như là các mảng mới và áp dụng đệ quy thủ tục `QuickSort`.

Đầu tiên, chúng ta hãy chú ý tới tên của các chương trình con sử dụng trong mục này và xem xét sự phù hợp của chúng với chương trình con `QuickSort`; không khó để xác định như sau.

```
○ QuickSort(A[], Min, Max)
○ Integer: Pivot, J, K, L
·  FindPivot(A[], Min, Max, J)                    …… (i)
J > -1
 ·  Pivot ← A[J]
 ·  Arrange(A[], Min, Max, Pivot, K)              …… (ii)
 ·  L ← K - 1
 ·  QuickSort(A[], Min, [     A     ])
 ·  QuickSort(A[], [     B     ], Max)            …… (iii)
```

Ô trống A và B:

Chúng ta tiếp tục phần giải thích với giả sử là chúng ta biết lời gọi đệ quy thủ tục `QuickSort` chứa 2 ô trống này, tương ứng với (iii). Bảng 1 đưa ra danh sách các tham số của `QuickSort`; có 3 tham số là theo thứ tự "`A` (mảng để sắp xếp), `Min` (chỉ số của phần tử đầu tiên trong phạm vi sắp xếp), và `Max` (chỉ số của phần tử cuối cùng trong phạm vi sắp xếp)". Xem xét cái để chèn vào các ô trống này, đưa ra nội dung các tham số này. Ô trống A phải là chỉ số của phần tử cuối cùng trong phạm vi sắp xếp, tương ứng với "`i - 1`" trong (iii) của mô tả. Ô trống B là chỉ số của phần tử đầu tiên trong phạm vi sắp xếp, tương ứng với "`i`" trong (iii). Bây giờ, chú ý hai giá trị, "`i - 1`" và "`i`", chúng ta đọc trong (ii) của mô tả, "tất cả các phần tử nhỏ hơn `Pivot` là `A[Min]`, …, `A[i - 1]` và tất cả các phần tử bằng hoặc lớn hơn `Pivot` là `A[i]`, …, `A[Max]`". Vì thế, "`i - 1`" là chỉ số của phần tử cuối cùng nhỏ hơn `Pivot` trong khi "`i`" là chỉ số của phần tử đầu tiên lớn hơn hoặc bằng `Pivot`. Các phần tử của mảng để sắp xếp `A[Min]`, …, `A[Max]`, được đưa ra theo thứ tự này tại thời điểm bắt đầu. Bằng `FindPivot` của (i), một giá trị `Pivot` được chọn, và bằng `Arrange` trong (ii), các phần tử được sắp xếp. Nhìn vào các tham số của hàm `Arrange` trong bảng 3. Có 5 tham số, nhưng tham số cuối cùng `Ret` nói rằng "Tái sắp xếp các phần tử sao cho giá trị `A[Min]`, …, `A[i - 1]` là nhỏ hơn `Pivot` và giá trị `A[i]`, …, `A[Max]` là lớn hơn hoặc bằng `Pivot`." Trong chương trình, `K` được sử dụng thay cho `Ret` (chú ý "`Arrange(A[],Min,Max,Pivot,K)`"), do đó giá trị `K` này tương ứng với `[i]` chỉ tới trong (ii). Do đó ô trống A có thể là "`K - 1`", nhưng trong nhóm trả lời không có lựa chọn này. Nhìn trở lại chương trình, chúng ta thấy rằng có một dòng `L ← K – 1` ngay sau khi gọi `Arrange`. Vì vậy `L = K – 1`, và đáp án đúng cho ô trống A là (b) `L`. Bây giờ đó là dấu hiệu cho thấy đáp án đúng cho ô trống B phải là (a) `K`.

Ô trống C và D:

Các ô trống này nằm trong chương trình con `FindPivot`, đã giải thích trong [Mô tả chương trình] (i). Mô tả (i) hơi dài, nhưng nội dung của nó có thể được tóm tắt như sau: " Tìm kiếm tuần tự trong mảng từ `A[Min+1]` đến `A[Max]` một phần tử có giá trị khác với `A[Min]` (Bao gồm tát cả các giá trị ngoại trừ `A[Min]`), lấy phần tử đầu tiên tìm thấy, so sánh với `A[Min]`, và chọn bất cứ giá trị nào lớn hơn làm giá trị tham chiếu (Chọn `Pivot`) ". Bây giờ xem xét các nhóm trả lời. Chú ý là tất cả các lựa chọn đều thiết lập giá trị cho `Ret`. Quan tâm tới biến `Ret`, bảng 2 giải thích các tham số của hàm `FindPivot`: "Trả về chỉ số của phần tử chứa giá trị tham chiếu". Chúng ta thấy rằng các ô trống này dùng để quyết định giá trị tham chiếu và thiết lập chỉ số của tham số `Ret`. Phần này chứa những ô trống được thực hiện khi điều kiện "`A[K] = Pivot`" là sai. Trong lúc đó, tại thời điểm đầu của chương trình, chúng ta đọc thấy "`Pivot ← A[Min]`" dẫn tới tại thời điểm này "`Pivot = A[Min]`". Vì vậy, điều kiện "`A[K] = Pivot`" là sai nghĩa là giá trị của `A[K]` khác với giá trị của `A[Min]`. Đúng hoặc Sai của điều

kiện "A[K] > Pivot " quyết định giá trị của tham số Ret, phần này chính xác tương ứng với phần "chọn Pivot" đã mô tả ở trên. Ô trống C áp dụng khi A[K] > Pivot (= A[Min]). Nhớ lại Pivot là giá trị lớn hơn giữa A[K] (khác với A[Min]) và A[Min], và vì A[K] là lớn hơn nên A[K] sẽ trở thành Pivot. Ô trống C sau đó (d) Ret ← K. Ngược lại, ô trống D là trường hợp khác vì thế đáp án đúng là (f) Ret ← Min. Các đáp án từ (a) tới (c) đều thiết lập một phần tử của mảng cho tham số Ret. Bằng cách đọc cẩn thận giả thích trong bảng 2 ("trả về chỉ số phần tử chứa giá trị tham chiếu"), khi đó các đáp án sai ngay lập tức có thể được loại bỏ. Bởi vậy bạn còn lại có 3 lựa chọn. Sắp xếp nhanh nghe có vẻ là một khái niệm khó hiểu, nhưng hi vọng các bạn tin rằng đó không phải là một câu hỏi khó.

[Câu hỏi con 2]

Đây là câu hỏi truy vết. Các ô trống trong chương trình đã được điền vào, vì thế cơ bản là không có lựa chọn nhưng để chèn vào nội dung của các ô trống và theo vết chương trình một cách cẩn thận. Đây là câu hỏi con để kiểm tra bạn với trọng tâm của câu hỏi, đó là sắp xếp nhanh, thủ tục đệ quy.

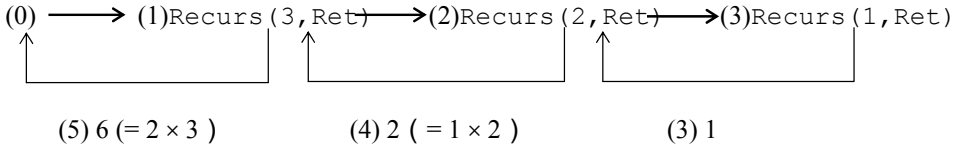
Đầu tiên chúng ta xem lại về đệ quy. Đệ quy có nghĩa là "Gọi chính nó". Một ví dụ thường được sử dụng để giả thích về đệ quy là hàm tính giai thừa. Chương trình được viết như sau. Nhớ lại là giai thừa của một số là kết quả phép nhân các số từ 1 cho tới số đó. Ví dụ giai thừa của 3 (3!) là 1 × 2 × 3 = 6.

[Ví dụ]

```
○ Recurs(n, Ret)
○ integer: n, Ret
  n > 1
  · Recurs(n-1, Ret)
  · Ret ← n × Ret
  ·
  · Ret ← 1
```

(Trong giả ngôn ngữ sử dụng với các câu hỏi thi, các tự hàm dường như không thể trả về giá trị, vì thế có vẻ khác so với chương trình thực tế viết trên ngôn ngữ giống như C).

Bây giờ, xem xét lời gọi "Recurs(3,Ret)" thực hiện thế nào khi sử dụng chương trình trên. Đầu tiên, vì tham số truyền vào là 3 nên giá trị của tham số n là "3", thỏa mãn "3 > 1", tiếp đó "Recurs(2,Ret)" được gọi. Tham số n là 2, và bởi vì "2 > 1", sẽ gọi hàm "Recurs(1,Ret)". Tiếp theo, khi tham số n là 1, và vì "1 > 1" là sai, nên tham số Ret được thiết lập là 1, và chương trình quay về nơi lời gọi thực hiện. Tại lời gọi (Recurs(2,Ret)), giá trị của Ret (= 1) được sử dụng để thực hiện "Ret ← Ret (= 1) × n (= 2)". Sau đó nó lại quay về thời điểm thực hiện lời gọi (Recurs(3,Ret)), với "2" là giá trị của Ret. Điều đó làm cho chương trình hoàn thành quá trình tìm "Recurs(3,Ret)" khi nó thực hiện "Ret ← Ret (= 2) × n (= 3)". Kết quả Ret là 6, là kết quả đúng. Biểu đồ sau thể hiện các bước thực hiện chương trình để thu kết quả.



Như nhìn thấy ở đây, chương trình gọi chính nó, nhưng theo thứ tự để có thể kết thúc quá trình, hàm được gọi cuối cùng phải kết thúc.

Bây giờ, chúng ta xem xét chương trình được cho trong câu hỏi. Như bảng 4 gợi ý, `QuickSort(A[],0,9)` được gọi đầu tiên. Tiếp đó, thực hiện lời gọi thứ hai `QuickSort(A[],0,4)`. Tại thời điểm này thấy rằng lời gọi đầu tiên chia mảng `A` thành `A[0]`, …, `A[4]`, có giá trị nhỏ hơn giá trị `Pivot`, và `A[5]`, …, `A[9]`, có giá trị lớn hơn hoặc bằng giá trị `Pivot`. Vì thế, theo các nhóm này (sắp xếp bởi `Arrange`), trên bề mặt là gọi các hàm "`QuickSort(A[],0,4)`" và "`QuickSort(A[],5,9)`". Tuy nhiên, chú ý các lời gọi không cần thiết theo đúng thứ tự này. Như đã đề cập trước, trước tiên bên trong của việc thực hiện lời gọi hàm "`QuickSort(A[],0,4)`" thì một lời gọi đệ quy khác được thực hiện, đây là lời gọi được thực hiện đầu tiên.

`QuickSort(A[],0,9)` ← Lời gọi đầu tiên

…

`FindPivot` và `Arrange` chia mảng thành hai nhóm: `A[0]`,…, `A[4]` và `A[5]`,…, `A[9]`.

…

`QuickSort(A[],0,4)` ← Lời gọi thứ hai.

`QuickSort(A[],5,9)` ← Có thể không là lời gọi thứ ba.

Lời gọi trong khi `QuickSort(A[],0,4)` được thực hiện trước.

Chúng ta theo vết của lời gọi thứ hai (`QuickSort(A[],0,4)`). Phạm vi các đối tượng để sắp xếp là `A[0]`, …, `A[4]`, và giá trị của các phần tử như sau (Lấy từ bảng 4).

0	1	2	3	4
3	2	1	4	0

Trước tiên, xem xét `Pivot`, là giá trị thu được bởi hàm `FindPivot`. Giá trị khác đầu tiên từ `A[0]` (=`A[Min]`) =3, ví dụ, giá trị lớn hơn giữa `A[1]=2` và `A[0]=3`, vì thế `A[0] = 3` được chọn. Tiếp đó, thiết lập `Pivot = 3`, bây giờ chúng ta theo vết hàm `Arrange(A[],0,4,3,K)`.

Bắt đầu chương trình có "`L ← Min` " và "`R ← Max`" vì thế `L` = 0 và `R` = 4. Sau đó, chương trình vào vòng lặp với điều kiện thỏa mãn `L <= R`.

(i) Đầu tiên có 3 câu lệnh: "`Tmp ← A[L]`", "`A[L] ← A[R]`", và "`A[R] ← Tmp`". Đây là cách chuẩn để chuyển đổi hai giá trị của hai biến. Nói cách khác giá trị của `A[0]` và `A[4]` được đổi cho nhau, dẫn tới `A[0] = 0` và `A[4] = 3`.

(ii) Trong vòng lặp tiếp theo, khi `A[L] < Pivot`, giá trị của `L` được tăng lên 1 mỗi lần. Vì `A[0] = 0, A[1] = 2, A[2] = 1, A[3] = 4`, ta thấy là `A[3] < Pivot` không thỏa mãn nên kết quả là `L = 3`.

(iii) Tiếp theo, trong vòng lặp dưới đây, trong khi "`A[R] >= Pivot`", giá trị của `R` được giảm đi 1 mỗi lần. Vì `A[4] = 3, A[3] = 4, và A[2] = 1,` ta thấy là `A[2] >= Pivot` không thỏa mãn nên kết quả là `R` = 2.

(iv) Vì `L = 3` và `R = 2`, điều kiện lặp không thỏa mãn nên chương trình con được hoàn thành. Biểu đồ dưới đây thể hiện quá trình.

Pivot		L	R	0	1	2	3	4
				3	2	1	4	0
3	(i)	0	4	0				3
	(ii)	0→3		0	2	1	4	
	(iii)		4→2			1	4	3
	(iv)	3	2	0	2	1	4	3

Khi vòng lặp kết thúc, giá trị của `L` (= 3) tại thời điểm đó sẽ được gán cho `Ret`, và chương trình quay về nơi lời gọi đầu tiên thực hiện.

Bây giờ chúng ta xem xét các ô trống. Đầu tiên nhìn vào ô trống. `A[0]` tới `A[4]` là trong điều kiện (iv) của biểu đồ trên. `A[5]` tới `A[9]` không thay đổi, vì thế chúng ta kế thừa điều kiện từ lời gọi thứ hai (trong tất cả các lựa chọn của nhóm trả lời, phần này là xác định). Vì thế đáp án đúng là (c). Tiếp theo xem xét ô trống F và G. Bằng cách truy vết thấy rõ ràng là giá trị của `Ret` (= `K`) là 3 và vì thế `A[0], …, A[2]` là các giá trị nhỏ hơn giá trị `Pivot` trong khi `A[3]và A[4]` là các giá trị lớn hơn hoặc bằng gí trị `Pivot`. Với kết quả này, lời gọi thứ hai (`QuickSort(A[], 0, 4)`) được thực hiện, trong đó các lời gọi sâu hơn `QuickSort(A[], 0, 2)`" và "`QuickSort(A[], 3, 4)`" sẽ được thực hiện. Vì thế lời gọi thứ 3 là "`QuickSort(A[], 0, 2)`", đáp án đúng cho ô trống F là (a) 0, và đáp án cho ô trống G là (c) 2.

Bây giờ, bạn nên học khả năng tìm ra cách để tìm được đáp án. Nếu bạn đã học về thuật toán sắp xếp nhanh trước khi kiểm tra, bạn sẽ cảm thấy dễ dàng và tự tin, và có lẽ bạn sẽ có nhiều thuận lợi. Tuy nhiên nếu bạn chưa học, câu hỏi này không phải là câu hỏi không thể trả lời miễn là bạn biết về chủ đề này với (quá trình lặp đệ quy thủ tục chia nhóm, trong đó dữ liệu được phân chia thành hai nhóm, một nhóm có giá trị lớn hơn và nhóm còn lại có giá trị nhỏ hơn hoặc bằng với một giá trị cho trước) và hiểu lời gọi ban đầu của quá trình đệ quy. Các câu hỏi thuật toán trong bài thi FE thông thường luôn chú trọng tới khả năng để truy vết hơn là khả năng xây dựng thuật toán; và trong vài năm gần đây, trong các bài thi khả năng truy vết chương trình càng trở nên quan trọng. Trong phần giải thích này chúng ta truy vết lời gọi thứ hai nhưng tốt nhất là nên truy vết lời gọi đầu tiên cũng như là lời gọi thứ ba và các lời gọi sau đó. Để bạn tham khảo chúng tôi kết thúc mục này bằng cách đưa ra biểu đồ truy vết lời gọi đầu tiên.

Pivot	L	R	0	1	2	3	4	5	6	7	8	9	Giải thích
			3	5	8	4	0	6	9	1	2	7	
5	0	9	7									3	Đảo vị trí
	0		7										Bước lặp với `L`
		9										3	Bước lặp với `R`
	0	9	3									7	Đảo vị trí
	0→1		3	5									Bước lặp với `L`
		9→8									2	7	Bước lặp với `R`
	1	8		2							5		Đảo vị trí
	1→2			2	8								Bước lặp với `L`
		8→7								1	5		Bước lặp với `R`
	2	7			1					8			Đảo vị trí
	2→5				1	4	0	6					Bước lặp với `L`
		7→4					0	6	9	8			Bước lặp với `R`
	5	4	3	2	1	4	0	6	9	8	5	7	`L > R`

7 Thiết kế chương trình

[Phạm vi các câu hỏi]
Qui trình phát triển hệ thống,
Quá trình thiết kế chương trình, thiết kế có cấu trúc,
Thiết kế module,
Tài liệu thiết kế chương trình,….

Câu hỏi 1

Q1. Đọc mô tả sau về thiết kế chương trình và trả lời các câu hỏi con từ 1 đến 4.

Một hệ thống được thiết kế để cung cấp những tin tức trong nước dựa vào sở thích của người dùng. Dưới đây là phác thảo về hệ thống

[Phác thảo hệ thống]

(1) Tin tức cần cung cấp được lưu trong một tệp tin tức theo định dạng dưới đây. Tệp tin tức này là một tệp tuần tự bao gồm các bản ghi, mỗi bản ghi chứa một mục tin tức, và các mục tin tức được ghi lại theo thứ tự mà chúng được đăng kí.

Định dạng bản ghi của tệp tin tức

Ngày đăng kí	Thời gian đăng kí	Ngày xảy ra	Thời gian xảy ra	Loại	Tiêu đề	Tóm tắt	Chi tiết	Địa chỉ tệp ảnh

Ngày đăng kí là một chuỗi kí tự có 8 chữ số đại diện cho năm, tháng và ngày khi mục tin tức được đăng kí. Ví dụ "20080401" biểu diễn cho ngày mùng 1 tháng 4 năm 2008. Thời gian đăng kí là chuỗi kí tự có 4 chữ số đại diện cho giờ và phút khi mục tin tức được đăng kí. Ví dụ :"1830" thay thế cho 6h30 chiều. Ngày xảy ra là năm, tháng và ngày xảy ra sự kiện của tin tức. Ngày xảy ra có cùng định dạng với ngày đăng kí. Thời gian xảy ra là thời điểm xảy ra sự kiện nêu trong tin tức và có cùng định dạng với thời gian đăng kí. Loại là loại tin tức, và lưu trữ một trong các loại sau: "Chăm sóc sức khỏe", "Giáo dục", "Kinh tế", "Giải trí", "Khoa học", "Xã hội", "Thể thao", và "Chính trị". Bên cạnh đó, cũng cần lưu thông tin về dòng tiêu đề, tóm tắt của mục tin, nội dung chi tiết và địa chỉ tệp ảnh liên quan.

(2) Thông tin người dùng được đăng kí trong tệp người dùng theo định dạng sau. Tệp người dùng là một tệp được đánh chỉ mục sử dụng khóa là định danh người dùng (user ID).

Định dạng bản ghi của tệp người dùng

Định danh người dùng (User ID)	Loại 1	Loại 2	Loại 3	Loại 4	Loại 5	Ngày sử dụng cuối cùng	Thời gian sử dụng cuối cùng

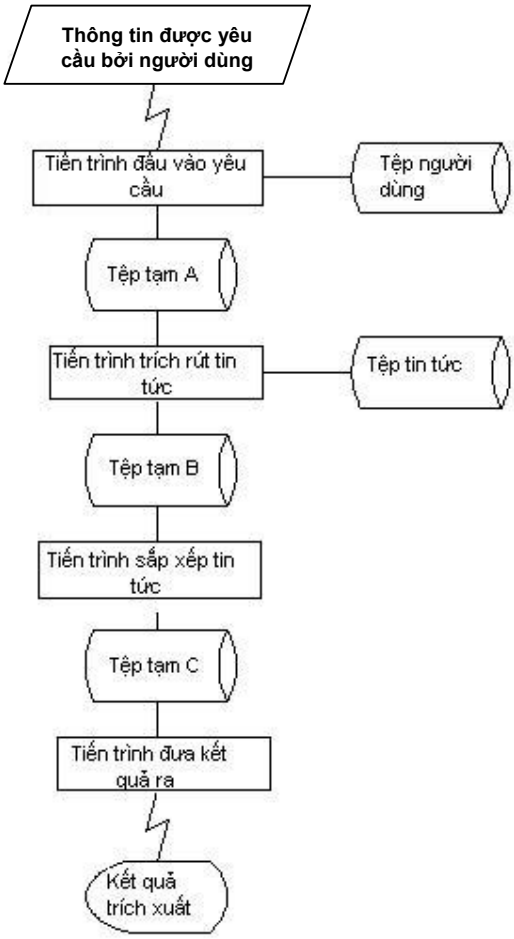
Tệp người dùng đăng kí tối đa 5 loại cho các tin tức ưa thích của người dùng trong các trường từ loại 1 đến loại 5. Nếu ít hơn 5 loại thì các trường loại còn lại sẽ lưu giá trị NIL (rỗng). Hệ thống cung cấp tối đa 10 mục tin tức lùi dần theo ngày giờ cho mỗi loại đã đăng kí.

Ngày sử dụng cuối cùng và thời gian sử dụng cuối cùng là ngày và thời điểm khi người dùng sử dụng hệ thống lần cuối. Ngày sử dụng cuối cùng và thời gian sử dụng cuối cùng là chuỗi kí tự có định dạng lần lượt giống với ngày đăng kí và thời gian đăng kí.

(3) Nếu quá 24 giờ đã trôi qua từ ngày giờ của lần sử dụng cuối cùng, tất cả mọi mục tin tức đã đăng kí trong 24 giờ trước được đưa ra để trích rút. Nếu lần sử dụng hệ thống trước đó của

một người dùng trong vòng 24 giờ, thì chỉ những mục tin tức đã được đăng kí sau thời điểm sử dụng cuối cùng được đưa ra để trích rút.

(4) Dựa vào loại thiết bị cuối của người dùng, hệ thống sẽ thay đổi định dạng của kết quả trích rút cần cung cấp. Nếu thiết bị cuối của người dùng là điện thoại di động, thì kiểu thiết bị cuối thích hợp là "Đơn giản", và hệ thống đưa ra một đoạn tóm tắt của của mỗi mục tin tức. Nếu thiết bị cuối của người dùng là một máy tính cá nhân thì một kiểu thiết bị cuối tương thích là "Chi tiết", và hệ thống sẽ cung cấp những chi tiết của mỗi tin tức và các hình ảnh liên quan.



Hình. Luồng tiến trình

[Mô tả tiến trình]

(1) Trong tiến trình đầu vào yêu cầu, một bản ghi có định danh người dùng giống với định danh người dùng có trong thông tin được yêu cầu sẽ được chọn từ trong tệp người dùng. Thông tin được yêu cầu từ người dùng có định dạng sau.
Loại liên quan, "Đơn giản" hay "Chi tiết" được lưu trong trường kiểu thiết bị cuối.

Định dạng của thông tin được yêu cầu

Định danh người dùng	Kiểu thiết bị cuối

Các bản ghi trong định dạng sau được ghi vào tệp tạm *A* từ thông tin được yêu cầu và bản ghi được lựa chọn.

Định dạng bản ghi của tệp tạm *A*

Loại 1	Loại 2	Loại 3	Loại 4	Loại 5	Ngày bắt đầu trích rút	Thời gian bắt đầu trích rút	Kiểu thiết bị cuối

Cuối cùng, ngày của lần sử dụng cuối và thời gian của lần sử dụng cuối trong bản ghi người dùng liên quan trong tệp người dùng được cập nhật theo ngày giờ hiện tại.

(2) Trong tiến trình trích rút tin tức, dựa vào tệp tạm *A*, bản ghi thỏa mãn 2 điều kiện sau được trích rút đồng thời từ tệp tin tức.
(i) Ngày giờ đăng kí của bản ghi tin tức trong tệp tin tức gần hiện tại hơn so với ngày giờ bắt đầu trích rút từ tệp tạm *A*.
(ii) Loại của các bản ghi tệp tin tức phù hợp với 1 trong các trường từ loại 1 đến loại 5 trong tệp tạm *A*. Thông tin cần thiết được thêm vào bản ghi thỏa mãn những điều kiện này và sau đó bản ghi này được ghi vào tệp tạm *B*.

(3) Trong tiến trình sắp xếp tin tức, tệp tạm B được sắp xếp rồi ghi vào tệp tạm *C*.

(4) Trong tiến trình đưa kết quả ra, cứ 10 mục được trích rút cho mỗi loại từ tệp tạm *C*, và sau đó kết quả được ghi theo định dạng sau để phù hợp với kiểu thiết bị cuối.

Định dạng được sử dụng khi kiểu thiết bị cuối là "Đơn giản":

Ngày xảy ra	Thời gian xảy ra	Loại	TIêu đề	Tóm tắt

Định dạng được sử dụng khi kiểu thiết bị cuối là "Chi tiết":

Ngày xảy ra	Thời gian xảy ra	Loại	TIêu đề	Tóm tắt	Địa chỉ tệp ảnh

(5) Ở đây, thông tin giữa các tiến trình được chuyển cho nhau chỉ khi dùng các tệp được chỉ ra trong hình vẽ.

Câu hỏi con 1

Từ nhóm các câu trả lời dưới đây, hãy chọn câu trả lời đúng để điền vào ô trống [ ] trong mô tả sau.

Một bản ghi của người dùng trong tệp người dùng được chỉ ra dưới đây:

XR205	Kinh tế	Giải trí	NIL	NIL	NIL	20080401	2038

Nếu người dùng này sử dụng hệ thống lúc 17:00 ngày 3 tháng 4 năm 2008, [A] được lưu trong trường Thời gian bắt đầu trích rút trong tệp tạm *A* bởi tiến trình đầu vào yêu cầu và [B] được lưu trong trường Thời gian bắt đầu trích rút.

Nhóm câu trả lời:

a) "0000" b) "1700" c) "2038"
d) "20080401" e) "20080402" f) "20080403"

Câu hỏi con 2

Định dạng bản ghi của tệp tạm *B* được ghi bởi tiến trình trích rút tin tức được chỉ ra dưới đây. Từ nhóm các câu trả lời ở dưới, chọn câu trả lời đúng để điền vào các ô trống [ ] trong định dạng này.

Ngày xảy ra	Thời gian xảy ra	[C]	Tiêu đề	Tóm tắt	Chi tiết	Địa chỉ của tệp ảnh	[D]

Nhóm câu trả lời:

a) Kiểu thiết bị cuối
b) Thời gian bắt đầu trích rút
c) Ngày bắt đầu trích rút
d) Thời gian đăng kí
e) Ngày đăng kí
f) Loại
g) Định danh người dùng

Câu hỏi con 3

Từ nhóm câu trả lời ở dưới, chọn câu trả lời đúng để điền vào các ô trống [ ] trong mô tả sau.

Trong tiến trình sắp xếp tin tức, các bản ghi thu được trong tiến trình trích rút tin tức được đưa ra để sắp xếp. Trong trường hợp này, khóa sắp xếp đầu tiên là [E], và các bản ghi được sắp xếp theo thứ tự tăng các mã kí tự. Khóa sắp xếp thứ hai là [F], và các bản ghi được sắp xếp thep thứ tự giảm dần. Khóa sắp xếp thứ 3 là [G], và các bản ghi được sắp xếp theo thứ tự giảm dần.

Nhóm câu trả lời:

a) Tóm tắt
b) Địa chỉ tệp ảnh
c) Chi tiết
d) Thời gian bắt đầu trích rút
e) Thời gian đăng kí
f) Ngày đăng kí
g) Thời gian xảy ra
h) Ngày xảy ra
i) Loại

Câu hỏi con 4

Từ nhóm câu trả lời ở dưới, chọn câu trả lời đúng để điền vào các ô trống [ ] trong mô tả sau.

Một bản ghi của người dùng trong tệp người dùng được chỉ ra dưới đây:

DT512	Chính trị	Kinh tế	Thể thao	NIL	NIL	20080401	0400

Số mục tin tức được đăng kí cho mỗi loại lúc 11:00 ngày 1 tháng 4 năm 2008 được chỉ ra dưới đây. Nếu người dùng đó sử dụng hệ thống tại cùng thời điểm trong cùng ngày đó, số lượng bản ghi được ghi vào tệp tạm *C* bởi tiến trình sắp xếp tin tức là [H], và số mục tin tức được hiển thị bởi tiến trình đưa kết quả ra là [I].

Bảng Số mục tin tức được đăng kí

Loại	Số mục tin tức được đăng kí mỗi giờ										
	00:00 đến 00:59	01:00 đến 01:59	02:00 đến 02:59	03:00 đến 03:59	04:00 đến 04:59	05:00 đến 05:59	06:00 đến 06:59	07:00 đến 07:59	08:00 đến 08:59	09:00 đến 09:59	10:00 đến 10:59
Chăm sóc sức khỏe	0	0	0	0	0	0	2	0	1	0	0

Giáo dục	0	0	0	0	0	1	0	1	0	0	0
Kinh tế	0	0	0	1	0	2	2	6	8	2	4
Giải trí	1	0	1	0	1	2	0	1	2	1	1
Khoa học	1	0	1	0	1	1	3	2	3	2	5
Xã hội	0	1	2	0	1	0	2	3	5	4	3
Thể thao	0	0	0	1	0	0	1	1	0	3	2
Chính trị	0	1	0	0	0	1	4	2	3	6	3

Nhóm câu trả lời:

a) 27 b) 28 c) 50 d) 53

e) 60 f) 98 g) 108

Đáp án câu 1

Cung cấp những tin tức trong nước dựa vào sở thích của người dùng

Đáp án đúng

[Câu hỏi con 1] A – e, B – b

[Câu hỏi con 2] C – f, D – a (thứ tự không liên quan trong C và D)

[Câu hỏi con 3] E – i, F – h, G – g

[Câu hỏi con 4] H – c, I – a

Giải thích

Chủ đề của câu hỏi này là thiết kế chương trình của một hệ thống cung cấp tin tức trong nước dựa vào sở thích của người dùng. Internet được sử dụng rất phổ biến trong những năm gần đây tới mức mà hầu hết các thí sinh đều hình dung được, nhờ vào việc đọc các mô tả của câu hỏi này, một hệ thống trong đó người dùng có thể đăng kí các loại mà người đó quan tâm sao cho những bài báo tin tức đó được hiển thị, từ những bài gần đây nhất, chỉ trong các loại mà người dùng đã đăng kí. Giữ lại khái niệm này trong đầu khi bạn đọc nội dung câu hỏi.

Nội dung câu hỏi gồm có [Phác thảo hệ thống], Luồng Tiến trình (biểu đồ), [Mô tả Tiến Trình], và các câu hỏi con từ 1 đến 4. Về cơ bản, bạn nên đọc các phần trên theo trình tự từ đầu, nhưng bạn cần sáng tạo một chút trong việc phân phối thời gian đọc các câu hỏi. Trong một bài thi bình thường, "Đọc câu hỏi cẩn thận, thiết lập cài đặt và các điều kiện khác gọn gàng, và sau đó trả lời câu hỏi" là một cách giải quyết tốt; tuy nhiên, trong bài thi FE thông thường, đặc biệt là bài thi chiều, thí sinh thường thiếu thời gian. Bạn cần nhớ điều này và bắt kịp với cách đọc các câu hỏi một cách hiệu quả. Thực ra, bạn đọc mô tả của câu hỏi là để trả lời các câu hỏi con. Thậm chí nếu bạn dùng nhiều thời gian đọc mô tả câu hỏi trước câu hỏi con, khi bạn trả lời câu hỏi con, bạn sẽ đọc lại mô tả câu hỏi một lần nữa để chắc chắn. Với điều đó trong ý nghĩ, bạn sẽ đọc câu hỏi với tâm niệm rằng cuối cùng bạn vẫn đọc lại nó một lần nữa.

Đọc [Phác thảo hệ thống] gần như để biết 4 điều sau:

(1) mô tả của tệp tin tức
(2) mô tả của tệp người dùng
(3) mô tả liên quan đến thời gian người dùng sử dụng dịch vụ lần cuối và phạm vi của mục tin tức được hiển thị
(4) mô tả liên quan đến kiểu thiết bị cuối của người dùng

Tất nhiên, nếu thông tin chi tiết hơn có thể dễ hiểu hơn thì không cần phải cố gắng không nhìn vào nó. Nếu có bất kì con số hoặc điều kiện nào thu hút được sự chú ý của bạn, hãy đi tới và gạch chân chúng. Nhưng tại thời điểm này không cần thiết phải cố gắng để hiểu hoặc ghi nhớ chúng.

Tiếp theo, ta đọc [Mô tả tiến trình]. Đi theo luồng tiến trình trong biểu đồ đã cho, khi bạn đọc dọc theo. Cũng cần biết mô tả này giải thích từ (1) đến (4) cho mỗi tiến trình được chỉ ra trong biểu đồ và các trạng thái trong (5) thực tế là thông tin giữa các tiến trình chỉ được trao đổi khi sử dụng các tệp được chỉ ra trong biểu đồ.

[Câu hỏi con 1]

Hình đã cho chỉ ra nội dung của tệp người dùng, và câu hỏi con này hỏi về các giá trị của các nội dung nào đó (ngày bắt đầu trích rút và thời gian bắt đầu trích rút) của tệp tạm *A* đã được chuẩn bị bởi tiến trình đầu vào yêu cầu khi người dùng sử dụng hệ thống.

Đầu tiên, đọc bản ghi của người dùng. Sau đó đọc [Phác thảo hệ thống] (2), và biết rằng định danh người dùng là "XR205", người dùng đã đăng kí hai loại (“kinh tế” và “giải trí”) và lần cuối người dùng truy cập hệ thống này là 20:38 ngày 1 tháng 4 năm 2008. Tại thời điểm này, cần chú ý phần mô tả “hệ thống cung cấp tối đa 10 mục tin tức lùi dần theo ngày giờ cho mỗi loại được đăng kí”.

Điều được hỏi bao gồm các nội dung trong một tệp tạm được tạo bởi tiến trình đầu vào yêu cầu, nên tiếp theo bạn cần đọc (1) của [Mô tả tiến trình]. Ở đó, bạn sẽ thấy tệp tạm *A* bao gồm thông tin được yêu cầu (định danh người dùng, kiểu thiết bị cuối) và bản ghi đã lựa chọn (nội dung của tệp người dùng tương ứng với định danh người dùng), nhưng mô tả này không chỉ rõ ngày hoặc thời gian bắt đầu trích rút sẽ được sử dụng như thế nào. Vì thế bạn nên tiếp tục đọc tiếp. Khi bạn đọc đến (2), bạn sẽ thấy ngày giờ bắt đầu trích rút được sử dụng như những điều kiện trích rút bản ghi cho tệp tin tức. Nói cách khác, chương trình so sánh ngày giờ đăng kí của bản ghi trong tệp tin tức với ngày giờ bắt đầu trích rút và trích rút những bản ghi đã đăng kí sau thời điểm bắt đầu trích rút ((2)(i)).

Ở đây, tin tức được trích rút từ tệp tin tức để chỉ hiển thị những mục tin tức mà thỏa mãn những điều kiện nào đó. Từ ý nghĩa của mục tin tức, ta thấy điều này liên quan đến điều kiện về thời gian. Nhớ rằng có một mô tả có phần liên quan đến điều này trong (3) của [Phác thảo hệ thống], vì thế cần đọc lại nó một lần nữa. Theo đó, các trạng thái là khác nhau phụ thuộc vào việc liệu thời điểm khi người dùng truy cập vào hệ thống có qua 24 giờ sau thời điểm khi người dùng truy cập lần cuối vào hệ thống. Nếu bạn kiểm tra điều này cho người dùng thì kết quả là nhiều hơn 24 giờ, vì thế mọi mục tin tức đã đăng kí trong vòng 24 giờ cuối được đưa ra để trích rút. Do đó, ngày giờ bắt đầu trích rút trong tệp tạm A là những giá trị ám chỉ thời điểm 24 giờ trước thời điểm hiện tại.

Thời điểm hiện tại là khi người dùng sử dụng hệ thống này, đó là 17:00 ngày 3 tháng 4 năm 2008. Thời điểm 24 giờ trước đó là 17:00 ngày 2 tháng 4 năm 2008. Do đó ô trống A là (e) "20080402", và ô trống B là (b) "1700".

[Câu hỏi con 2]

Câu hỏi này về định dạng bản ghi trong tệp tạm *B*, đó là đầu ra của tiến trình trích rút tin tức. Vì thế bạn cần đọc lại (2) trong [Mô tả tiến trình]. Ở đó, bạn sẽ thấy tệp này dựa vào tệp tạm *A* và chứa các bản ghi được trích rút từ tệp tin tức cũng như vài thông tin cần thiết thêm vào, nhưng bạn sẽ không biết định dạng chi tiết của các bản ghi (cấu tạo mục).

Nói chung một tệp đầu vào là một tệp tổng kết mọi dữ liệu cần thiết cho tiến trình đó. Tệp tạm *B* được tạo ra để đưa tin tức ra trên thiết bị cuối của người dùng trong tiến trình đưa kết quả ra. Vì thế, cần cân nhắc xem liệu tất cả các mục cần thiết để tạo đầu ra có sẵn dùng hay không. Đây không phải là điều duy nhất cho câu hỏi này. Trong bất kì câu hỏi nào, nếu vị trí bạn điền vào ô trống trong một định dạng tệp thì cần chú ý tới nội dung đưa ra (tệp hay báo cáo in) sử dụng tệp đó và tìm kiếm các mục còn thiếu.

Có hai điều bạn phải cẩn thận trong điểm này. Một là trường hợp khi có một bước riêng khác giữa bước trong đó nội dung đầu ra liên quan đã sẵn sàng và bước đang xét. Trong trường hợp này, tiến trình của bước riêng giữa hai bước cũng phải được xem xét. Hai là liệu giá trị của mục tin đó có thể tìm thấy hay không trong bước tại đó tệp đưa ra để xem xét đang được tạo ra.

Giữa tiến trình trích rút tin tức nơi tệp tạm B được tạo ra và tiến trình đưa kết quả ra nơi nội dung đầu ra đã sẵn sàng, có một tiến trình sắp xếp tin tức. Nhưng vì bước này chỉ liên quan đến việc sắp xếp nên các tệp và giá trị tệp không thay đổi. Do đó, bạn không cần lo nghĩ về tiến trình sắp xếp tin tức, chỉ cần so sánh nội dung đầu ra của (4) với định dạng bản ghi của tệp tạm B. Hiển nhiên là định dạng bản ghi đang bị thiếu trường "loại". Vì thế sau một chút băn khoăn, bạn cần nhớ lại mô tả của (5) trong [Mô tả tiến trình]. Vì thông tin trao đổi giữa các tiến trình là giới hạn với các tệp, bạn bắt đầu suy nghĩ, "Không có bất kì thông tin nào khác cần thiết để tạo nội dung đầu ra?" Nội dung đầu ra có nhiều định dạng phụ thuộc vào kiểu thiết bị cuối, vì thế bạn cần kiểu của thiết bị cuối. Lúc này, chúng ta có hai lựa chọn cho câu trả lời: "loại" và "kiểu thiết bị cuối". Nếu bạn có thể tìm ra xem liệu những nội dung này có thể sẵn sàng trong tiến trình trích rút tin tức hay không, thì bạn có thể tìm ra câu trả lời đúng. Cả hai trường này được chứa trong tệp tạm A, tệp đầu vào, nên cả hai đều đúng . Vì thế, câu trả lời chính xác là (f) "loại" cho ô trống C và (a) "kiểu thiết bị cuối" cho ô trống D. Tất nhiên, thứ tự mà chúng được viết là không liên quan, nhưng thứ tự được liệt kê ở đây là thích hợp trên phương diện của tệp tạm A và thành phần của nội dung đầu ra. Không cần thiết phải chuyển đổi thứ tự.

[Câu hỏi con 3]

Câu hỏi này bao gồm những khóa sắp xếp cho tiến trình sắp xếp tin tức. Các câu hỏi về thiết kế của chương trình xử lý tệp hầu như luôn hỏi về khóa sắp xếp, vì thế bạn cần biết cách trả lời các câu hỏi này.

Không cần phải nói, khi đang xem xét các khóa sắp xếp, bạn cần hiểu được mục đích của việc sắp xếp. Thông thường, có bốn mục đích cho việc sắp xếp, vì thế cần biết những điều này và

sử dụng chúng để tham khảo khi bạn xem xét các mục đích có thể để sắp xếp.

Bốn mục đích của sắp xếp

(i) **Đối sánh tệp**: để đối sánh giữa các tệp được tổ chức tuần tự, mỗi tệp phải được sắp xếp theo thứ tự giống với khóa đối sánh. Quá trình sắp xếp được thực hiện để chuẩn bị cho tiến trình này.

(ii) **Tính tổng**: khi tính tổng cho mỗi nhóm cụ thể, chẳng hạn như cho một số người dùng hoặc cho một ngày, dữ liệu trong cùng một nhóm cần đi cùng nhau. Sắp xếp được thực hiện bởi (các) mục khóa cho phân nhóm cụ thể .

(iii) **Xếp loại**: để xếp loại nhiều dữ liệu, dữ liệu phải được sắp xếp từ vị trí đầu tiên trở xuống theo mục được sử dụng để xếp loại. Sắp xếp được thực hiện nhằm mục đích này .

(iv) **Đưa ra các báo cáo**: Trong một báo cáo đưa ra, các nội dung cần thiết về cơ bản được in ra theo thứ tự của tệp đầu vào. Sắp xếp được thực hiện để điều khiển thứ tự của việc đưa dữ liệu ra trên báo cáo.

Trong tiến trình sắp xếp tin tức, tệp tạm B đã sẵn sàng trong tiến trình trích rút tin tức được sắp xếp đầu tiên, sau đó kết quả (tệp tạm C) được chuyển giao cho tiến trình đưa kết quả ra. Do đó, lý do cho việc sắp xếp là tiến trình đưa kết quả ra giả định rằng dữ liệu đã được sắp xếp. Nội dung của tiến trình đưa kết quả ra là trích rút tối đa 10 mục cho mỗi loại từ tệp tạm C, chỉnh sửa và ghi kết quả tùy theo kiểu thiết bị cuối. Vì giới hạn của quá trình trích rút tối đa là 10 mục, bạn có thể nghĩ rằng mục đích không nằm trong bốn điều đã liệt kê ở trên. Tuy nhiên, những gì được trích rút không đơn thuần là lấy ra 10 mục một cách ngẫu nhiên mà trích rút theo quyền ưu tiên. Theo điểm (2) của [Phác thảo hệ thống], quyền ưu tiên là "có tối đa 10 mục tin tức lùi theo ngày giờ cho mỗi loại đã đăng kí", vì thế nó tương ứng với việc xếp loại. Với việc xếp loại này, các mục phải được sắp xếp cho mỗi loại từ những mục mới nhất theo ngày giờ xảy ra. Xếp loại từ những mục tin mới nhất theo ngày giờ xảy ra là sắp xếp từ lớn nhất (nghĩa là theo thứ tự giảm dần. Vì thế, câu trả lời cho ô trống E là loại (i). Đối với hai nhóm trả lời cuối, đáp án là (h) ngày xảy ra cho ô trống F và (g) thời gian xảy ra cho ô trống G..

[Câu hỏi con 4]

Điều được hỏi ở đây là số lượng bản ghi trong tệp tạm C và số mục tin tức hiển thị dưới các điều kiện đã cho. Vượt ra với câu hỏi, bạn đã gần như hiểu hầu hết nội dung tiến trình, vì thế bạn có thể trả lời mà không cần đọc câu hỏi lại. Nghĩa là, bạn có thể gần như dễ dàng theo kịp câu hỏi đến một mức nào đó mà chúng ta đã đọc lại phần giải thích kĩ càng hơn.

Từ bản ghi ta thấy người dùng đã đăng kí ba loại "chính trị", "kinh tế", "thể thao" và người dùng đã sử dụng hệ thống lần cuối vào lúc 4:00 ngày 1 tháng 4 năm 2008. Thời gian hiện tại của việc sử dụng là 11:00 ngày 1 tháng 4 năm 2008, vì thế nó nằm trong 24 giờ và phạm vi thời gian bắt đầu từ các bản ghi được trích rút từ 4:00 ngày 1 tháng 4 năm 2008.

Ở tệp tạm C, trong số tất cả các tin tức được trích rút trong suốt thời gian này, chỉ những tin tức trong ba loại trên được đưa ra. Số bản ghi này là, như được chỉ ra dưới đây, 50 (24 + 7 + 19). Do đó, ô trống H là (c).

Thời hạn cho việc trích rút →

Các loại đích cần trích rút	Loại	Số lượng mục đã đăng kí trong mỗi giờ											
		00:00 ~ 00:59	01:00 ~ 01:59	02:00 ~ 02:59	03:00 ~ 03:59	04:00 ~ 04:59	05:00 ~ 05:59	06:00 ~ 06:59	07:00 ~ 07:59	08:00 ~ 08:59	09:00 ~ 09:59	10:00 ~ 10:59	
	Sức khỏe	0	0	0	0	0	0	2	0	1	0	0	Tổng
	Giáo dục	0	0	0	0	0	1	0	1	0	0	0	
➡	Kinh tế	0	0	0	1	0	2	2	6	8	2	4	24
	Giải trí	1	0	1	0	1	2	0	1	2	1	1	1
	Khoa học	1	0	1	0	1	1	3	2	3	2	5	5
	Xã hội	0	1	2	0	1	0	2	3	5	4	3	3
➡	Thể thao	0	0	0	1	0	0	1	1	0	3	2	7
➡	Chính trị	0	1	0	0	0	1	4	2	3	6	3	19

Nói cách khác, với mỗi loại số lượng mục tin tức được hiển thị là tối đa là 10; vì thế số mục được hiển thị cho kinh tế và chính trị, đều vượt quá 10, sẽ là 10, và với thể thao, nhỏ hơn 10 mục, sẽ là 7. Vì ô trống I là tổng số mục được hiển thị, nên nó sẽ là (a) 27.

Câu hỏi 2

Q2. Đọc mô tả về thiết kết chương trình, sau đó trả lời các câu hỏi con từ 1 đến 3.

Một chương trình được phát triển cho việc truy vấn kiểm kê như là một phần của hệ thống quản lí hàng tồn kho. Chương trình truy vấn kiểm kê này nắm bắt 2 loại giao dịch.

Loại giao dịch "A" là các truy vấn về sự chuyển dịch (vào hoặc ra) của các thành phần trong một khoảng thời gian cụ thể. Một bảng các chuyển dịch thành phần là đầu ra của loại giao dịch này.

Loại giao dịch "S" là các truy vấn về hàng tồn kho hiện tại. Một bảng các trạng thái hàng tồn kho là đầu ra của loại giao dịch này. Nếu như có những phần trong hàng tồn nhỏ hơn giá trị nhỏ nhất của số lượng hàng thì các phần này được thể hiện với màu đỏ.

Mỗi giao dịch có cấu trúc như hình bên dưới. Giá trị của các con số, loại dữ liệu,… trong mỗi trường được kiểm tra. Bên cạnh đó, hệ thống đòi hỏi một giá trị số nhằm xác nhận quyền sử dụng các bảng có quan hệ hàng tồn. Sự tồn tại của giá trị số xác nhận quyền sử dụng trong giao dịch cũng được kiểm tra trong bảng giá trị xác nhận quyền sử dụng. Các giao dịch không vượt qua được sự kiểm tra này sẽ được xử lí như các lỗi.

[Cấu trúc giao dịch]

(1) Giao dịch loại "A"

A	Loại giao dịch	Thời điểm bắt đầu (năm, tháng, ngày)	Thời điểm kết thúc (năm, tháng, ngày)

Số lượng các số thứ tự thành phần	Số thứ tự thành phần	Số thứ tự thành phần	. . .	Số thứ tự thành phần

(2) Giao dịch loại "S"

A	Loại giao dịch

Số lượng các số thứ tự thành phần	Số thứ tự thành phần	Số thứ tự thành phần	. . .	Số thứ tự thành phần

[Thành phần trong cấu trúc dữ liệu có liên quan tới các truy vấn kiểm kê]

(1) Bảng thành phần

Số thứ tự thành phần	Tên thành phần	Đơn vị sắp xếp	Số lượng hàng thấp nhất

(2) Bảng kiểm kê

Số thứ tự thành phần	Số lượng hàng

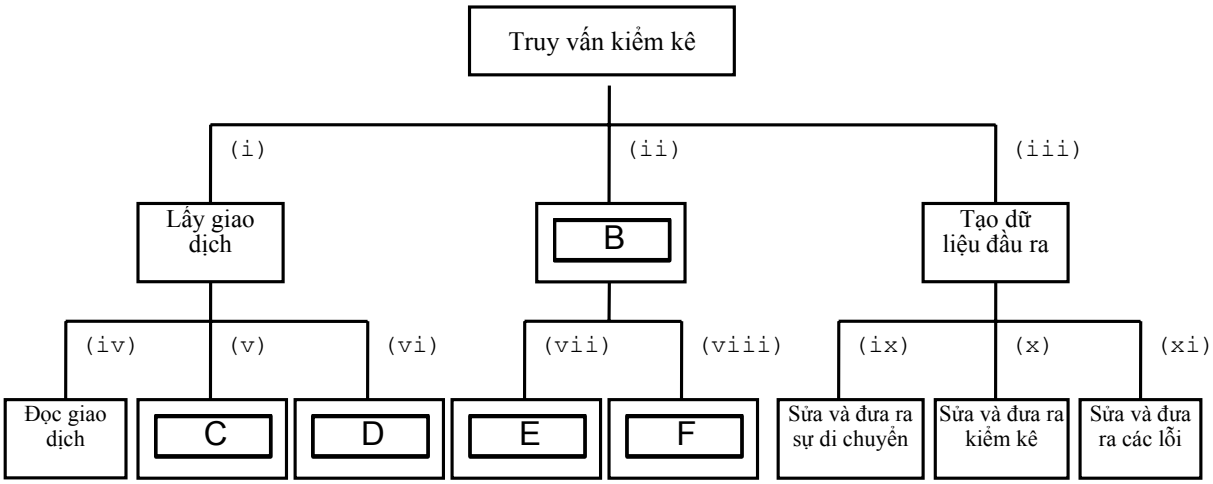
(3) Bảng giá trị xác nhận quyền sử dụng

Giá trị xác nhận quyền sử dụng

(4) Bảng di chuyển

Số thứ tự thành phần	Vào/ra	Số lượng	Tem thời gian (năm, tháng, ngày, giờ, phút)

[Lược đồ cấu trúc mô đun]



[Các giao diện liên mô đun]

Số thứ tự giao diện	Đầu vào	Đầu ra
(i)	–	Giao dịch, cờ kiểm ra cấu trúc, cờ kiểm tra quyền sử dụng
(ii)	Giao dịch	Các bản ghi kết nối, cờ kiểm soát thấp, cờ mất số thứ tự thành phần
(iii)	Các bản ghi kết nối, G, cờ kiểm soát thấp, cờ mất số thứ tự thành phần, cờ kiểm tra cấu trúc, cờ kiểm tra quyền sử dụng	–
(iv)	–	Giao dịch
(v)	Giá trị quyền sử dụng	Cờ kiểm tra quyền sử dụng
(vi)	Giao dịch	Cờ kiểm ra cấu trúc
(vii)	Giao dịch	Các bản ghi kết nối (bảng các thành phần, bảng di chuyển), cờ mất số thứ tự thành phần
(viii)	Giao dịch	Các bản ghi kết nối (bảng các thành phần, bảng kiểm soát), cờ kiểm soát thấp, cờ mất số thứ tự thành phần
(ix)	Các bản ghi kết nối (bảng các thành phần, bảng di chuyển)	–
(x)	Các bản ghi kết nối (bảng các thành phần, bảng kiểm soát), cờ kiểm soát thấp	–
(xi)	Giao dịch, H	–

[Cấu trúc cờ]

(1) Cờ kiểm ra cấu trúc, cờ kiểm tra quyền sử dụng

Cờ

(Không lỗi: 0, Lỗi: 1)

(2) Cờ kiểm soát thấp

Số các số thứ tự thành phần	Cờ	Cờ	. . .	Cờ

(Trong thứ tự các số thứ tự thành phần trong giao dịch, lớn hơn hoặc bằng giá trị hàng nhỏ nhất tồn tại: 0, nếu kiểm soát thấp hơn: 1)

(3) Cờ mất số thứ tự thành phần

Số các số thứ tự thành phần	Cờ	Cờ	. . .	Cờ

(Theo thứ tự của các số thứ tự thành phần trong giao dịch, số thứ tự thành phần tồn tại: 0, số thứ tự thành phần bị mất: 1)

Câu hỏi con 1

Từ nhóm các câu trả lời bên dưới, chọn ra câu trả lời đúng để điền vào các ô trống ☐ trong cấu trúc giao dịch bên trên.

Nhóm câu trả lời cho A:

a) "A"
b) "AS"
c) "S"
d) Số các giao dịch
e) Di chuyển
f) Tem thời gian (năm, tháng, ngày, giờ, phút)
g) Cờ
h) Giá trị số xác nhận quyền sử dụng

Câu hỏi con 2

Từ nhóm các câu trả lời bên dưới, chọn ra câu trả lời đúng để điền vào các ô trống ☐ trong lược đồ cấu trúc đun bên trên.

Nhóm câu trả lời từ B tới F:

a) Lấy các bản ghi kết nối
b) Xử lí các bản ghi có liên quan tới các trạng thái hàng tồn.
c) Cập nhật bảng hàng tồn
d) Kiểm tra cấu trúc giao dịch
e) Xử lí các bản ghi có liên quan tới các hoạt động di chuyển
f) Cập nhật mục nhập và gửi bảng đi
g) Tạo ra thông tin xếp đặt vị trí
h) Kiểm tra giá trị xác nhận quyền truy cập

Câu hỏi con 3

Từ nhóm các câu trả lời bên dưới, chọn ra câu trả lời đúng để điền vào các ô trống ☐ trong bảng các giao diện lược đồ tương tác ở trên.

Nhóm câu trả lời cho G:

a) Giao dịch
b) Loại giao dịch
c) Số các giao dịch

d) Tem thời gian (năm, tháng, ngày, giờ, phút)
e) Số thứ tự thành phần
f) Số các số thứ tự thành phần
g) Tên thành phần
h) Giá trị xác nhận quyền sử dụng
i) Bảng giá trị xác nhận quyền sử dụng

Nhóm câu trả lời cho H :

a) Cờ kiểm soát thấp, cờ mất số thứ tự thành phần
b) Cờ kiểm soát thấp, cờ mất số thứ tự thành phần, cờ kiểm tra cấu trúc
c) Cờ kiểm soát thấp, cờ mất số thứ tự thành phần, cờ kiểm tra xác nhận quyền sử dụng
d) Cờ kiểm soát thấp, cờ kiểm tra cấu trúc
e) Cờ kiểm soát thấp, cờ kiểm tra cấu trúc, cờ kiểm tra xác nhận quyền sử dụng
f) Cờ kiểm soát thấp, cờ kiểm tra xác nhận quyền sử dụng
g) Cờ mất số thứ tự thành phần, cờ kiểm tra cấu trúc
h) Cờ mất số thứ tự thành phần, cờ kiểm tra cấu trúc, cờ kiểm tra xác nhận quyền sử dụng
i) Cờ mất số thứ tự thành phần, cờ kiểm tra xác nhận quyền sử dụng
j) Cờ kiểm tra cấu trúc, cờ kiểm tra xác nhận quyền sử dụng

Đáp án câu 2

Đáp án đúng:

[Câu hỏi con 1] A – h

[Câu hỏi con 2] B – a, C – h, D – d, E – e, F – b

[Câu hỏi con 3] G – a, H – h

Giải thích

Đây là một câu hỏi đề cập đến việc thiết kế chương trình tạo truy vấn trên dữ liệu kiểm kê. Các câu hỏi thành phần có liên quan đến dữ liệu giao dịch, biểu đồ cấu trúc giao dịch, và giao diện giữa các mô đun, tất cả trước đây thường xuất hiện dưới dạng các câu hỏi thiết kế nội dung chương trình, do đó hãy chắc chắn rằng bạn hiểu cách trả lời những câu hỏi này.

[Câu hỏi con 1]

Ô trống được đặt vào cấu trúc giao dịch và bạn được yêu cầu phải điền nội dung (tên khoản mục) vào các ô trống. Gợi ý là ô trống này được thiết lập cho các kiểu "A" và "S". Mặc dù chữ "giao dịch" có thể không quen thuộc với nhiều thí sinh, một cách cơ bản, nó có nghĩa là "dữ liệu vào". Chẳng hạn, trong một chương trình trực tuyến như chương trình này, một "giao dịch" là nội dung được đưa vào trong màn hình nhập liệu. Ngoài ra, nhắc lại rằng phân chia TR, một kỹ thuật phân chia mô đun thường xuất hiện trong các kỳ kiểm tra, được sử dụng để phân chia các mô đun tiến trình ra từng kiểu khi có nhiều kiểu giao dịch.

Ta hãy quay lại với câu hỏi của chúng ta và đặt sự chú ý vào các phần có liên quan của văn bản câu hỏi dựa trên sự gợi ý đã đề cập ở trên. Vì mục này là chung với nhiều kiểu, bạn nên biết rằng hoặc là phải nhận ra kiểu giao dịch hoặc là phải chỉ ra một tiến trình chung cho các kiểu giao dịch. Để nhận ra được kiểu giao dịch, chúng ta đã xem xét một nhóm gọi là "kiểu giao dịch", do vậy đó có thể là đáp án đúng. Do đó, ta hãy chú tâm vào một tiến trình chung. Ngay trên [cấu trúc giao dịch], đoạn văn bản chỉ ra rằng mỗi giao dịch "phải có một giá trị xác nhận quyền sử dụng để sử dụng được bảng quan hệ - hàng tồn kho. Sự tồn tại của giá trị xác nhận quyền sử dụng của giao dịch trong bảng các giá trị quyền sử dụng cũng được kiểm tra". Điều này được thực hiện bất chấp kiểu giao dịch gì, và nó bị mất trong [Cấu trúc giao dịch]. Do đó, đáp án đúng là (h) giá trị xác nhận quyền sử dụng.

[Câu hỏi con 2]

Mục đích của câu hỏi này là điền vào ô trống trong biểu đồ cấu trúc mô đun. Nhóm câu trả lời chỉ ra tên mô đun, nhưng không có một sự giải thích chi tiết nào đề cập đến tiến trình,… do đó có thể có một số khó khăn. Trong một chừng mực nào đó, các tên mô đun có thể đem lại cho bạn một ý tưởng nào đó, nhưng trong các ô trống song song như C và D cũng như E và F, bạn có thể lo lắng về việc thứ tự sẽ được nhận biết như thế nào. **Tuy vậy, tất cả các câu hỏi đều được thiết kế theo cách chúng có thể trả lời được, phải có một gợi ý ở đâu đó.** Đừng lo lắng, hãy tìm câu trả lời.

[Biểu đồ cấu trúc mô đun] có các số từ `(i)` đến `(xi)`. Chú ý rằng những số này cũng được tìm thấy trong [Giao diện liên mô đun] như những con số giao diện. Do đó, chẳng hạn, mô đun tương ứng với ô trống C sẽ tương ứng với con số giao diện (v), chỉ ra rằng nó nhận một "giá trị xác nhận quyền sử dụng" từ mô đun lớp trên của nó là "lấy về giao dịch" và trả về một "cờ kiểm tra quyền sử dụng". Do đó, bạn biết được ô trống C phải là một mô đun thực thi một tiến trình liên quan đến việc kiểm tra các giá trị xác nhận quyền sử dụng. Tìm một câu trả lời đúng tương ứng trong nhóm các câu trả lời, bạn sẽ thấy câu trả lời đúng là (h) "Kiểm tra giá trị xác nhận quyền truy cập".

Bây giờ, bạn sẽ thấy cách để tìm ra những câu trả lời đúng, ta hãy tiến tới điền vào những ô trống còn lại. Mô đun ở ô trống D tương ứng với con số giao diện (vi), vì thế đầu vào là "giao dịch" và đầu ra là "cờ kiểm tra cấu trúc". Do đó, đáp án đúng cho ô trống D là "kiểm tra cấu trúc giao dịch". Các ô trống khác, B, E, và F có một quan hệ mật thiết. Thường thì trong những hoàn cảnh như thế này, mô đun - lớp trên B sẽ có tên gọi mở các mô đun ở dưới nó, chúng sẽ tương ứng với các ô trống E và F. Do đó chúng ta sẽ xem xét đến các ô trống E và F trước. Đối với ô trống E, con số giao diện là (vii), vì thế đầu vào là "giao dịch" và đầu ra là "các bản ghi kết nối (bảng các thành phần, bảng dịch chuyển), cờ mất số thứ tự thành phần". Từ thực tế là các bản ghi kết nối chứa bảng các thành phần cũng như bảng các dịch chuyển, bạn có thể dễ dàng hình dung ra được đó phải là một tiến trình tương ứng với các câu truy vấn liên quan đến các hành vi dịch chuyển của giao dịch loại "A". Tìm môt lựa chọn tương ứng với những điều này trong nhóm các câu trả lời, bạn sẽ tìm thấy (e) "Xử lí các bản ghi có liên quan tới các hoạt động di chuyển" và (f) "Cập nhật mục nhập và gửi bảng đi". Tuy nhiên, nhắc lại rằng mục đích của chương trình là về các truy vấn trên dữ liệu hàng tồn kho. Vì nó là về các truy vấn, câu trả lời dường như không cần phải sửa đổi. Do đó, câu trả lời đúng là (e) "Xử lí các bản ghi có liên quan tới các hoạt động di chuyển" đối với ô trống E. Vì đầu ra cho ô trống F có chứa "các bản ghi kết nối (bảng các thành phần, bảng hàng tồn kho)" , bạn có thể thấy câu trả lời đúng phải là (b) "Xử lí các bản ghi có liên quan tới các trạng thái hàng tồn".

Bây giờ, ô trống còn lại là B. Đây là một mô đun thống nhất một tiến trình của các bản ghi liên quan đến các hành vi di chuyển và một tiến trình của các bản ghi liên quan đến trạng thái hàng tồn. Có một gợi ý khác. Ta đã sử dụng các câu trả lời (b), (d), (e) và (h) để điền vào các ô trống C, D, E và F. Văn bản câu hỏi không nói rằng các lựa chọn câu trả lời có thể được sử dụng nhiều lần, do đó ta có thể loại trừ các câu trả lời đã được lựa chọn. Ngoài ra, (c) cập nhật bảng hàng tồn còn (f) cập nhật bảng các dịch chuyển là các lựa chọn không thích hợp đối với một chương trình truy vấn kiểm kê, do đó chúng cũng bị loại trừ ra. Chúng ta bây giờ còn lại hai lựa chọn: (a): "lấy các bản ghi kết nối" và (g): "Tạo thông tin xếp đặt vị trí". Đối với câu hỏi truy vấn về trạng thái hàng tồn kho, văn bản câu hỏi chi ra rằng: "nếu có một thành phần nào đó trong hàng tồn kho có giá trị dưới số lượng dữ trữ nhỏ nhất thì những thành phần này được biểu thị màu đỏ". Bạn có thể dễ dàng hình dung ra rằng điều đó có nghĩa là "hãy chọn những thành phần này". Tuy nhiên, trong truy vấn của các hành vi dịch chuyển điều đó không thể xảy ra. Như đã đề cập trước đây, mô đun trong ô trống này hợp nhất hai mô đun tương ứng với hai ô trống E và F, vì thế (g) "Tạo thông tin xếp đặt vị trí" không phải là đáp án đúng. Trái lại, vì đầu ra của các giao diện của hai mô đun tương ứng với các ô trống E và F, nghĩa là trả về từ những mô đun này, đều chứa "các bản ghi kết nối", sẽ không có vấn đề gì nếu ta chọn đáp án (a) "lấy các bản ghi kết nối". Do đó, đây là câu trả lời đúng. Ngoài ra, mô đun bên trái của ô trống này là: "lấy giao dịch", do đó nội dung của các mô đun tương đương với nhau và đó là đáp án thích hợp.

[Câu hỏi con 3]

Cuối cùng chúng ta có những câu hỏi yêu cầu điền vào các ô trống trong [giao diện liên mô đun]. Nếu tiến trình của một mô đun thậm chí được mô tả một cách ngắn gọn, bạn có thể nghĩ tới nội dung cần thiết của đầu vào cũng như nội dung cần thiết của đầu ra khi xử lí kết quả, nhưng câu hỏi này không giống như thế. Tuy nhiên, hãy đừng nản chí. **Chắc chắn có những gợi ý ở đâu đó.** Tĩnh tâm và suy nghĩ cũng là một điều quan trọng. Cố gắng nhớ lại những loại mục nào cần thiết trong một giao diện liên mô đun. Những mục cần thiết bao gồm những thứ chỉ được sử dụng trong một mô đun. Không phải những mục này hay là nội dung - có thể nhận được bằng việc tham chiếu tới tệp tin hoặc tính toán trong mô đun – cần nhận được từ mô đun lớp trên. Các mục cần phải xử lí qua các mô đun mức thấp được phân phát như các đầu vào cho các mô đun mức cao hơn, và các mục yêu cầu bởi các mô đun cấp cao hơn được trả về như các đầu ra từ các mô đun cấp thấp hơn. Như bạn thấy ở biểu đồ cấu trúc mô đun, các mô đun có cấu trúc phân cấp. Chẳng hạn, nội dung có tên "giao dịch" là cần thiết cho việc chỉnh sửa và đưa ra lỗi trong (xi). Và những gì nhận nội dung này được gọi là "đọc giao dịch" (iv). Tuy nhiên, giữa "đọc giao dịch" (iv) và "chỉnh sửa và đưa ra lỗi" (xi), không có một giao diện trực tiếp nào, vì thế thông tin không thể được phân phát trực tiếp. Điều này là hiển nhiên. Tuy nhiên, thường những loại điểm này được xem như một manh mối trong câu hỏi liên quan đến giao diện liên mô đun. Trong ví dụ này, ta hãy xem xét cách mà những nội dung giao dịch được phân phát từ (iv) tới (xi). Bạn sẽ thấy rằng đường dẫn là `(iv)` đọc giao dịch → `(ii)` nhận giao dịch → truy vấn trên dữ liệu hàng tồn kho → `(iii)` truy vấn dữ liệu đầu ra → `(xi)` chỉnh sửa và đưa ra thông báo lỗi. Nói cách khác, để tạo dữ liệu đầu ra (iii) – là mô đun cấp trên cho (ix), (x) và (xi), tất cả các mục được phân phát từ (ix) cho đến (xi) phải được thêm vào, trừ trường hợp có thể những mục đó nhận được từ chính trong mô đun và những mục đó nhận được từ mô đun cấp thấp hơn. Từ quan điểm này, ta hãy nhìn vào nội dung theo đó phải được phân phát từ (ix) đến (xi) (bao gồm tất cả các nội dung từ (ix) đến (xi)). Đó là "các bản ghi kết nối, cờ kiểm soát thấp, giao dịch, và ô trống H". Trái lại, hãy so sánh những nội dung đó với đầu vào của (iii), bạn sẽ thấy rằng (a) "giao dịch" bị thiếu. Do đó, đó là đáp án đúng cho ô trống G. Trong hệ thống thực, thường các giao dịch nhận lại được ở (iii) và (xi), dẫn đến khả năng bảo trì thấp. Tuy nhiên, trong một câu hỏi thi, bạn không cần phải lo lắng đến vấn đề này. Hãy nghĩ đơn giản và dễ hiểu.

Bây giờ, đối với ô trống H, ta hãy nghĩ ngược lại. Tất cả các mục mà (iii) nhận được từ mô đun cấp trên của nó, chẳng hạn truy vấn trên dữ liệu tồn kho, hoặc là sẽ được sử dụng bởi chính (iii) hoặc sẽ được phân phát cho mô đun cấp thấp hơn (hoặc cả hai, tất nhiên). Vì quá trình của mỗi mô đun không được nói rõ trong câu hỏi này, hiện tại, chúng ta hãy giả sử rằng tất cả được giao cho các mô đun cấp thấp. Khi đó bạn sẽ thấy rằng, "cờ mất số hiệu thành phần, cờ kiểm tra cấu trúc, và cờ kiểm tra xác nhận quyền sử dụng" không được sử dụng. Có một cờ khác – cờ kiểm soát thấp, nhưng nó đã được (x) dùng. Trong câu truy vấn trạng thái tồn kho, nếu lượng hàng tồn của một thành phần nào đó là thấp (dưới một số lượng dự trữ tối thiểu nào đó), thành

phần đó được biểu thị màu đỏ, do đó, lượng hàng tồn kho thấp rõ ràng không được xử lí như một lỗi. Mặt khác, bạn cũng có thể hiểu rằng, ba cờ khác chỉ ra rằng một lỗi nào đó trong nội dung của giao dịch, đòi hỏi phải có một thông điệp báo lỗi được chỉnh sửa và trả về. Do đó, câu trả lời đúng là (h) "cờ mất số hiệu thành phần, cờ kiểm tra cấu trúc, cờ kiểm tra quyền sử dụng". Thực tế, có thể sẽ không có vấn đề gì ngay cả khi cờ kiểm soát thấp được chứa ở đây, nhưng không có một lựa chọn nào trong nhóm các câu trả lời chứa bốn cờ trên.

Trong một bài kiểm tra, sẽ có các câu hỏi liên quan đến thiết kế chương trình và thiết kế trong; tuy nhiên, không phải là "thiết kế" theo đúng nghĩa của từ đó. **Văn bản câu hỏi chứa một lí do cho mỗi câu trả lời đúng do đó, chỉ có một câu trả lời đúng khả dĩ.** Đồng thời cũng chỉ có một vài mẫu các câu hỏi. Do đó, hãy học các câu hỏi đã thi để học được phương pháp (cách tìm ra lí do cho những câu trả lời), và khi đó bạn có thể chắc chắn tìm thấy những câu trả lời đúng.

8 Phát triển chương trình

[Phạm vi câu hỏi]
Các ngôn lập trình (C, Java™),
Viết chương trình,
Môi trường phát triển
Các phương pháp kiểm thử,….

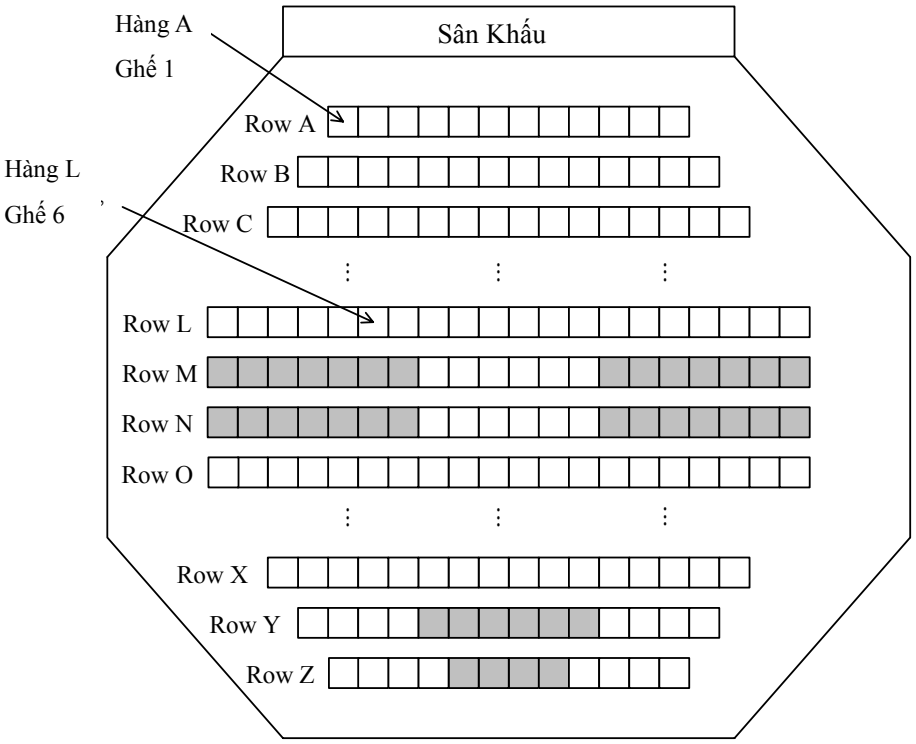
Câu hỏi 1 Lập trình C

Q1. Đọc mô tả chương trình và chương trình C dưới đây rồi trả lời các câu hỏi con.

[Mô tả chương trình]

Đây là một chương trình đặt chỗ ngồi cho một phòng hòa nhạc. Chương trình lấy số lượng liên tiếp các chỗ ngồi cần được đặt trước làm dữ liệu đầu vào, sau đó phân phối lại các chỗ ngồi, và trả về kết quả sau khi phân phối.

(1) Sự sắp xếp chỗ ngồi trong phòng hòa nhạc được thể hiện trong hình dưới đây:



Hình 8.1.1. Sắp xếp chỗ ngồi trong phòng hòa nhạc

(i) Có 26 hàng ghế được mã hóa từ A đến Z. Mỗi hàng có số lượng ghế khác nhau. Số ghế trong mỗi hàng được lưu trong mảng `cnum` theo thứ tự từ hàng A đến Z.

(ii) Số ghế bao gồm mã hàng ghế và một số. Số đó chỉ ra vị trí của ghế bắt đầu từ bên trái của mỗi hàng nhìn lên phía sân khấu. Ví dụ, ghế "hàng L, số 6" là ghế thứ 6 từ bên trái khi đứng đối diện với sân khấu, trong hàng thứ 12 (hàng L) từ phía trên sân khấu.

(iii) Các ghế trong vùng tô đậm trong hình vẽ đã được đặt trước.

(2) Số lượng ghế liên tiếp đã được chọn sẽ được lưu vào tham số `number` và truyền vào chương trình.

(3) Chương trình bắt đầu tìm kiếm từ chiếc ghế ngoài cùng bên trái trong hàng đầu tiên (hàng A, số 1) khi đứng đối diện sân khấu, tìm số lượng các ghế liên tiếp được yên cầu trong cùng một hàng, và chọn cụm các ghế trống đầu tiên liên tiếp nhau mà nó tìm thấy. Nếu các ghế không thể được tìm thấy trong hàng A, chương trình sẽ tìm tuần tự từ ghế ngoài cùng bên trái trong hàng B, C… cho đến khi tìm thấy cụm các ghế trống đầu tiên liên tiếp nhau. Nếu số lượng các ghế liên tiếp nhau được yêu cầu không thể tìm được, chương trình sẽ xác định các ghế không thể tìm được đó.

(4) Nếu số lượng yêu cầu của các ghế liên tiếp nhau được tìm thấy, chương trình trả về con trỏ trỏ đến cấu trúc số ghế (`SEAT`), cấu trúc này chứa mã hàng của các ghế, và số ghế tính từ ghế ngoài cùng bên trái (khi đứng đối diện với sân khấu). Nếu không tìm thấy, kết quả trả về con trỏ rỗng (`NULL`).

(5) Cấu trúc SEAT được mô tả như sau:

```
typedef struct {
    char  row;   /* mã hàng */
    int   no;    /* số      */
} SEAT;
```

(6) Trạng thái của mỗi ghế trong phòng hòa nhạc được lưu trong biến toàn cục `status`. Nếu ghế được tìm thấy, nó được đánh dấu là đã đặt trước.

`Status[i][j]=` { `' '`: Ghế trống.
{ `'R'`: Ghế đã có người.

Giá trị của chỉ số i trong khoảng từ 0 đến 25 tương ứng với các hàng từ A đến Z. Giá trị của chỉ số j trong khoảng từ 0 đến n-1 tương ứng với các số từ 1 đến n.

[Chương trình]

```
#define MAXNUM 30    /* Số lượng ghế tối đa trong một hàng*/
#define ROWNUM 26    /* Số lượng hàng */

typedef struct {
    char  row;       /* Mã hàng */
    int   no;        /* Số ghế */
} SEAT;

static char     rname[] = {"ABCDEFGHIJKLMNOPQRSTUVWXYZ"},
                status[ROWNUM][MAXNUM];
static int      cnum[ROWNUM] = {12, 14, 16, 18, 20, 22, 24, 26, 28,30, 30,
30, 30, 30, 30, 30, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12};
static SEAT     empty;

SEAT *book_seat(int);

SEAT *book_seat(int number)
{
    int ridx, cidx, eidx, flg = 0;

    for (ridx = 0; ridx < ROWNUM; ridx++) {
        for (cidx = 0; cidx <= cnum[ridx] - number; cidx++) {
            if (status[ridx][cidx] == ' ') {
                [ A ];
                for (eidx = [ B ]; cidx < eidx; eidx--)
                    if (status[ridx][eidx] == 'R') {
                        flg = 0;
                        break;
                    }
                if (flg == 1) break;
            }
        }
        if (flg == 1) break;
    }

    if (flg == 0)
        return NULL;
    for (eidx = cidx + number - 1; cidx <= eidx; eidx--)
        status[ridx][eidx] = 'R';
    empty.row = [ C ];
    empty.no  = [ D ];
    return &empty;
}
```

Câu hỏi

Từ các nhóm câu trả lời dưới đây, chọn các câu trả lời đúng để điền vào trong ô trống ☐ trong chương trình trên.

Nhóm câu trả lời cho A:

a) `cidx++`
b) `cidx--`
c) `cidx += number`
d) `cidx += number - 1`
e) `flg = 0`
f) `flg = 1`

Nhóm câu trả lời cho B:

a) `0`
b) `cidx`
c) `cidx + 1`
d) `cidx + number`
e) `cidx + number - 1`
f) `number`
g) `number – 1`

Nhóm câu trả lời cho C:

a) `ridx`
b) `ridx + 1`
c) `ridx - 1`
d) `rname[ridx]`
e) `rname[ridx + 1]`
f) `rname[ridx - 1]`

Nhóm câu trả lời cho D:

a) `cidx`
b) `cidx + 1`
c) `cidx - 1`
d) `ridx + cidx`
e) `ridx - cidx`

Đáp án câu 1

Tìm kiếm và chọn các ghế trống liên tiếp trong một phòng hòa nhạc.

Đáp án đúng

A – f, B – e, C – d, D – b

Giải thích

Đây là câu hỏi tương đối dễ dàng tương tự như quá trình tìm kiếm một xâu kí tự được lưu trong một mảng 2 chiều. Khi bạn đọc chương trình, đặc biệt chú ý tới ba điểm sau: "vai trò của mảng đang được dùng", "mục đích và nội dung của các biến đang được dùng" và "cấu trúc của các tiến trình". Sau đó bạn có thể sẽ phải điền vào các ô trống trong quá trình đọc chương trình.

Trong phần giải thích này, trước tiên chúng ta sẽ cố gắng nắm được tổng quan của chương trình, thiết lập các đối tượng sẽ được trả lời, và sau đó tạo các giải thích chi tiết ở những nơi cần thiết. Chương trình liệt kê dưới đây là hàm "book_seat()" được viết bằng ngôn ngữ mô phỏng thêm vào đó số hiệu các dòng để giải thích, các dòng phụ trợ này giúp chúng ta thiết lập miền và trật tự của từng khối tiến trình.

```
 1  SEAT *book_seat(int number)
 2  {
 3     oint ridx, cidx, eidx, flg = 0;
 4
 5     ■for (ridx = 0; ridx < ROWNUM; ridx++) {
 6        ■for (cidx = 0; cidx <= cnum[ridx] - number; cidx++) {
 7           ▲if (status[ridx][cidx] == ' ' ) {
 8                    a     ;
 9              ■for (eidx =     b     ; cidx < eidx; eidx--)
10                 ▲if (status[ridx][eidx] == 'R') {
11                          flg = 0;
12                          break;
13              ■   ▼}
14              if (flg == 1) break;
15           ▼}
16        ■}
17        if (flg == 1) break;
18     ■}
19     ▲if (flg == 0)
20     ▼   return NULL;
21     ■for (eidx = cidx + number - 1; cidx <= eidx; eidx--)
22     ■    status[ridx][eidx] = 'R' ;
23     empty.row =     c     ;
24     empty.no  =     d     ;
25     return &empty;
26   }
```

Đọc lướt qua [Mô tả chương trình] của yêu cầu chính. Lướt qua toàn bộ [Chương trình], tập trung vào "mảng", "biến", và "cấu trúc tiến trình", bạn sẽ thu được các thông tin dưới đây:

(Mảng)

`rname[]`	Chứa các hàng mã hóa từ A đến Z.
`Status[][]`	Chứa kí tự ' ' hoặc 'R' biểu diễn trạng thái ghế.
`cnum[]`	Chứa số lượng các ghế trên từng hàng. Được dùng trong dòng 6 theo mẫu `cnum[ridx]`.

(Cấu trúc)

`empty.row`	Chứa mã hàng của số lượng các ghế trống liên tiếp nhau được lựa chọn. Dữ liệu dạng ký tự.
`empty.no`	Chứa số thứ tự tính từ phía ngoài cùng bên trái của các ghế trống liên tiếp nhau được lựa chọn. Dữ liệu dạng số nguyên.

(Biến)

`Number`	Số lượng lượng các ghế trống liên tiếp nhau cần tìm; được truyền vào như một tham số.
`Ridx`	Dòng 7 có "`status[ridx][cidx]`", và dòng 10 có "`status[ridx][eidx]`", biến này tương ứng với chỉ số i để chỉ ra hàng ghế trong ma trận ghế.
`Cidx`	Dòng 7 có "`status[ridx][cidx]`", biến này tương ứng với chỉ số j để chỉ ra số ghế trong ma trận ghế.
`Eidx`	Trong các dòng 10 và 22 có "`status[ridx][eidx]`", biến này tương ứng với chỉ số j để chỉ ra số ghế trong ma trận ghế.
`Flg`	Các dòng 19-20 có "`if (flg == 0) return NULL;`", do đó biến `flg` là biến trạng thái để chỉ ra có tìm thấy số lượng các ghế trống liên tiếp được yêu cầu được tìm thấy hay không; giá trị 0 tức là không tìm thấy, và bằng 1 tức là tìm thấy.

(Cấu trúc tiến trình)

Các dòng 5-18	Khối này dùng để tìm các ghế trống liên tiếp nhau. Gồm 3 vòng lặp lồng nhau. Các dòng 5-18: biến `ridx` được dùng để duyệt các hàng từ trên xuống dưới trong quá trình tìm kiếm. Lines 6-16: biến `cidx` được dùng để kiểm tra các ghế trống từ trái sang phải. Lines 9-13: biến `eidx` được dùng để kiểm tra các ghế trống từ phải sang trái.
Các dòng 19-20	Khối này được thực hiện khi không tìm thấy các ghế trống liên tiếp nhau. Có thể dễ dàng nhìn thấy tại dòng 20: "`return NULL;`".
Các dòng 21-25	Khối này được thực hiện khi tìm thấy các ghế trống liên tiếp nhau. Các dòng 21-22: biến `eidx` được dùng để chuyển sang trạng thái ghế "đã có người" từ phải qua trái. dòng 23: Gán mã hàng (kiểu `char`) vào biến `empty.row`. dòng 24: Gán số (kiểu `int`) vào biến `empty.no`. dòng 25: Hàm trả về địa chỉ của biến "`empty`".

Trong quá trình tổ chức chương trình như bảng trên, bạn sẽ dễ dàng nhìn ra các câu hỏi và những quan sát về vai trò của mảng cũng như các vai trò và các chức năng của các biến.

(1) Vai trò của mảng "`rname[]`"

[Mỗ tả chương trình] không đem lại lời giải thích rõ ràng. Ngay cả ở trong [Chương trình], chúng ta cũng không thể tìm ra bất cứ tiến trình nào sử dụng mảng này. Điều đó gợi ý rằng sẽ có một trong các ô trống chứa mảng này.

(2) Cách sử dụng biến "`eidx`" và vai trò độc lập với biến "`cidx`"

Giá trị khởi tạo của biến "`eidx`" ở dòng 9 là giá trị trong ô trống B, vì vậy ngay tại thời điểm gán, giá trị đó chưa bị xóa. Liên hệ với điều này, chúng ta không cần biết ngay tại sao hai biến chỉ số "`cidx`" và "`eidx`" đều cùng được dùng cho các số ghế. Các dòng 21 và 22 bao gồm tiến trình tương tự như mẫu của các dòng 9 và 10, vì vậy nó có thể đưa ra cho chúng ta một đầu mối.

(3) Cách sử dụng biến "`flg`"

Dòng 3 khởi tạo biến bằng 0. Sau đó, trong dòng 11, nó lại được gán bằng 0, nhưng không có tiến trình nào giá trị của biến được gán là 1. Do đó, trong một ô trống nào đó giữa các dòng 3 và 11, giá trị của biến phải được gán là 1.

Dưới đây là tóm tắt của những việc được yêu cầu trong từng ô trống:

Ô trống A: Tiến trình được thực hiện đầu tiên khi biến trạng thái "`status[ridx][cidx]`" là ‘ ‘ (còn ghế trống)

Ô trống B: Giá trị khởi tạo của biến "`eidx`"

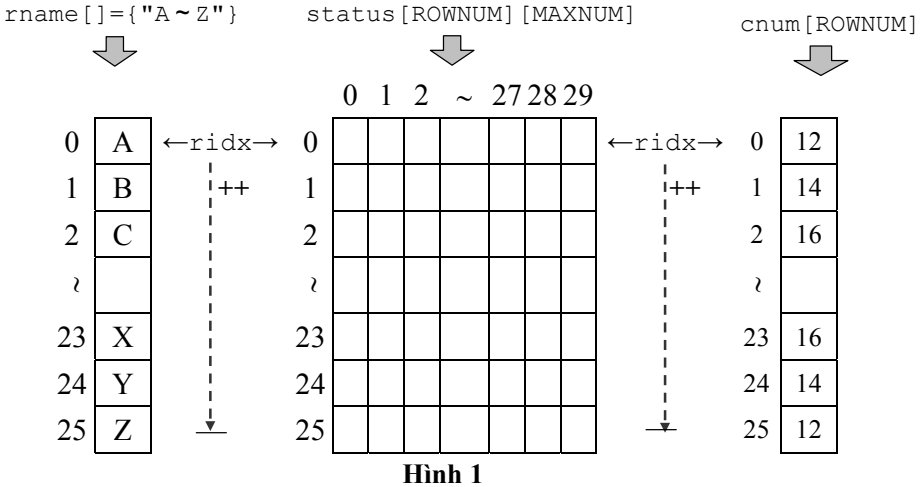
Ô trống C: Mã hàng tại số hiệu của các ghế trống liên tiếp nhau được tìm thấy; được gán cho biến "`empty.row`"

Ô trống D: Số lượng ghế ngoài cùng bên trái của số hiệu các ghế trống liên tiếp nhau; được gán cho biến "`empty.no`"

Bạn nên điền vào các ô trống khi bạn hiểu được những gì đã biết và những gì chưa biết với dữ liệu tới thời điểm hiện tại. Tiếp đó, chúng ta cùng điền vào các ô trống khi trả lời các câu hỏi con.

(1) Vai trò của mảng "`rname[]`" → Ô trống C

Để hiểu vai trò của mảng "`rname[]`", chúng ta có thể vẽ ra bảng dưới đây, các mảng được phân ra làm 3 có quan hệ lẫn nhau, sử dụng bảng như một lời gợi ý để chỉ ra số lượng các phần tử trên từng hàng và cách sử dụng chúng.



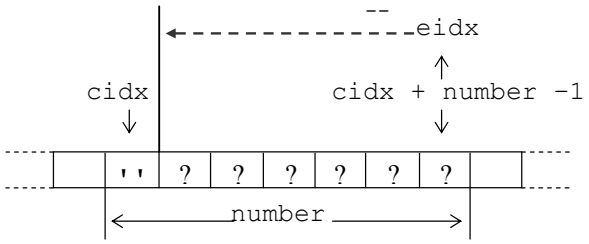
Hình 1

Một đầu mối khác là ở dòng 6 chỉ ra số lượng các ghế trong từng hàng qua cách dùng biến `cnum[ridx]`. Từ hình 1 và `cnum[ridx]`, chúng ta thấy rằng `rname[ridx]` biểu diễn mã hàng tương ứng với biến `ridx`. Nói một cách khác, mảng `rname[]`có chức năng chuyển đổi biến `ridx` thành mã hàng tương ứng. Nguyên nhân mà biến `ridx` cần phải chuyển đổi tương ứng sang mã hàng là trong dòng 23, mã hàng được gán cho biến `empty.row`. Từ những nhận xét trên, chúng ta thấy ý trả lời đúng cho ô trống C là (d) `rname[rdix]`. Các mảng như mảng `rname[]` và mảng `cnum[]` được gọi là các bảng chuyển đổi. Bạn nên nhớ cách dùng chúng vì đó là một kĩ thuật thường được sử dụng.

(2) Cách sử dụng biến "`eidx`" và vai trò độc lập với biến "`cidx`" → Các ô trống B và D

Để hiểu hoạt động của biến "`eidx`", chúng ta xác nhận giá trị cuối cùng, những lúc tăng và giảm, và cách dùng của biến. Trong dòng 9, biến "`eidx`" giảm cho đến khi giá trị tiếp theo của nó bằng với giá trị biến "`cidx`". Trong dòng tiếp theo, dòng 10, chương trình kiểm tra ghế được xác định bởi biến "`eidx`" đã có người ngồi hay chưa. Nói một cách khác, biến "`eidx`" hoạt động như một biến chỉ số để kiểm tra một dãy các ghế còn trống hay không, và giá trị khởi tạo của nó nên là giá trị chỉ ra phần tử ngoài cùng bên phải cần được kiểm tra. Bởi vậy, giá trị khởi tạo của biến "`eidx`" phải bằng tổng của biến "`cidx`" và số lượng ghế yêu cầu được lưu trong biến "`number`", trừ đi 1. Do đó, đáp án cho ô trống B là (e) `cidx + number - 1`.

Hình dưới đây minh họa cho những lập luận trên:



Hình 2

Đáp án chính xác cho ô trống B cũng có thể được tìm ra nhờ tiến trình thay đổi trạng thái của số lượng các ghế tìm được thành "đã đặt chỗ" bắt đầu từ ghế được chỉ định bởi biến "`cidx`" trong các dòng 21 và 22. Tuy nhiên, hai tiến trình này có khác nhau chút ít trong cách xác định các ghế liên tiếp nhau. Trong dòng 9, từ khi biết được ghế được chỉ định trong biến "`cidx`" là còn trống, biến "`eidx`" chỉ cần phải duyệt đến ghế ngay sau ghế mà biến "`cidx`" trỏ tới. Trong dòng 21, ngược với tiến trình trên, trạng thái của số lượng các ghế tìm thấy được chuyển thành "đã đặt chỗ", vì vậy biến "`eidx`" cần được duyệt lùi về giá trị của biến "`cidx`".

Bây giờ, chúng ta hãy tổng kết vai trò của các biến "`cidx`" và "`eidx`". Chúng ta thấy rằng biến "`cidx`" có chức năng chỉ ra ghế ngoài cùng bên trái của các ghế trống liên tiếp nhau, trong khi biến "`eidx`" có chức năng kiểm tra số lượng các ghế liên tiếp nhau cần tìm còn hay không. Như vậy, giá trị cần được gán cho biến "`empty.no`" trong dòng 24 là số ghế tương ứng với biến "`cidx`", đó là chỉ số của ghế ngoài cùng bên trái của các ghế trống liên tiếp nhau. Để chuyển đổi chỉ số sang số ghế, chúng ta cần cộng thêm 1 vàp chỉ số, nên ô trống D phải là `cidx + 1`. Vậy đáp án đúng là (b).

(3) Cách sử dụng biến "`flg`" → Ô trống A

Khi chúng ta lần theo hoạt động của biến "`flg`", chúng ta thấy rằng ban đầu biến được khởi tạo bằng 0 trong dòng 3 và lại được gán bằng 0 trong dòng 11. Do đó, giá trị 1 cần được gán cho biến ở chỗ nào đó giữa các dòng 3 và 11. Hoặc bạn có thể phát hiện ra ở tiến trình trong dòng 14, giá trị của biến được so sánh với 1, và kết luận rằng giá trị 1 cần được gán cho biến giữa các dòng 3 và 14. Dù bằng cách lập luận nào, việc gán giá trị cho biến phải được điền vào ô trống A. Bằng những lời giải thích của các dòng trên, chúng ta biết rằng giá trị 1 nên được gán ngay sau khi tìm ra ghế ngoài cùng bên trái của dãy ghế trống yêu cầu, nên chúng ta có thể xác nhận đấy là vị trí thích hợp. Vì thế, (f) `flg = 1` phải được điền vào ô trống A.

Bây giờ tất cả các ô trống đều đã được điền, chúng ta hãy kiểm tra chương trình hoạt động ra sao khi nó tìm thấy số lượng các ghế trống liên tiếp nhau được yêu cầu. Ở đây chúng ta bỏ qua trường hợp bài toàn vô nghiệm. Để thuận tiện hơn, chúng ta thêm vào các dòng phụ sử dụng ngôn ngữ mô phỏng cho chương trình ở trên vì thế chương trình có thể dễ dàng được kiểm tra.

Khi một ghế trống được tìm thấy trong dòng 7, ở ô trống A của dòng 8, giá trị 1 được gán cho biến `flg`. Vòng lặp của các dòng 9~13 kiểm tra ở đó có đủ số lượng ghế trống yêu cầu không. Nếu đủ, biến `flg` trả về giá trị 1, và chương trình đến dòng 14 để xử lý. Điều kiện của dòng 14 là đúng, chương trình thoát khỏi vòng lặp của các dòng 6~16 và nhảy đến dòng 17. Điều kiện của dòng 14 cũng đúng, chương trình thoát khỏi vòng lặp của các dòng 5~18 và nhảy đến dòng 19. Điều kiện của dòng 19 là sai, nên dòng 20 không được thực hiện, và chương trình chuyển đến tiến trình đặt ghế trước trong các dòng 21 và 22.

Như vậy, chúng ta chắc chắn rằng chương trình hoạt động chính xác khi `flg = 1` được điền vào ô trống A.

Câu hỏi được phân loại ở mức cơ bản, với độ khó thấp. Nếu bạn gặp khó khăn với việc trả lời câu hỏi này, thì hãy sử dụng các gợi ý ở trên để tìm cách đọc các chương trình và phương pháp trả lời các câu hỏi cho riêng bạn. Nếu bạn trả lời được trọn vẹn câu hỏi này, thì hãy dùng nó như một bài tập để trả lời các câu hỏi với mức độ khó cao hơn; tìm một vài hướng để dẫn dắt bạn đến những trả lời chính xác, và tập hợp chúng lại.

Câu hỏi 2 Lập trình C

Q2. Đọc mô tả chương trình và chương trình C dưới đây rồi trả lời các câu hỏi con 1 và 2.

[Mô tả chương trình 1]

Đây là chương trình đọc một chương trình nguồn được viết bằng ngôn ngữ lập trình C từ thiết bị vào chuẩn, loại bỏ các chú thích, sau đó đưa kết quả ra thiết bị ra chuẩn.

(1) Mô tả kí hiệu của các chương trình nguồn:

(i) "Các giải thích" được xử lý bởi chương trình là các xâu kí tự bắt đầu bằng "/*" và kết thúc bằng "*/", ngoài ra chúng còn bao gồm các hằng kí tự, các xâu kí tự và các chú thích.

(ii) Các kí tự hợp lệ được cho trong bảng.

Space	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(	8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[	k	{
,	<	L	\	l	|
-	=	M	]	m	}
.	>	N	^	n	~
/	?	O	_	o	

(iii) Các mô tả được cho dưới đây là không hợp lệ.

- Các giải thích lồng nhau

  Ví dụ `/* aaaaa /* bbbbbb */ ccccc */`

- Kí hiệu có 3 kí tự dành cho các kí tự đồ họa

  `??= ??( ??' ??< ??> ??) ??! ??-`

(iv) Không chứa các lỗi ngữ pháp.

(2) Chương trình 1 loại bỏ các chú thích theo đúng như thủ tục bên dưới. Từ đó chương trình dễ dàng xử lí việc phân tích các hằng kí tự, các xâu kí tự và các chú thích, chúng có thể không được nhận dạng chính xác phụ thuộc vào việc viết mã nguồn chương trình, đó là nguyên nhân gây lỗi.

(i) Khi tìm ra một dấu nháy đơn hay một dấu nháy kép, chương trình hiểu nó như là nơi bắt đầu của hằng kí tự hay xâu kí tự và sau đó sử dụng hàm `quote` để đọc và xuất ra xâu kí tự cho đến khi chương trình phát hiện ra dấu nháy đơn hay một dấu nháy kép kết thúc tương ứng.

(ii) Khi phát hiện ra xâu "`/*`", chương trình hiểu đó là bắt đầu đoạn chú thích và bỏ đi các kí tự trước xâu "`*/`" xuất hiện đầu tiên.

(3) Dưới đây là một ví dụ loại bỏ chú thích của chương trình 1.

Mã nguồn đọc dữ liệu

```
/* Chương trình dùng hàm fgets để xuất
 * một dòng từ file lên màn hình. */
#include <stdio.h>
int main( void )
{
   FILE *stream;    /* con trỏ file */
   char line[100];  /* dữ liệu đầu vào */

   if( (stream = fopen( "crt_fgets.txt", "r" )) != NULL )
   {
      if( fgets( line, 100, stream ) == NULL)
         printf( "fgets error\n" );  /* thông báo lỗi */
      else
         printf( "%s", line);
      fclose( stream );
   }
}
```

Ghi kết quả sau khi loại bỏ các chú thích

```
#include <stdio.h>
int main( void )
{
   FILE *stream;
   char line[100];

   if( (stream = fopen( "crt_fgets.txt", "r" )) != NULL )
   {
      if( fgets( line, 100, stream ) == NULL)
         printf( "fgets error\n" );
      else
         printf( "%s", line);
      fclose( stream );
   }
}
```

Hình. Ví dụ thực hiện loại bỏ chú thích

[Chương trình 1]

```
#include <stdio.h>
void quote( char );

main()
{
    int c1, c2;

    while ( (c1 = getchar()) != EOF ) {
        /* phát hiện dấu nháy đơn    */
        if ( c1 == '\'' ) quote( '\'' );
        /* phát hiện dấu nháy kép    */
        else if ( c1 == '\"' ) quote( '\"' );
        /* phát hiện dấu '/'         */
        else if ( c1 == '/' ) {
            c2 = getchar();
            /* khi kí tự tiếp theo là dấu sao */
            if ( c2 == '*' ) {
                /* loại bỏ xâu kí tự chú thích */
                while ( 1 ) {
                    while ( (c1 = getchar()) != '*' );
                    c2 = getchar();
                    if ( c2 == '/' ) break;
                }
            }
            /* trường hợp còn lại    */
            else {
                putchar(c1);
                putchar(c2);
            }
        }
        else putchar(c1); /* xuất ra một kí tự mã chương trình */
    }
}

void quote( char c )
{   /* lấy ra từng hằng kí tự và xâu kí tự    */
    char cc;

    putchar(c);
    while ( (cc = getchar()) != c ) putchar(cc);
    putchar(cc);
}
```

Câu hỏi con 1

Từ nhóm câu trả lời bên dưới, chọn đoạn mã làm sai lệch hoạt động khi được đem vào chương trình 1.

Nhón đáp án:

a) `/*   "aaaaaaa"  */`
b) `/* aaa  'a'     */`
c) `if ( c == '\'' ) {`
d) `printf( "  \'     " );`
e) `printf( "aaa  /*  comment  */ \n" );`

[Mô tả chương trình 2]

Để giải quyết vấn đề được chỉ ra trong phần (2) của **[Mô tả chương trình 1]**, chương trình 2 được viết lại dưới đây.

(1) Tiến trình được chia thành ba chế độ: hằng kí tự, xâu kí tự và chú thích.
(2) Sự xuất hiện của dấu nháy đơn báo hiệu bắt đầu hoặc kết thúc của "chế độ hằng kí tự". Tuy nhiên, nó không được áp dụng cho đoạn mã được mô tả mở rộng bởi việc sử dụng kí tự "\", cho đoạn mã ở trong xâu kí tự, hoặc đoạn mã ở trong phần chú thích.
(3) Sự xuất hiện của dấu nháy kép báo hiệu bắt đầu hoặc kết thúc của "chế độ xâu kí tự". Tuy nhiên, nó không được áp dụng cho đoạn mã được mô tả mở rộng bởi việc sử dụng kí tự "\", cho đoạn mã ở trong hằng kí tự, hoặc đoạn mã ở trong phần chú thích.
(4) Sự xuất hiện của cặp xâu "/*" và "*/" báo hiệu bắt đầu và kết thúc của "chế độ chú thích". Tuy nhiên, nó không được áp dụng cho đoạn mã ở trong hằng kí tự hoặc cho đoạn mã ở trong xâu kí tự.

[Chương trình 2]

```
#include <stdio.h>
main()
{
    int c1, c2;
    int c_mode = 0;   /* khởi tạo kết thúc chế độ chú thích     */
    int quote1 = 0;   /* khởi tạo kết thúc chế độ hằng kí tự    */
    int quote2 = 0;   /* khởi tạo kết thúc chế độ xâu kí tự     */

    for ( c1 = getchar(); ( c2 = getchar()) != EOF; c1 = c2 ) {

        if ( !c_mode ) {       /* khi hết chế độ chú thích */
         /* phát hiện nếu kí tự \ ở trong một hằng kí tự hoặc trong một xâu kí tự.*/
            if ( [A] && c1 == '\\' ) {
                putchar(c1);
                putchar(c2);
                c2 = getchar();
                continue;
            }
            /* kiểm tra nếu dấu nháy đơn không nằm trong xâu kí tự */
            else if ( !quote2 && c1 == '\'' )
                [B];
            /* kiểm tra nếu dấu nháy kép không nằm trong hằng kí tự */
            else if ( !quote1 && c1 == '\"' )
                [C];
            /* kiểm tra nếu kí tự / và * không nằm tron hằng kí tự */
            /* và không nằm trong xâu kí tự */
            else if ( [D] && c1 == '/' && c2 == '*' ) {
                [E];
                c2 = getchar();
                continue;
            }
            putchar(c1);
        }

        else {
            if ( c1 == '*' && c2 == '/' ) {  /* kết thúc câu chú thích? */
                [E];
                c2 = getchar();
            }
        }
    }
    putchar(c1);
}
```

Câu hỏi con 2

Từ các nhóm câu trả lời dưới đây, chọn các câu trả lời đúng để điền vào trong ô trống [ ] trong chương trình 2.

Nhóm câu trả lời cho A và D:

a) `!quote1`
b) `!quote2`
c) `(!quote1 || !quote2)`
d) `(!quote1 && !quote2)`
e) `(quote1 || quote2)`
f) `(quote1 && quote2)`

Nhóm câu trả lời cho B, C và E:

a) `c_mode = !c_mode`
b) `c_mode = quote1 && quote2`
c) `quote1 = !quote1`
d) `quote1 = !quote2`
e) `quote1 = quote2`
f) `quote2 = !quote1`
g) `quote2 = !quote2`
h) `quote2 = quote1`

Đáp án câu 2

Loại bỏ các chú thích trong mã nguồn của ngôn ngữ C

Đáp án đúng

[Câu hỏi con 1] c

[Câu hỏi con 2] A – e, B – c, C – g, D – d, E – a

Giải thích

Đây là câu hỏi liên quan đến chương trình đọc mã nguồn được viết bằng ngôn ngữ C, loại bỏ các chú thích, và đưa ra kết quả. Nội dung của câu hỏi chiếm số lượng trang khá lớn, nhưng bạn có thể trả lời được câu hỏi này trong thời gian ngắn bằng việc sử dụng các mô tả chương trình và các chú thích ở trong chương trình, vì thế hãy cố gắng trả lời các câu hỏi này mà không lo ngại về độ dài của chúng.

[Câu hỏi con 1]

Đây là câu hỏi tìm trường hợp gây ra lỗi khi đem chương trình ra thực hiện. Không cần đọc [Chương trình 1], từ thủ tục được giải thích trong phần (2) của [Mô tả chương trình 1], bạn có thể chỉ ra trường hợp xảy ra lỗi. Cụ thể hơn, chúng ta hãy đọc phần (2) của [Mô tả chương trình 1]: "chúng có thể không được nhận dạng chính xác … đó là nguyên nhân gây lỗi". Chúng ta có thể tìm ra đáp án dựa trên chú thích đó, vì việc này được làm theo chú thích của thủ tục loại bỏ các chú thích. Để kiểm tra chương trình hoạt động ra sao với từng lựa chọn trong nhóm trả lời, chúng ta hãy xem dưới đây:

(a) và (b) rơi vào trường hợp (ii), xóa mọi kí tự xâu "`/*`" đến xâu "`*/`" được hiểu là các chú thích. Thao tác này không phát sinh bất kì lỗi nào.

(c) rơi vào trường hợp (i) xuất ra mọi kí tự giữa dấu nháy đơn đầu tiên (`'`) và dấu nháy đơn thứ hai được hiểu là hằng kí tự. Sau đó, dấu nháy đơn thứ ba được hiểu là bắt đầu của hằng kí tự khác, kí tự sẽ được đưa ra nguyên vẹn. Tuy nhiên, trong thực tế, dấu nháy đơn thứ ba chỉ ra vị trí kết thúc của hằng kí tự, vì thế nó không được nhận dạng chính xác. Do đó, khi xâu "`/*`" xuất hiện trong dòng tiếp theo chỉ ra các lời chú thích thì cũng được đưa ra mà không cần kiểm tra, đây là nguyên nhân sinh ra lỗi. Một ví dụ đặc trưng được chỉ ra dưới đây, các chú thích được xuất ra.

```
if ( c == '\'' ) {
/* Is "c" a single quotation mark?  */
```

(d) và (e) rơi vào trường hợp (i), xuất ra mọi kí tự giữa dấu nháy kép đầu tiên (`"`) và dấu nháy kép thứ hai được hiểu là xâu kí tự. Thao tác này không phát sinh bất kì lỗi nào.

Vì thế, đáp án đúng là (c).

Giờ đây, chúng ta đã trả lời xong câu hỏi mà không thực sự cần đọc chương trình; đáp án thu được chỉ dựa vào phần mô tả chương trình. Trong bài thi thực tế, câu hỏi của dạng này nên được trả lời theo cách trên, và sau đó, nếu còn thừa thời gian, bạn có thể đọc kĩ chương trình để kiểm tra lại đáp án.

[Câu hỏi con 2]

Các nhóm câu trả lời được chia thành các câu cho các ô trống A và D và các câu cho các ô trống B, C, và E. Có thể có một vài đọc giả băn khoăn khi nhìn thấy nhóm trả lời A và D. Mỗi một ô trống đó ở trong một nhánh lệnh "if". Nhìn chung, điều kiện thường được biểu diễn bởi việc liên kết nội dung biến với giá trị bởi việc sử dụng phép toán quan hệ, ví dụ như "`c1 == '\\'`". Tuy nhiên, không có lựa chon nào trong nhóm trả lời của các ô trống A và D có dạng đó. Tất cả chúng chỉ có các biến hoặc dấu chấm than (`!`) đặt trước các biến, không kèm theo đó các phép toán quan hệ.

Các mô tả điều kiện trong dạng trên là cơ bản về mặt ngữ pháp, nhưng phần lớn thí sinh có thể chưa từng gặp trước đó bởi vì việc dùng kí hiệu này đôi khi bị cấm bởi các luật viết mã tại các môi trường phát triển, hoặc vì ý nghĩa ban đầu có thể được viết bằng các phép toán quan hệ khác. Trong ngôn ngữ C, các điều kiện "`!= 0`" và "`== 0`" có thể bị bỏ qua trong mô tả. Ví dụ, nếu một biểu thức điều kiện là "`quote1 != 0`", nó có thể được viết là "quote1", bỏ qua phép toán quan hệ và bất cứ thứ gì sau đấy. Nếu một biểu thức điều kiện là "`quote1 == 0`", nó tương đương với cách viết "`!quote1`". Đó là vì các dạng dữ liệu trong C không chứa kiểu lô-gíc; thay vào đó, kiểu số nguyên (`int`) được sử dụng, giá trị 0 là sai và mọi giá trị khác 0 có nghĩa là đúng. Vì thế, một phép toán quan hệ trả về giá trị khác 0 nếu biểu thức đã cho là đúng và trả về 0 nếu nó là sai.

Bây giờ, dù là bạn không biết vấn đề gì đã được trình bày ở trên, bạn vẫn có thể hình dung ra như sau: Để ý lệnh "`if`" bên trên hai dòng ô trống A với chú thích "khi hết chế độ chú thích". Điều kiện của nó được chỉ ra là "`!c_mode`". Tại đó, người đọc nên hiểu rằng "`!c_mode`" nghĩa là "`c_mode == 0`" (bởi vì giá trị khởi tạo của biến c_mode là kết thúc chế độ, và giá trị của nó là 0).

Trong phần giải thích, đầu tiên chúng ta sẽ thảo luận về các ô trống A và D và sau đó giải quyết các ô trống B, C, và E. Các tiến trình trong các ô trống của cả hai nhóm được giải thích trong các câu chú thích, vì thế chúng ta chỉ cần chuyển chúng thành các biểu thức trong chương trình. Nói theo kĩ thuật, bất kì số khác 0 nào cũng có thể được gán là đúng, nhưng trong các chú thích ở đây, chúng ta sẽ dùng giá trị "1".

Ô trống A

Dựa vào các chú thích, câu lệnh "`if`" chứa ô trống A đang kiểm tra nếu kí tự "`\`" nằm trong một hằng kí tự hay một xâu kí tự. Ở đây, "việc kiểm tra kí tự `\`" tương ứng với biểu thức "`c1 == '\\'`", vì thế ô trống A sẽ tương ứng với "trong một hằng kí tự hoặc trong một xâu kí tự". Chúng ta cần phải chuyển sang một biểu thức trong ngôn ngữ C. "trong …" gợi ý rằng chế độ hằng kí tự hoặc chế độ xâu kí tự sẽ được bắt đầu (1). Việc chuyển đổi đó theo cách dễ hiểu, chúng ta có (`quote1 != 0 || quote2 != 0`), nhưng nó không có trong nhóm câu trả lời. Tuy vậy, khi xem lại nhóm câu trả lời một lần nữa, chúng ta thấy biểu thức (`quote1 || quote2`) dựa trên cơ sở là chế độ bắt đầu (1) và chế độ kết thúc (0) của `quote1` và `quote2` tương ứng với giá trị đúng (1) và sai (0). Vì vậy, đáp án đúng cho ô trống A là (e).

Ô trống D

Dựa vào các chú thích, câu lệnh "`if`" chứa ô trống D đang kiểm tra nếu "kí tự / và * không nằm trong hằng kí tự và xâu kí tự". Ở đây, việc kiểm tra "/" và "*" tương ứng với biểu thức "`c1 == '/' && c2 == '*'`", vì thế ô trống D sẽ tương đương với "không nằm trong hằng kí tự và xâu kí tự". Chúng ta cần phải chuyển sang một biểu thức trong ngôn ngữ C. "Không…" hiểu là điều kiện của chế độ hằng kí tự là kết thúc (0). Do đó, biểu thức phủ định "`!quote1`" là cần thiết. Tương tự như vậy, đó là điều kiện của chế độ xâu kí tự là kết thúc (0), vì thế biểu thức phủ định "`!quote2`" cũng là cần thiết. Điều kiện "and" hiểu là chúng ta cần liên kết biểu thức `!quote1` và biểu thức `!quote2` bằng phép toán "`&&`". Vì vậy, đáp án đúng sẽ là (`!quote1 && !quote2`), đó là ý (d).

Ô trống B

Dựa vào các chú thích, ô trống B là tiến trình được thực hiện khi "chế độ chú thích kết thúc" và khi không có "kí tự \ được phát hiện trong hằng kí tự hoặc xâu kí tự" và khi "dấu nháy đơn được tìm thấy nhưng không nằm trong một xâu kí tự". Dấu nháy đơn (') được dùng để chỉ ra chỗ bắt đầu và kết thúc của một hằng kí tự, vì thế việc cần phải làm là chuyển chế độ hằng kí tự. Cụ thể hơn, nếu chế độ đó là bắt đầu (1), nó cần được chuyển sang chế độ kết thúc (0); và ngược lại. Nói cách khác, giá trị phủ định của biến `quote1` được gán cho biến `quote1`, do đó chúng ta cần biểu thức "`quote1 = !quote1`". Đáp án đúng cho ý B là (c).

Nếu bạn không tin tưởng vào kết luận rằng đó là điều kiện để bắt đầu hoặc kết thúc một hằng kí tự, lưu ý đến điều kiện đối lập, ví dụ, vị trí nơi dấu nháy đơn được tìm thấy nhưng không chỉ ra nơi bắt đầu hoặc kết thúc của một hằng kí tự. Điều đó xảy ra nếu "chế độ chú thích bắt đầu" hoặc ngay sau "kí tự \ được tìm ra bên trong hằng xâu kí tự hoặc xâu kí tự" hoặc nếu một dấu nháy đơn được tìm thấy "bên trong một xâu kí tự". Kiểm tra điều kiện phủ định, và bạn có thể kiểm tra điều kiện bắt đầu hoặc kết thúc một hằng kí tự.

Ô trống C

Ô trống C có cùng cách suy luận như ô trống B.

Dựa vào các chú thích, tiến trình nên được điền vào ô trống C là khi "chế độ chú thích kết thúc" và khi không có "kí tự \ được tìm thấy trong một hằng xâu kí tự hoặc trong một xâu kí tự" và khi "dấu nháy kép được tìm ra nhưng không nằm trong hằng kí tự". Dấu nháy kép (") được dùng để chỉ ra vị trí bắt đầu hoặc kết thúc của một xâu kí tự, do đó việc cần thiết phải được làm ở đây là chuyển chế độ xâu kí tự. Vì vậy, biểu thức "`quote2 = !quote2`" được điền vào ô trống C, đáp án là (g).

Ô trống E

Ô trống E xuất hiện ở 2 vị trí. Một là khi "chế độ chú thích kết thúc" và khi "kí tự / và * không được tìm thấy bên trong một hằng kí tự và một xâu kí tự". "Kí tự / và *" (tức là xâu "`/*`") chỉ ra vị trí bắt đầu của một chú thích, do đó, việc cần phải làm ở đây là chuyển chế độ chú thích từ kết thúc (0) sang bắt đầu (1). Tại vị trí còn lại là khi không phải "chế độ chú thích kết thúc" (hiểu là khi đó chế độ là bắt đầu) và tại nơi "kết thúc chú thích". Do đó, điều kiện của câu lệnh "`if`" bị đảo ngược, tức là, để tìm ra "* và /" hay "`*/`", nơi kết thúc của chú thích. Việc cần phải làm ở đây là chuyển chế độ chú thích từ bắt đầu (1) sang kết thúc (0). Đoạn mã có thể làm cả hai việc này là "`c_mode = !c_mode`". Vì vậy, đáp án cho ô trống E là (a).

(Các chú ý thêm)

Cuối cùng, mặc dù không liên quan trực tiếp đến các câu hỏi, chúng ta sẽ thêm một lưu ý ngắn gọn ở đây về kí hiệu có 3 kí tự dành cho các kí tự đồ họa xuất hiện trong phần (iii) của [Mô tả chương trình 1].

Kí hiệu có 3 kí tự dành cho các kí tự đồ họa còn được gọi là ba kí tự đồ họa liên tiếp, và nó liên quan đến tập hợp các xâu kí tự để biểu diễn những kí tự như "#" và "[" trong môi trường mà ở đó các kí tự này không thể sử dụng được. Ví dụ, "??=" biểu diễn "#", "??(" biểu diễn "[" và "??'" biểu diễn "^".

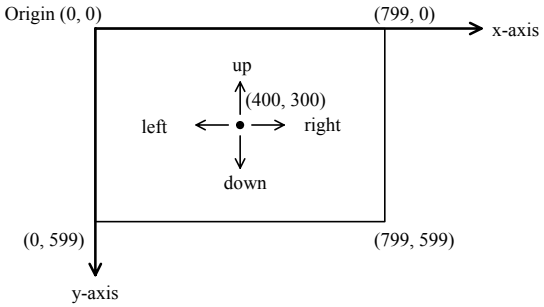
Câu hỏi 3 Lập trình C

Q3. Đọc mô tả chương trình và chương trình C dưới đây rồi trả lời các câu hỏi con.

[Mô tả chương trình]

Hàm `execute` thực hiện vẽ các đường thẳng sử dụng con trỏ trên màn hình.

(1) Một màn hình bitmap với kích thước chiều rộng 800 pixels và chiều cao 600 pixels. Hình 1 biểu diễn hệ tọa độ của màn hình. Con trỏ (• trong Hình.1) có tọa độ và hướng được thể hiện trên màn hình. Con trỏ có khả năng di chuyển theo 4 hướng: lên trên, xuống dưới, sang trái, sang phải. Con trỏ sẽ được vẽ khi nó di chuyển.



Hình 1 Hệ tọa độ của màn hình và trạng thái ban đầu của con trỏ

(2) Con trỏ được biểu diễn bằng một biến `mark` có kiểu là bản ghi `MARKER`. Khi chương trình bắt đầu, vị trí của con trỏ được gán giá trị (400, 300) và di chuyển hướng lên trên.

```
typedef struct {
    int x;   /* tọa độ x của con trỏ */
    int y;   /* tọa độ y của con trỏ */
    int dir; /* hướng di chuyển  0:phải, 1:lên, 2:trái, 3:xuống */
} MARKER;
MARKER mark = {400,  /* tọa độ x ban đầu */
               300,  /* tọa độ y ban đầu */
               1     /* hướng di chuyển ban đầu (lên) */
               };
```

(3) Con trỏ được điều khiển thông qua các lệnh. Mỗi lệnh được biểu diễn bằng một bản ghi kiểu `INST` bao gồm mã lệnh và giá trị.

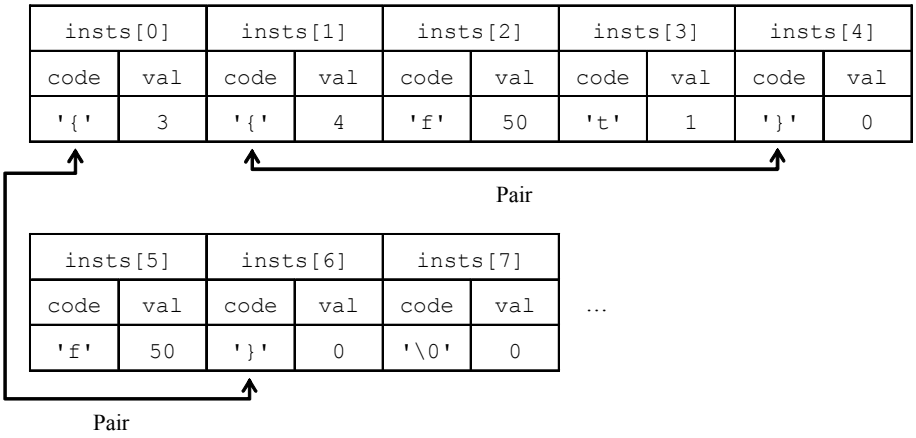
```
typedef struct {
    char code;   /* mã lệnh */
    int  val;    /* giá trị */
} INST;
```

Các lệnh được lưu trữ trong mảng `insts` (mảng các bản ghi kiểu `INST`) và được sắp xếp lần lượt theo thứ tự thực hiện.

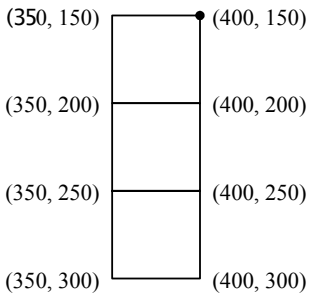
(4) Bảng sau liệt kê tất cả các mã lệnh và mô tả tương ứng.

Mã Lệnh	Mô tả
`{`	Lệnh lặp, lặp `val` lần các lệnh bắt đầu từ lệnh ngay sau nó tới lệnh cuối cùng trước lệnh có mã `'}'` (tạo thành một cặp `'{}'`). `val` là một số nguyên có giá trị lớn hơn 1.
`t`	Lệnh chuyển hướng, nhiệm vụ thay đổi hướng di chuyển của con trỏ một góc 90° × `val` theo hướng ngược chiều kim đồng hồ. `val` là một số nguyên dương.
`f`	Lệnh di chuyển, di chuyển con trỏ theo hướng hiện tại `val` pixels và vẽ một đường thẳng từ vị trí ban đầu tới vị trí đích. `val` là một số nguyên.
`}`	Đánh dấu kết thúc một đoạn lệnh lặp. `val` không sử dụng.
`\0`	Đánh dấu kết thúc việc thực hiện điều khiển con trỏ. `val` không sử dụng.

(5) Hình 3 biểu diễn kết quả của việc thực hiện hàm `execute` với các lệnh được lưu trong mảng `insts` trong Hình 2. Chú ý, các giá trị tọa độ trong Hình 3 được thêm vào với mục đích giải thích chứ không được thực sự ghi ra trong kết quả.



Hình 2. Ví dụ các lệnh được lưu trữ trong mảng `insts`



Hình 3. Kết quả thực hiện các lệnh trong Hình 2

(6) Hàm sau được sử dụng để vẽ đường thẳng:

```
void drawLine( int x1, int y1, int x2, int y2 );
```

Chức năng: vẽ một đường thẳng nối tọa độ (`x1`, `y1`) và tọa độ (`x2`, `y2`) trên màn hình.

(7) Các hàm sau thực hiện các chức năng liên quan tới con trỏ:

```
void eraseMarker( MARKER mark );
```

Chức năng: không thể hiện con trỏ trên màn hình.

```
void paintMarker( MARKER mark );
```

Chức năng: thể hiện con trỏ trên màn hình.

(8) Ngay khi con trỏ không được biểu diễn trên màn hình, tọa độ và hướng di chuyển của nó vẫn không thay đổi.

[Chương trình]

```
#define INSTSIZE 100 /* số lệnh tối đa */
#define STACKSIZE 50 /* số vòng lặp tối đa lồng nhau */

typedef struct {
    int x;       /* tọa độ x của con trỏ */
    int y;       /* tọa độ y của con trỏ */
    int dir;     /* hướng di chuyển 0:phải, 1:lên, 2:trái, 3:xuống */
} MARKER;

typedef struct {
    char code;   /* mã lệnh */
    int  val;    /* giá trị */
} INST;

typedef struct {
    int opno;    /* phần tử thứ No. của mảng insts bắt đầu vòng lặp */
    int rest;    /* số vòng lặp còn lại */
} STACK;
void drawLine( int, int, int, int );
```

```
void eraseMarker( MARKER );
void paintMarker( MARKER );

INST insts[INSTSIZE];     /* mảng chứa các lệnh */
MARKER mark = {400,       /* tọa độ x ban đầu */
               300,       /* tọa độ y ban đầu */
               1          /* hướng di chuyển ban đầu (lên) */
              };

void execute(){
    STACK stack[STACKSIZE];
    int opno = 0;    /* phần tử thứ No. của mảng insts chứa lệnh được thực hiện */
    int spt = -1;    /* con trỏ stack */
    int dx, dy;
    paintMarker( mark );
    while( insts[opno].code != '\0' ){
        switch( insts[stack].code ){
            case '{':
                    stack[ A ].opno = opno;
                    stack[spt].rest = insts[opno].val;
                    break;
            case 't':
                    mark.dir = B ;
                    break;
            case 'f':
                    eraseMarker( mark );
                    dx = ( mark.dir % 2 == 0 ) ? C ;
                    dy = ( mark.dir % 2 == 0 ) ? D ;
                    drawLine( mark.x, mark.y,
                              mark.x + dx, mark.y + dy );
                    mark.x += dx;
                    mark.y += dy;
                    paintMarker( mark );
                    break;
            case '}':
                    if ( stack[spt].rest E ){
                        opno = stack[spt].opno;
                        stack[spt].rest--;
                    } else {
                        F ;
                    }
                    break;
        }
        G ;
    }
}
```

Câu hỏi con

Từ các nhóm câu trả lời dưới đây, chọn các đáp án đúng để điền vào trong ô trống ☐ trong chương trình trên.

Nhóm câu trả lời cho A:

a) `++spt` b) `--spt` c) `spt`
d) `spt++` e) `spt--`

Nhóm câu trả lời cho B:

a) `( mark.dir + insts[opno].val ) % 2`
b) `( mark.dir + insts[opno].val ) % 3`
c) `( mark.dir + insts[opno].val ) % 4`
d) `mark.dir + insts[opno].val`
e) `mark.dir + insts[opno].val % 2`
f) `mark.dir + insts[opno].val % 3`
g) `mark.dir + insts[opno].val % 4`

Nhóm câu trả lời cho C và D:

a) `( 1 - mark.dir ) * insts[opno].val : 0`
b) `( 2 - mark.dir ) * insts[opno].val : 0`
c) `( mark.dir - 1 ) * insts[opno].val : 0`
d) `( mark.dir - 2 ) * insts[opno].val : 0`
e) `0 : ( 1 - mark.dir ) * insts[opno].val`
f) `0 : ( 2 - mark.dir ) * insts[opno].val`
g) `0 : ( mark.dir - 1 ) * insts[opno].val`
h) `0 : ( mark.dir - 2 ) * insts[opno].val`
i) `0 : mark.dir * insts[opno].val`
j) `mark.dir * insts[opno].val : 0`

Nhóm câu trả lời cho E:

a) `< 0` b) `< 1` c) `== 0`
d) `> 0` e) `> 1`

Nhóm câu trả lời cho F và G:

a) `mark.dir++` b) `mark.dir--` c) `opno++`
d) `opno--` e) `spt++` f) `spt--`

Đáp án câu 3

Điều khiển con trỏ vẽ màn hình

Đáp án đúng

A – a, B – c, C – a, D – h, E – e, F – f, G – c

Giải thích

Đây là câu hỏi liên quan tới một chương trình điều khiển con trỏ vẽ trên màn hình theo các lệnh được lưu trữ trong một mảng. Cấu trúc của chương trình đơn giản và dễ hiểu. Các hàm và các biến được sử dụng, ngoại trừ `dx` và `dy`, đều đã được giải thích rõ ràng.

Trong chương trình này, vai trò của cấu trúc dữ liệu và các biến được sử dụng khá phức tạp (cấu trúc mảng được sử dụng). Các chương trình dạng này nên được xem xét các vấn đề lần lượt như sau: vai trò của mảng bản ghi và các biến → tổ chức cấu trúc của chương trình → hiểu nội dung của chương trình. Thêm vào đó, để hiểu nội dung của chương trình, có thể sử dụng dữ liệu mẫu trong phần câu hỏi. Để giải thích cụ thể hơn, phía dưới là đoạn chương trình của hàm "`execute()`" cùng với các chú thích thêm gồm số thứ tự của dòng, các khối lệnh và ghi chú.

```
 1   void execute(){
 2
 3     STACK stack[STACKSIZE];
 4     int opno = 0; /* phần tử thứ No. của mảng insts chứa lệnh được thực hiện */
 5     int spt = -1; /* con trỏ stack */
 6     int dx, dy;
 7
 8     paintMarker( mark );
 9
10     while( insts[opno].code != '\0' ) {
11         switch( insts[opno].code ) {
12               case '{':
13                     stack[ [ a ] ].opno = opno;
14                [1]  stack[spt].rest = insts[opno].val;
15                     break;
16               case 't':
17                     mark.dir = [ b ];
18                [2]  break;
19               case 'f':
20                     eraseMarker( mark ) ;
21                     dx = ( mark.dir % 2 == 0 ) ? [ c ];
22                     dy = ( mark.dir % 2 == 0 ) ? [ d ];
23                     drawLine( mark.x, mark.y,
24                [3]        mark.x + dx, mark.y + dy );
25                     mark.x += dx;
26                     mark.y += dy;
27                     paintMarker( mark );
28                     break;
29               case '}':
30                     if ( stack[spt].rest [ e ] ){
31                     opno = stack[spt].opno;
32                [4]  stack[spt].rest--;
33                     } else {
34                           [ f ];
35                     }
36                     break;
37         }  <-- Kết thúc switch
38           [ g ] ;
39     }  <-- Kết thúc while
40   }
```

Hàm "`execute()`"

(Tóm tắt các cấu trúc dữ liệu và biến được sử dụng)

Khi đọc từ đầu chương trình tới dòng 6, từ nội dung của câu hỏi và các chú thích, vai trò và nội dung của mảng cấu trúc và các biến được làm rõ.

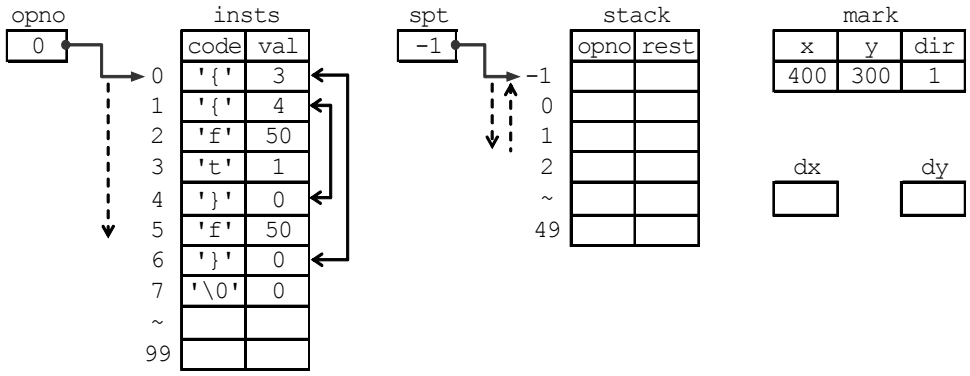
Biến "opno" : chỉ số của mảng "insts", mảng chứa toàn bộ các lệnh dành cho con trỏ. Dòng 4 khởi tạo giá trị 0 cho biến này. Đây là một biến đóng vai trò là chỉ số dùng khi duyệt mảng, do vậy trong chương trình biến này được tăng dần để duyệt và thực hiện các lệnh trong mảng "insts" theo đúng thứ tự.

Biến "spt" : chỉ số của mảng "stack"; dòng 5 khởi tạo giá trị -1 cho biến này. Cho tới thời điểm này, mục đích sử dụng cụ thể của mảng "stack" chưa thực sự rõ ràng, nhưng dựa vào chú thích tại dòng mảng "stack" được khai báo, có thể dự đoán rằng mảng này được sử dụng để điều khiển số lượng các vòng lặp lồng nhau. Thêm nữa, vì biến "spt" được khởi tạo giá trị -1, vì vậy biến "spt" có thể sẽ được tăng lên trước khi quá trình lưu trữ vào "stack" được thực hiện. Cuối cùng, vì là một ngăn xếp nên có thể dự đoán rằng biến này sẽ có thể được giảm đi tương ứng khi thực hiện quá trình lấy dữ liệu ra khỏi ngăn xếp.

Biến bản ghi "mark" : với các thành phần x và y tương ứng với tọa độ x và y của con trỏ, vì vậy biến này được sử dụng trong quá trình vẽ khi gặp lệnh 'f', có thể đoán được rằng sau khi vẽ sẽ là công đoạn cập nhật tọa độ mới cho con trỏ. Thành phần "dir" chỉ hướng di chuyển hiện thời của con trỏ, được sử dụng trong quá trình vẽ khi gặp lệnh 'f', và được cập nhật khi gặp lệnh 't'.

Biến "dx" and "dy" : vai trò của các biến này chưa được rõ ràng với những thông tin đã có. Qua tên biến, có thể dự đoán chúng được sử dụng trong quá trình vẽ khi gặp lệnh 'f', vai trò của các biến này sẽ được làm rõ trong phần sau.

Hình 1 tóm tắt lại các giải thích trên. Để dễ dàng hình dung hoạt động của biến "opno", mảng "insts" được biểu diễn thẳng đứng như trong hình.



Hình 1: Vai trò và nội dung của các mảng và biến được sử dụng

(Tóm tắt cấu trúc chương trình)

Đoạn quan trọng nhất của hàm "execute" được thực hiện bên trong một vòng lặp "while-do" với điều kiện kết thúc là khi giá trị của "insts[opno].code" khác '\0'. Bên trong vòng lặp "while-do" là một câu lệnh rẽ nhánh "switch-case" với 4 nhánh tương ứng với giá trị của "insts[opno].code" và khoảng trắng G. Trong câu lệnh switch-case, có bốn đoạn xử lý [1][2][3][4], mỗi đoạn tương ứng với một quá trình được chỉ ra trong mã lệnh. Khoảng trắng G nằm bên ngoài câu lệnh switch-case, do đó nó sẽ được thực hiện sau khi tất cả các lệnh được lưu trong mảng đã hoàn thành. Dựa vào điều này, có thể đoán rằng khoảng trắng G chính là quá trình tăng giá trị của biến chỉ số "opno" đối với mảng "insts", điều này sẽ được khẳng định sau khi

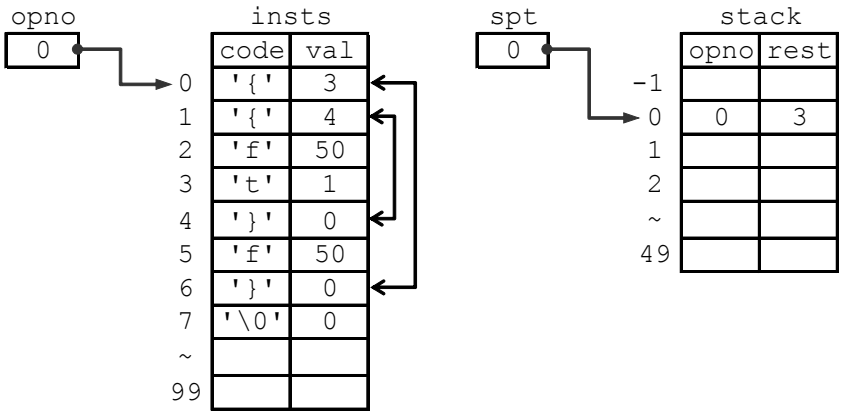
hiểu rõ nội dung cấu trúc của cả chương trình.
Cấu trúc của hàm "execute" rất đơn giản. Mặc dù vậy, để tránh lỗi và nhầm lẫn, có một lời khuyên là nên tự mình thêm vào các dòng phân chia chú thích như trên khi làm bài. Ví dụ, khoảng trắng G ngay lập tức được thấy bên ngoài câu lệnh switch-case. Từ đó, đáp án đúng cho G cũng dễ dàng được suy ra.

(Nội dung chương trình)
Sau khi hiểu rõ các mảng và biến được sử dụng cũng như cấu trúc chương trình nhiệm vụ tiếp theo là điền vào các chỗ trống trong chương trình dựa trên các dữ liệu mẫu được cho trong câu hỏi. Các ô trống được xử lý theo thứ tự thực hiện các lệnh lưu trong mảng trong dữ liệu mẫu. Phần tử đầu tiên của mảng là lệnh '{', vì thế chúng ta bắt đầu từ lệnh trong phần tử thứ 1, tương đương ô trống A.

Ô trống A:

Ô trống A chứa chỉ số của mảng "stack". Như được ghi chú tại dòng thứ 5 dành cho biến "spt" và từ các dòng sau ô trống, chỉ số của mảng "stack" phải là "spt". Nhưng vì giá trị khởi tạo của biến "spt" là -1, nên ta chưa thể sử dụng ngay biến này. Việc tăng chỉ số này lên là cần thiết trước khi kiểm tra giá trị của biến "opno", một thành phần của "stack". Do đó, ô trống A cần được điền (a) ++spt.

Hình 2 dưới đây mô tả rõ khi '{', giá trị mã lệnh của phần tử đầu tiên trong dữ liệu mẫu, được xử lý qua dòng 14.



Hình 2: Trạng thái khi '{' trong phần tử đầu tiên được xử lý qua dòng 14

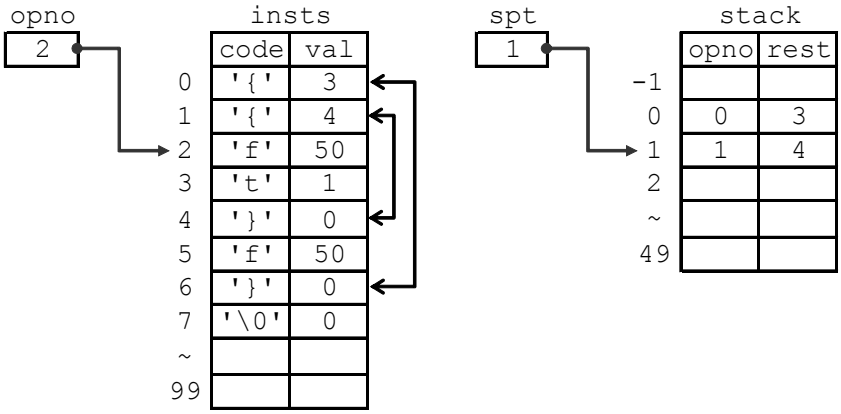
Sau Hình 2, trong dòng thứ 15, chương trình thoát khỏi câu lệnh rẽ nhánh, bỏ qua các trường hợp "switch" khác và chuyển tới dòng 38, chứa khoảng trắng G.

Ô trống G:

Khi chúng ta nghiên cứu cấu trúc chương trình, một điều nhận thấy rằng ô trống G luôn được thực hiện sau mỗi lần câu lệnh lưu trong mảng "insts" được xử lý. Dự đoán rằng giá trị của biến chỉ số "opno" sẽ được tăng lên tại đây. Ta sẽ khẳng định điều đó với dữ liệu mẫu.

Hình 2 là một ví dụ trạng thái ngay trước khi thực hiện lệnh trong ô trống G. Nếu không có gì được thực hiện tại đây và chương trình tiếp tục tới các dòng 10 và 11, giá trị của biến "opno" sẽ luôn là 0, do đó '{' tại câu lệnh đầu tiên, đã được xử lý, sẽ tiếp tục được xử lý. Nhưng, thực tế, '{' tại phần tử thứ 1 cần được xử lý. Nói cách khác, chỉ số "opno" cần phải được tăng lên trong ô trống G. Ta có thể khẳng định "opno++" là đáp án của ô trống G. Đáp án do đó là (c).

Hình 3 dưới đây chỉ ra trạng thái sau khi lệnh '{' tại phần tử thứ 1 được xử lý qua dòng 38 và "opno" tăng lên nhận giá trị là 2.



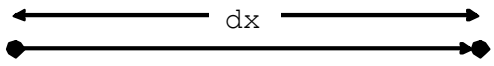
Hình 3: Lệnh '{' tại phần tử thứ 1 được xử lý qua dòng 38 và giá trị của "opno" được tăng lên nhận giá trị 2.

Nếu chương trình tiếp tục, nội dung lệnh tiếp theo "insts[opno].code" sẽ là 'f', chương trình sẽ thực hiện tới khối lệnh [3], tức là các ô trống C và D. Các dòng này gần như tương tự nhau. Ta sẽ điền C trước và từ đó suy ra giá trị của ô trống D.

Ô trống C:

Trong dòng 21, chứa ô trống C, giá trị của biểu thức sử dụng toán tử điều kiện "?" được gán cho biến "dx". Do do, một cách tự nhiên, ta sẽ xem xét vai trò của biến "dx" trước.

"dx" được sử dụng trong dòng 24 và 25. Trong dòng 24, nó được sử dụng làm tham số cho hàm "drawLine", làm nhiệm vụ vẽ một đường thẳng. Trong dòng 25, giá trị này được cộng thêm vào "mark.x". Bởi vì dòng 27 sử dụng hàm "paintMarker" để vẽ con trỏ, ta hiểu rằng việc này là để cập nhật giá trị tọa độ x của con trỏ. Do đó, ta suy ra được dx chính là khoảng cách giữa tọa độ x của con trỏ trước khi di chuyển và tọa độ x của nó sau khi di chuyển. Nói cách khác, biến dx là khoảng cách trục hoành khi di chuyển, như minh họa dưới đây.

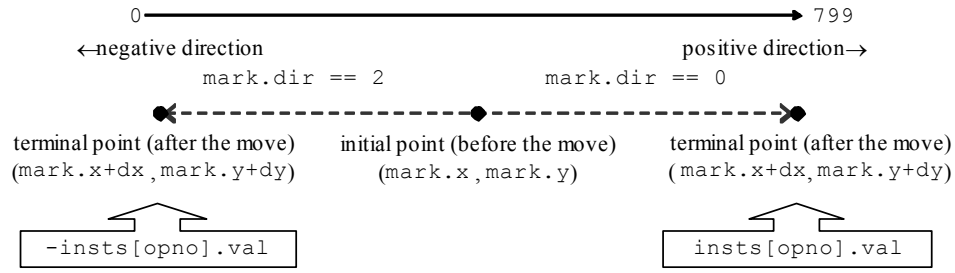


gốc (trước khi di chuyển) (mark.x , mark.y)

đích (sau khi di chuyển) (mark.x+dx , mark.y+dy)

Hình 4: Vai trò của biến "dx" (Phần 1)

Mặc dù vậy, dựa vào câu hỏi, khoảng cách di chuyển đươc lưu trong "insts[opno].val". Điều này có nghĩa là tọa độ x của điểm đích sẽ nhận giá trị "mark.x + insts[opno].val". Khi chúng ta tiếp tục xem xét, một điều nhận thấy là khi di chuyển về phía trái, ta cần trừ đi một lượng "insts[opno].val" từ "mark.x". Nói cách khác giá trị của biến "dx" khi đó sẽ có trị tuyệt đối bằng "insts[opno].val" nhưng mang dấu âm. Hình vẽ sau giải thích cụ thể điều này:



Hình 5: Vai trò của biến "dx" (Phần 2)

Sauk hi xem xét tất cả các vấn đề trên, ta sẽ tìm công thức cho chúng ta giá trị của biến "dx".

Toán tử điều kiện "?" được sử dụng để tìm giá trị của biến "dx". Toán tử này được sử dụng "công thức 1 ? công thức 2: công thức 3". Nếu công thức 1 nhận giá trị đúng, công thức 2 sẽ được thực hiện và giá trị của nó được trả lai; ngược lại nếu 1 sai, công thức được thực hiện sẽ là công thức 3. Ở đây, công thức 1 là "mark.dir %2 ==0", chỉ nhận giá trị đúng khi "mark.dir" nhận giá trị 0 (phải) hoặc 2 (trái). Do đó, công thức 2 được thực hiện và gán cho biến "dx" khi con trỏ di chuyển sang phải hoặc trái, công thức 3 được thực hiện khi con trỏ di chuyển lên hoặc xuống.

"dx" chứa khoảng cách trục hoành, vì thế khi di chuyển lên xuống, giá trị này bằng 0. Điều này thu hẹp các đáp án khả dĩ xuống, chỉ giữ lại các trường hợp mà công thức 3 là 0: (a)-(d) và (j). Từ những kết quả còn lại này, ta sẽ lựa chon ra đáp án có giá trị của công thức 2 là "+insts[opno].val" khi di chuyển sang phải (mark.dir nhận giá trị 0) và "-insts[opno].val" khi di chuyển sang trái (mark.dir nhận giá trị 2). Điều kiện này thỏa mãn với "(1 - mark.dir) * insts[opno].val : 0", tương đương "1 * insts[opno].val" khi "mark.dir" là 0 (phải) và "-1 * insts[opno].val" khi "mark.dir" là 2 (trái). Cuối cùng, (a) là kết quả của ô trống C.

Ta sẽ tổng kết lại quá trình suy luận để từ đó có thể sử dụng lại suy luận này với ô trống D.

Biến "dx" là khoảng cách trục hoành khi di chuyển, nhận giá trị dương "insts[opno].val" khi "mark.dir" là 0 (phải), và giá trị âm "insts[opno].val" khi "mark.dir" là 2 (trái), và giá trị 0 khi "mark.dir" là 1 hoặc 3 (lên hoặc xuống). Đáp án sau

cho ô trống C là hợp lý.

```
dx = ( mark.dir % 2 == 0 ) ? (1 – mark.dir) * insts[opno].val : 0
```

Ô trống D:

Ta sử dụng lại các suy luận đã dùng với ô trống C.

Biến "`dy`" là khoảng cách trục tung khi di chuyển. Khi "`mark.dir`" là 0 (phải) hoặc 2 (trái), biến này nhận giá trị 0. Nếu "`mark.dir`" là 1 (lên), nó nhận giá trị âm "`insts[opno].val`". Khi "`mark.dir`" là 3 (xuống), nó nhận giá trị dương "`insts[opno].val`".

Do đó công thức 2 trong biểu thức điều kiện phải là 0. Đáp án đúng cho D phải là các đáp án từ (e)~(i). Công thức 3 sẽ là "`-1 * insts[opno].val`" khi "`mark.dir`" là 1 (lên) và "`1 * insts[opno].val`" khi "`mark.dir`" là 3 (xuống). Cuối cùng, công thức sau là đáp án hợp lý.

```
dy = ( mark.dir % 2 = = 0 ) ? 0 : (mark.dir – 2 ) * insts[opno].val
```

Do đó, (h) là đáp án đúng cho D.

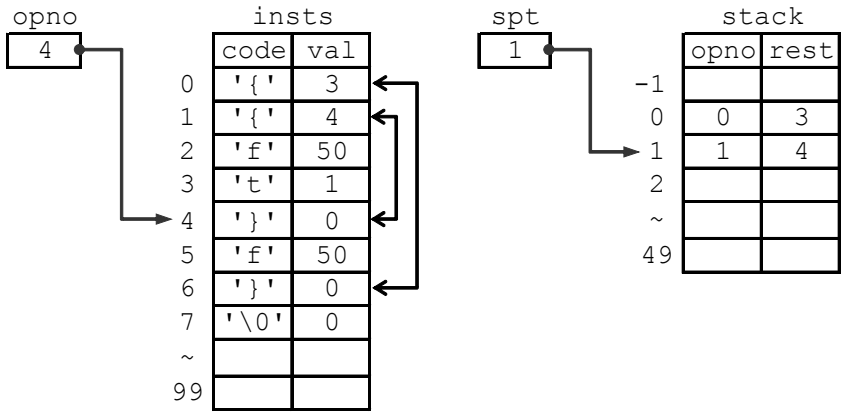
Sau khi '`f`' tại câu lệnh có chỉ số 2 được xử lý ở khối mã [3], dòng 38 tăng giá trị của biến chỉ số "`opno`", nhận giá trị 3. Vì vậy, mã lệnh "`insts[opno].code`" trở thành '`t`', khối mã [2] được thực hiện, chứa ô trống B.

Ô trống B:

Theo câu hỏi, nếu mã lệnh "`insts[opno].code`" là '`t`', hướng di chuyển của con trỏ sẽ lệch đi 90 độ × val ngược chiều kim đồng hồ. Nói cách khác, ô trống B tính toán hướng di chuyển mới của con trỏ sau khi gặp lệnh chuyển hướng. Giá trị chỉ hướng là các số nguyên từ 0~3, do đó ta cần chọn một công thức chỉ cho kết quả từ 0~3. Trong số các đáp án, công thức duy nhất thỏa mãn điều kiện này là `( mark.dir + insts[opno].val ) % 4`. Cuối cùng, (c) là đáp án đúng của B.

Vi dụ, ta thấy (g) không phải là đáp án đúng. Nếu "`mark.dir`" là 3 và "`insts[opno].val`" là 5, kết quả sẽ là 3 + 5 % 4 = 3 + 1 = 4, không phải là hướng có thể.

Trong hình 6, giải thích khi lệnh '`t`' của phần tử với chỉ số 3 được xử lý trong khối mã [2] và chỉ số "`opno`" được tăng lên, nhận giá trị 4 trong dòng 38.

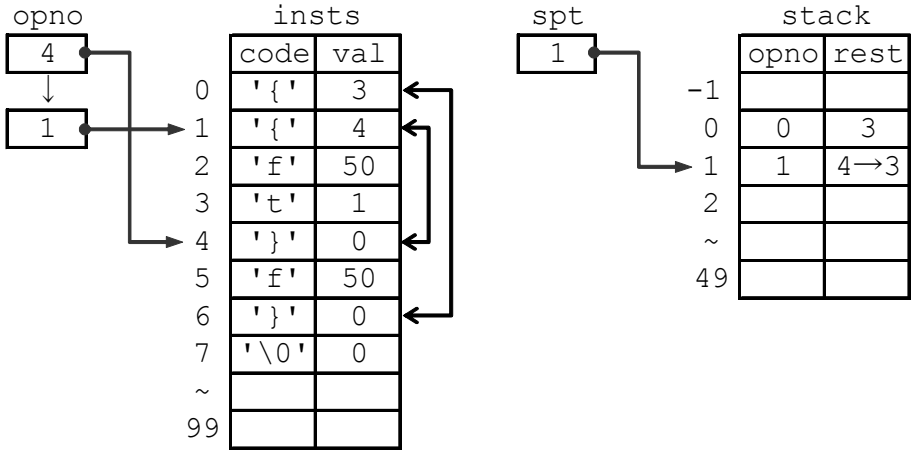


Hình 6: Chỉ số "opno" được tăng lên trong dòng 38 và nhận giá trị 4

Tới thời điểm này, mã lệnh "`insts[opno].code`" là '`}`', vì vậy chương trình sẽ thực hiện khối mã [4], chứa các ô trống E và F.

Ô trống E:

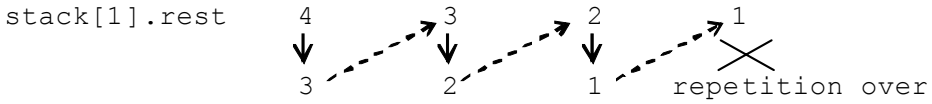
Khối mã [4], chứa ô trống E, được xử lý khi giá trị của "`insts[opno].code`" là '`}`', vì thế có thể dự đoán đây sẽ là quá trình xử lý việc lặp đoạn mã giới hạn bởi '`{`' và '`}`'. Điều này có thể khẳng định qua dòng 31 và 32. Ví dụ, trạng thái trong hình 6, nếu dòng 31 được thực hiện, giá trị của biến chỉ số "`opno`" thay đổi từ 4 thành 1, chỉ số của lệnh '`{`' cùng cặp với lệnh '`}`' hiện tại. Chỉ số "`opno`" được sử dụng để truy nhập mảng "`insts`"; nó luôn được tăng thêm 1 tại dòng 38 mỗi khi có một câu lệnh được thực hiện. Do đó, gán giá trị của biến "`opno`" một giá trị cụ thể sẽ chuyển việc thực hiện tiếp theo tới lệnh mang giá trị đó. Nói cách khác, các lệnh nằm trong "`{`" và "`}`" sẽ được thực hiện lại (lặp). Khi dòng 32 được xử lý, giá trị của thành phần "`rest`" (số lần lặp) trong "`stack`" sẽ được giảm xuống.



Hình 7: Trạng thái sau khi dòng 31 được thực hiện

Như thấy ở đây, dòng 31 và 32 được dành cho việc xử lý lặp, do đó câu lệnh "`if`" trong dòng 30, chứa ô trống E, phải là điều kiện cho việc lặp. Ta sẽ phải kiểm tra giá trị của "`stack[stp].rest`" để tìm ra điều kiện khi nào việc lặp được tiếp tục.

Giá trị ban đầu của "`stack[stp].rest`" là 4, và dòng 32 giảm giá trị này đi 1 mỗi lần được thực hiện. Như đã trình bày ở trên, sau khi dòng này được thực hiện, chương trình sẽ thực hiện việc lặp. Cụ thể hơn nếu có 4 lần lặp, giá trị này sẽ giảm từ 4 xuống 3 sau khi thực hiện xong chu trình đầu tiên và chu trình thứ 2 bắt đầu, giá trị này giảm xuống 2 khi chu trình thứ 3 bắt đầu và 1 đối với chu trình thứ 4. Do đó, ngay trước khi thực hiện lần lặp thứ 4, giá trị này là 1. Sau khi thực hiện xong chu trình này, việc lặp đã kết thúc; dòng thứ 34 sẽ được thực hiện.

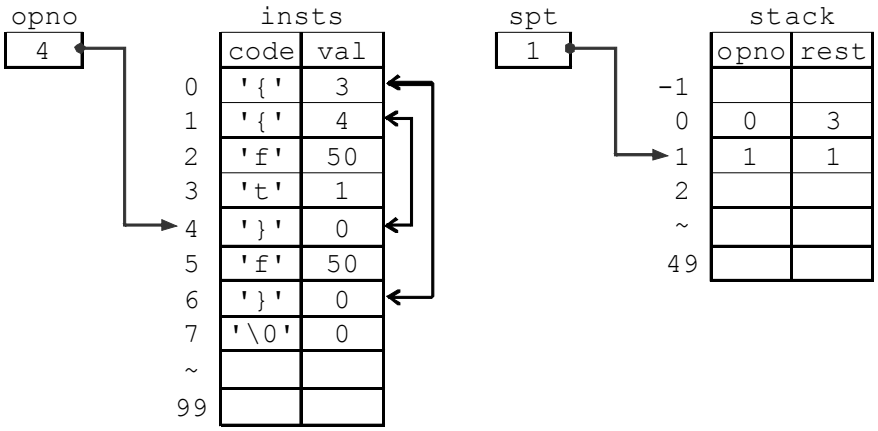


Hình 8

Tổng kết lại. Ta cần điền vào ô trống G một biểu thức điều kiện sao cho khi giá trị của "`stack[stp].rest`" lớn hơn 1, biểu thức này nhận giá trị đúng và dòng 31, 32 được thực hiện. Cuối cùng, đáp án (e) "`>1`" là đáp án đúng của ô trống E.

Ô trống F:

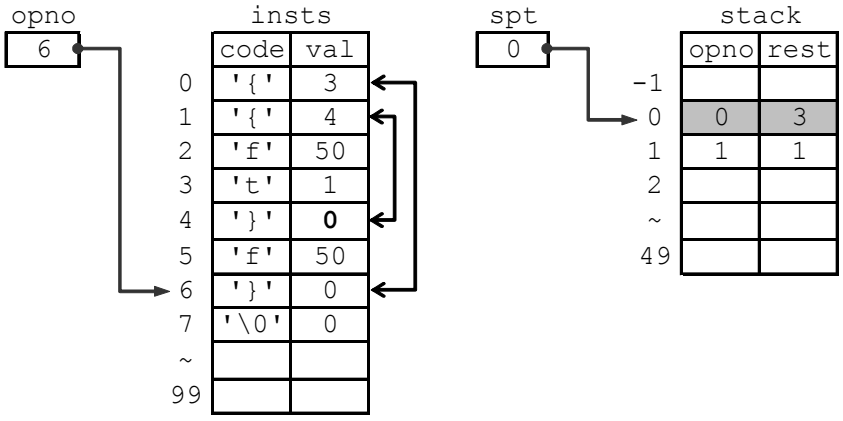
Được thực hiện khi mã lệnh "`insts[opno].code`" là '`}`' và "`stack[spt].rest`" nhận giá trị 1, khi vòng lặp kết thúc. Ví dụ, các lệnh nằm trong '`{`' tại phần tử thứ 1 và '`}`' tại phần tử thứ 4 được lặp 4 lần, khi ô trống F trong dòng 34 được thực hiện, mảng "`insts`" và "`stack`", cùng với các chỉ số của nó như sau.



Hình 9: Mảng "`insts`" và "`stack`" cùng với chỉ số tương ứng

Để tìm ra đáp án cho F, thử đặt ra vấn khi ta tiếp tục chương trình mà không thực hiện gì trong ô trống F

Giả sử không có gì được thực hiện từ trạng thái ở hình 9 tới quá trình tiếp theo. Đầu tiên, biến chỉ số "`opno`" được tăng từ 4 lên 5, mã lệnh "`insts[opno].code`" sẽ là '`f`'. Khối mã [3] sẽ được thực hiện. Không có gì đặc biệt xảy ra tại khối mã này. Sau khi hoàn thành khối mã [3], chỉ số "`opno`" tăng từ 5 lên 6 tại dòng 38. Lúc này, "`insts[opno].code`" là '`}`', chương trình vì thế sẽ chuyển tới thực hiện khối mã [4]. Lúc này, nếu giá trị của biến chỉ số "`spt`" của mảng "`stack`" vẫn là 1 thì vấn đề sẽ xảy ra. Nguyên nhân bởi vì vòng lặp giới hạn bởi phần từ thứ 1 và phần tử thứ 4 đã được thực hiện xong, do đó nó cần phải được loại bỏ khỏi stack. Nói cách khác, chỉ số "`spt`" cần phải có giá trị 0. Điều này được giải thích cụ thể như sau.



Hình 10: Chỉ số "`spt`"

Để tạo ra trạng thái như trong hình 10 khi chỉ số "`opno`" là 6, ta cần giảm giá trị của biến chỉ số "`spt`" sau khi những lệnh giới hạn bởi '`{`' tại lệnh thứ 1 và '`}`' tại lệnh số 4 đã được lặp đủ 4 lần (Hình 9). Cuối cùng, (f) "`spt--`" cần phải là kết quả của ô trống F. Nó không thực sự loại bỏ các giá trị ra khỏi stack mà chỉ tương đương với việc thay đổi chỉ số chỉ vị trí của đỉnh stack và khi một lênh '{' tiếp theo được xử lý, "`++spt`" sẽ trỏ vào đỉnh mới của stack.

Một phương pháp để điền vào F là tìm kiếm một vị trí thích hợp để giảm giá trị của chỉ số stack "`spt`", bởi vì chưa có câu lệnh nào giảm giá trị của nó. Thời điểm cần giảm chỉ số này là ngay khi "`stack[spt].rest`" hoàn thành vai trò của mình hay khi vòng lặp kết thúc. Do đó F là vị trí thích hợp để thực hiện điều này.

Câu hỏi 4 Lập trình Java

Q4. Đọc mô tả chương trình và chương trình Java dưới đây rồi trả lời Câu hỏi con.

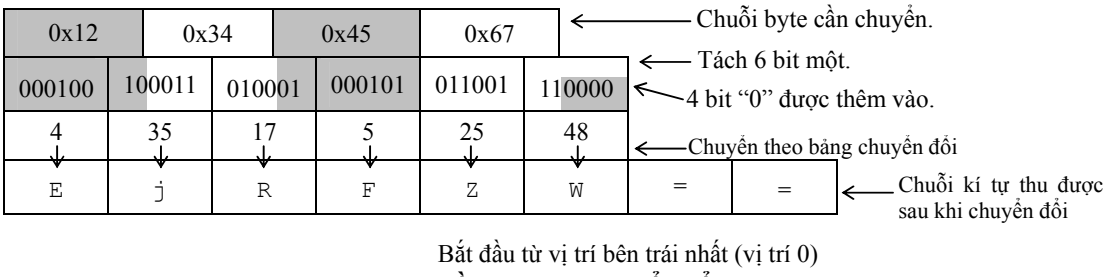
[Mô tả chương trình]

Chương trình bao gồm: a) lớp `Encoder`, có nhiệm vụ chuyển đổi chuỗi byte dữ liệu nhị phân theo một thuật toán nào đó và trả về kết quả là một chuỗi kí tự; b) một lớp chạy thử cho lớp `Encoder`. Thuật toán chuyển đổi như sau:

(1) Mỗi lần trích ra 6 bit từ đầu chuỗi byte và chuyển nó thành kí tự tương ứng theo bảng chuyển đổi. Nếu số lượng byte không phải là bội của 3 thì thêm 4 bit hoặc 2 bit "0" vào cuối chuỗi sao cho tổng số bit vẫn là bội của 6.

(2) Một số các kí tự "=" được thêm vào cuối chuỗi kí tự đã chuyển đổi sao cho tổng số kí tự là bội nhỏ nhất của 4 và lớn hơn hoặc bằng 4/3 lần số byte đã cho.

(3) Một ví dụ về quá trình chuyển đổi được thể hiện ở Hình 1 dưới đây



Hình 1. Ví dụ về quá trình chuyển đổi

(4) Lớp `Encoder` có 2 phương thức: `next` và `hasNext`. Phương thức `next` trả về lần lượt từng kí tự đã chuyển đổi theo thuật toán chuyển đổi. Nếu phương thức này được gọi khi không có kí tự nào trả về thì ném ra ngoại lệ `java.util.NoSuchElementException`.

Phương thức `hasNext` là `true` nếu trong lần gọi kế tiếp, phương thức `next` trả về một kí tự chứ không phát sinh ngoại lệ.

(5) Biến `CHARS` là một mảng 64 phần tử, đóng vai trò một bảng chuyển đổi số 6 bit sang một kí tự đơn lẻ. Biến `n` lưu giữ thông tin chỉ ra vị trí tiếp theo của kí tự được trả về trong lần gọi kế tiếp của phương thức `next` (vị trí của kí tự phía bên trái nhất là 0).

Biến `bin` lưu giữ chuỗi dữ liệu nhị phân cần biến đổi.

(6) Kết quả thực thi của lớp chạy thử (lớp `EncoderTest`) đối với chuỗi byte trong Hình 1 được chỉ ra trong Hình 2.

```
EjRFZw==
```

Hình 2. Kết quả chạy

[Chương trình]

```
public class Encoder {
    static final char[] CHARS = ("ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
    "abcdefghijklmnopqrstuvwxyz0123456789+/").toCharArray();
    private int n = 0;
    private byte[] bin;
    public Encoder(byte[] bin) {
        if (bin == null) this.bin = new byte[0];
        else this.bin = bin;
    }
    public boolean hasNext() {
        return n < [          A          ];
    }
    public char next() {
        char letter;
        int pos = (int)(n * 0.75);
        // Lấy được 2 byte dữ liệu gồm 6 bit dành cho việc chuyển đổi
        if (pos < bin.length) {
            int cell = bin[pos++] << 8;
            if (pos < bin.length) cell += bin[pos] & 255;
            // Lấy 6 bit cần chuyển đổi và chuyển đổi chúng thành ký tự tương ứng
            letter = CHARS[(cell >> (n + 3) % 4 * 2 + 4) & 63];
        } else {
            if ([          B          ])
                throw new java.util.NoSuchElementException();
            else letter = [          C          ];
        }
        n++;
        return letter;
    }
}

class EncoderTest {
    public static void main(String[] args) {
        byte[] bin = {0x12, 0x34, 0x45, 0x67};
        Encoder encoder = new Encoder(bin);
        while (encoder.hasNext()) {
            System.out.print(encoder.next());
        }
        System.out.println();
    }
}
```

Câu hỏi con

Từ các nhóm câu trả lời sau, chọn câu trả lời đúng để chèn vào các ô trống [ ] trong chương trình trên.

Nhóm câu trả lời cho A:

a) `(bin.length + 2) * 4 / 3`
b) `(bin.length + 2) / 3 * 4`
c) `bin.length * 4 / 3`
d) `bin.length / 3 * 4`

Nhóm câu trả lời cho B:

a) `!hasNext( )`
b) `hasNext( )`
c) `n < bin.length`
d) `n >= bin.length`

Nhóm câu trả lời cho C:

a) `' '`
b) `'='`
c) `CHARS[n]`
d) `next( )`

Đáp án câu 4

Chuyển đổi dữ liệu nhị phân

Đáp án

A – b, B – a, C – b

Giải thích

Đây là một chương trình đọc chuỗi byte dữ liệu nhị phân 6 bit một lần và chuyển đổi thành chuỗi kí tự tương ứng. Vì đây là biểu diễn dữ liệu 6 bit với 64 kiểu kí tự nên kiểu chuyển đổi này được gọi là "mã hóa BASE64". Vì dữ liệu nhị phân có thể được biểu diễn bởi các kí tự của bảng mã ASCII nên chúng được sử dụng để truyền và nhận dữ liệu nhị phân chủ yếu trong thư điện tử. Chương trình gồm một lớp `Encoder` không có bất kì chỗ nào khó hiểu về mặt cú pháp, nhưng có một vài khái niệm phức tạp, như là tính toán số lượng kí tự hay đọc chuỗi byte 6 bit một lần. Hơn nữa, chỉ có 3 ô trống nên mỗi ô trống trong câu hỏi có điểm nhiều hơn; vì thế phải rất cẩn thận để không mắc những lỗi do sơ suất.

[Mô tả chương trình]

Hằng "`CHARS`" được định nghĩa trong lớp `Encoder` là một bảng dùng cho chuyển đổi chuỗi 6 bit lấy từ dữ liệu nhị phân thành các kí tự. Một chuỗi kí tự trong Java là một tham chiếu tới một đối tượng `String`, nên có thể sử dụng một số phương thức như "`toCharArray`", như đã thấy trong chương trình.

Trong lớp `Encoder`, chương trình nhận chuỗi byte dữ liệu nhị phân với một phương thức khởi tạo (constructor) và trả về mỗi lần 1 kí tự bởi phương thức "`next`". Phương thức "`hasNext`" trả về giá trị kiểu `boolean`, thể hiện có kí tự tiếp theo hay không.

Đầu tiên, ta xem xét quá trình chuyển đổi dữ liệu nhị phân thành các kí tự, và được thực hiện bởi phương thức "`next`", mặc dù để trả lời được câu hỏi, có thể không nhất thiết phải hiểu nội dung được diễn giải ở đây.

Như trong đầu bài, biến "`n`" chứa vị trí kí tự được trả về tiếp theo ("0" cho kí tự đầu). Mảng dữ liệu nhị phân "`bin`" chứa 8 bit cho mỗi phần tử, chia thành các phần 6 bit, và chuyển đổi thành các kí tự đầu ra, nên số kí tự đầu ra sẽ không hoàn toàn tương ứng với số phần tử (chỉ số) của mảng "`bin`". Do đó, nhân "`n`" với 0.75 (= 6 / 8) sẽ thu được chỉ số phần tử trong mảng "`bin`" tương ứng với "`n`", và được lưu giữ trong biến "`pos`". Ở đây, khi gọi `(int)` là ta đã cắt bỏ phần thập phân của kết quả tính toán.

Vấn đề ở đây là trong dữ liệu 8 bit tương ứng với chỉ số thu được từ biến "`pos`", không phải lúc nào cũng bao gồm dữ liệu 6 bit cần thiết. Ví dụ, 6 bit tương ứng với kí tự đầu tiên rõ ràng là 6 bit đầu của phần tử có chỉ số bằng 0. Vậy dữ liệu tương ứng với kí tự tiếp theo (`n = 1`) thì sao? Đương nhiên, đó là 6 bit dữ liệu tiếp theo, nhưng trong số 8 bit của phần tử có chỉ số 0, chỉ còn thừa lại 2 bit. Do đó ta cần phải lấy tiếp 4 bit đầu của phần tử có chỉ số 1. Nói cách khác, khi `n = 1`, giá trị của biến "`pos`" là phần nguyên (cắt bỏ phần thập phân) của $1 \times 0.75 = 0.75$, là 0. Do đó, giá trị của "`pos`" thu được từ biến "`n`" thực ra là chỉ số của phần tử chứa bit đầu của chuỗi 6 bit đang xét, và 6 bit đang xét đó có thể không hoàn toàn nằm trong dữ liệu biểu diễn bởi chỉ số. Bởi vậy, nếu giá

trị của "pos" nhỏ hơn "bin.length", đoạn xử lí sau được sử dụng để lấy 16 bit dữ liệu cần thiết vào biến "cell" từ mảng "bin".

```
int cell = bin[pos++] << 8;
if (pos < bin.length)  cell += bin[pos] & 255;
```

Ở đây, chú ý đến phần xử lí "bin[pos++] << 8", bởi "bin" là biến mảng kiểu byte (8-bit) và bình thường nếu dịch trái 8 bit đối với dữ liệu 8 bit sẽ cho kết quả = 0. Tuy nhiên, với toán tử dịch bit (shift) trong Java, dù biến thuộc kiểu byte hay kiểu short (16-bit), mọi biến trước hết được chuyển đổi sang kiểu int (32-bit) và sau đó phép xử lí mới được thực hiện (nếu kết quả được lưu vào không gian nhỏ hơn 32 bit thì các bit cao bị cắt bỏ). Nếu tiếp sau vẫn còn dữ liệu (pos < bin.length), thì 8 bit của phần dữ liệu tiếp được thêm vào (cell += bin[pos] & 255) sao cho dữ liệu 16 bit có thể được lưu trong biến"cell". Phép xử lí bit "AND" với 255 (1111 1111 trong hệ nhị phân) được thực hiện trước phép cộng đơn giản chỉ để cộng với 8 bit thấp nhất của "bin [pos]" (trong trường hợp này, "bin[pos]" chỉ có 8 bit nên điều này không cần thiết, nhưng thông thường kiểu phép toán bit này hay được sử dụng). Sau khi đã có dữ liệu 16-bit tương ứng với vị trí kí tự được chỉ ra bởi biến "n" được lưu trong biến "cell", ta chuyển sang phần tiếp theo:

```
letter = CHARS[(cell >> (n + 3) % 4 * 2 + 4) & 63];
```

Chú ý phần "(cell >> (n + 3) % 4 * 2 + 4) & 63" ở dòng trên. Bằng sự tính toán này ta thu được 6 bit sẽ được chuyển đổi, và trở thành chỉ số của mảng "CHARS" đồng thời tìm ra được kí tự tương ứng.

Xem xét công thức "(cell >> (n + 3) % 4 * 2 + 4) & 63" một lần nữa. Thông thường, phép toán được chia làm 4 trường hợp tùy thuộc vào giá trị của n, nhưng ở đây 4 trường hợp được kết hợp trở thành một. Ví dụ, xem xét dữ liệu nhị phân dưới dây. Chú ý rằng phần "& 63" là để lấy về 6 bit thấp nhất bằng cách lấy "&" với $63_{10}=111111_2$.

	[0]	[1]	[2]	[3]	
Dữ liệu của mảng "bin"	0x12	0x34	0x45	0x67	……
(số nhị phân)	00010010	00110100	01000101	01100111	……

Ba khối dữ liệu 8 bit cho ta chính xác 4 khối dữ liệu 6 bit, nên 3 khối dữ liệu 8 bit đầu tiên trong bin[0], bin[1], và bin[2]

pos =	0	1	2
	00010010	00110100	01000101

cho 4 khối dữ liệu 6 bit (kí tự):

n =	0	1	2	3
	000100	100011	010001	000101

[1] Khi n=0

"cell" chứa chuỗi bit 00010010 00110100 là kết quả ghép nối của bin[0] và bin[1]. Để thu được 6 bit trong hộp, ta dịch phải chuỗi trên đi 10 bit và lấy 6 bit thấp nhất.

(0 + 3) % 4 * 2 + 4 = 3 * 2 + 4 = 10

[2] Khi n=1

"cell" chứa chuỗi bit 0001001010 0011 0100 là kết quả ghép nối của bin[0] và bin[1]. Để

thu được 6 bit trong hộp, ta dịch phải chuỗi trên đi 4 bit và lấy 6 bit thấp nhất.

(1 + 3) % 4 * 2 + 4 = 0 * 2 + 4 = 4

[3] Khi `n=2`

Lúc này `pos = 1` nên "`cell`" chứa chuỗi bit 0011**0100 01**000101 là kết quả ghép nối của `bin[1]` và `bin[2]`. Để thu được 6 bit, ta dịch phải chuỗi trên đi 6 bit và lấy 6 bit thấp nhất

(2 + 3) % 4 * 2 + 4 = 1 * 2 + 4 = 6

[4] Khi `n=3`

Vì `pos = 2` nên "`cell`" chứa chuỗi bit 01**000101** 01100111 là kết quả ghép nối của `bin[2]` và `bin[3]`. Để thu được 6 bit, ta dịch phải chuỗi trên đi 8 bit và lấy 6 bit thấp nhất.

(3 + 3) % 4 * 2 + 4 = 2 * 2 + 4 = 8

Với `n>=4`, 4 trường hợp xử lí trên (`n=0~3([1]~[4])`) được lặp lại, vì thế điều kiện `n=0~3` có thể thay thế bởi `n % 4 = 0~3`. Như vậy, có thể tóm tắt về số bit biến "`cell`" cần dịch phải như sau:

`n % 4`	Số bit biến "`cell`" dịch phải
0	10
1	4
2	6
3	8

Biến đổi sang dạng (+4) giống trong công thức, ta có:

`n % 4`	Số bit biến "`cell`" dịch phải
0	6 + 4
1	0 + 4
2	2 + 4
3	4 + 4

Phân tích thành thừa số 2 cho phần đầu trong biểu thức, ta có:

`n % 4`	Số bit biến "`cell`" dịch phải
0	3 * 2 + 4
1	0 * 2 + 4
2	1 * 2 + 4
3	2 * 2 + 4

Do đó, tổng số bit biến "`cell`" cần dịch có thể tính được nhờ công thức "`(n + 3) % 4 * 2 + 4`". Khi biến "`cell`" được dịch phải một số bit thích hợp, ta thực hiện phép toán bit AND giữa kết quả thu được với 63. Bằng cách đó, ta có thể thu được 6 bit cần thiết.

[Câu hỏi con]

Ô trống A

Xem xét điều kiện để "`hasNext`" trả về true. Câu hỏi đưa ra mô tả sau: "Một số các kí tự "=" được thêm vào cuối chuỗi kí tự đã chuyển đổi sao cho tổng số kí tự là bội nhỏ nhất của 4 và lớn hơn hoặc bằng 4/3 lần số byte đã cho". Do đó, trong khi điều này còn đúng, các kí tự "=" tiếp tục được đưa ra thậm chí cả khi không còn dữ liệu nhị phân nào để chuyển đổi thành kí tự. Sau đó, lớp "`EncodeTest`" xuất ra các kí tự trong khi "`hasNext`" là true, vì thế ngay khi số kí tự thu được bởi phương thức "`next`" nhỏ hơn "bội nhỏ nhất của 4 và lớn hơn hoặc bằng 4/3 lần số byte đã cho", "`hasNext`" phải trả về true. Nói cách khác, biến "`n`" xuất hiện trước ô trống A biểu diễn số kí tự đã thu được bởi phương thức "`next`". Do đó, ta cần một công thức tính "bội nhỏ nhất của 4 và lớn hơn hoặc bằng 4/3 lần số byte đã cho" cho ô trống A. Bạn phải cẩn thận, trong Java, phép chia một số nguyên bởi một số nguyên, phần thập phân bị cắt bỏ, chỉ đưa ra kết quả cuối cùng là số nguyên. Từ đó có thể tự tìm câu trả lời. Câu trả lời đúng là (b) "`(bin.length + 2) /3 * 4`".

Lí do thêm 2 vào "`bin.length`" trước khi chia cho 3 là vì thương chính xác bằng `bin.length / 3` nếu "`bin.length`" là bội của 3 và thương bằng "`bin.length / 3 + 1`" trong các trường hợp khác. Nhân thương này với 4 cho ta chính xác " bội nhỏ nhất của 4 và lớn hơn hoặc bằng 4/3 lần số byte đã cho".

Ở các lựa chọn khác, (a) và (c) tiến hành nhân 4 trước khi chia cho 3, nên kết quả không thể là bội của 4. (d) sai vì kết quả nhỏ hơn 4/3 lần số byte đã cho.

Ô trống B

Ở đây ta cần tìm điều kiện để khi gọi phương thức "`next`", ngoại lệ `java.util.NoSuchElementException` được ném ra. Liên quan tới ngoại lệ là mục (4) của [Mô tả chương trình] trong phần câu hỏi. Ngoại lệ tương ứng được phát sinh "nếu phương thức này được gọi mà không có kí tự nào trả về…". Khi dữ liệu nhị phân được chuyển đổi (`pos < bin.length`), phương thức "`next`" chuyển đổi dữ liệu thành kí tự và trả về kí tự thu được. Ngay cả khi không thể chuyển đổi dữ liệu, nó phải tiếp tục xuất ra kí tự "=" cho đến khi đạt được số kí tự yêu cầu vì trong phần câu hỏi có viết: " Một số các kí tự "=" được thêm vào cuối chuỗi kí tự đã chuyển đổi sao cho tổng số kí tự là bội nhỏ nhất của 4 và lớn hơn hoặc bằng 4/3 lần số byte đã cho". Xem xét trong nhóm câu trả lời, không có câu nào thực hiện tính toán như thế. Tuy nhiên, với phần chú giải cho ô trống A ở trên, "`hasNext`" trả về một giá trị kiểu `boolean` cho biết đã đạt được tổng số kí tự yêu cầu hay chưa bằng cách tính toán tổng số kí tự. Ta sẽ sử dụng điều này. Nói cách khác, ta sử dụng thực tế là có các kí tự được trả về khi mà `hasNext` còn đúng. Ô trống B là điều kiện để ném ra ngoại lệ, do đó ta lấy phủ định của "`hasNext`"; câu trả lời đúng là (a) "`!hasNext()`".

Ô trống C

Như đã giải thích cho phần ô trống A và B, phương thức "`next`" tiếp tục xuất ra các kí tự "=" cho đến khi đạt được số kí tự yêu cầu, dù sau khi dữ liệu byte đã được chuyển đổi đã hết. Ô trống C chứa kí tự đầu ra cho mục đích đó. Chính vì thế, câu trả lời đúng là (b) '='.

Câu hỏi 5 Chương trình Java

Q4. Đọc mô tả chương trình và chương trình Java dưới đây rồi trả lời câu hỏi con.

[Mô tả chương trình]

Giao diện `CharIterator` định nghĩa một thao tác trích rút lần lượt các kí tự (`char`) từ thể hiện của nó, độc lập với cấu trúc dữ liệu. Trong `CharIterator`, các phương thức sau được định nghĩa:

`public char next()`

Nếu kí tự tiếp theo tồn tại thì trả về ký tự đó. Nếu không tồn tại, ngoại lệ `java.util.NoSuchElementException` sẽ được ném ra. Ví dụ như nếu `CharIterator` có *n* kí tự, thì kí tự đầu tiên được trả về cho lần gọi đầu tiên, và kí tự thứ 2 được trả về cho lần gọi thứ 2. Bằng cách này, các kí tự lần lượt được trả về cho đến khi thực hiện lần gọi thứ n. Với lần gọi thứ *n*+1 hoặc sau đó, ngoại lệ `NoSuchElementException` sẽ được ném ra.

Chú ý rằng, `NoSuchElementException` là lớp con của `java.lang.RuntimeException`.

`public boolean hasNext()`

Nếu kí tự tiếp theo tồn tại thì trả về `true`. Ngược lại, trả về `false`.

Lớp `CharIteratorFactory` định nghĩa một phương thức trả về `CharIterator` phù hợp với kiểu dữ liệu của đối số.

Trong `CharIteratorFactory`, các phương thức sau được định nghĩa:

`public static CharIterator getCharIterator(String data)`

Trả về `CharIterator`, trích rút lần lượt các kí tự từ đối số kiểu `String`. Nếu đối số truyền vào là `null` thì ngoại lệ `NullPointerException` được ném ra.

`public static CharIterator getCharIterator(char[][] data)`

Trả về `CharIterator`, trích rút lần lượt các kí tự từ đối số là một mảng của mảng (mảng kí tự 2 chiều) chứa các phần tử kiểu `char`. Nếu đối số truyền vào là `null` thì ngoại lệ `NullPointerException` được ném ra. Các kí tự được trích rút theo thứ tự tăng dần của giá trị chỉ số của mảng kí tự và mảng của nó. Ví dụ, với biến m, có kiểu `char[][]`, các thành phần `m[i][j]` của nó được trích rút theo thứ tự tăng dần của `i`, và với `i` như nhau, thì theo thứ tự tăng dần của `j`.

Lớp `CharIteratorTest` là một chương trình chạy thử các phương thức được định nghĩa trong `CharIteratorFactory`. Kết quả thực thi của phương thức `main` được thể hiện ở hình dưới:

```
'H' '1' '6'
'2' '0' '0' '4'
```

Kết quả thực thi của CharIteratorTest.main

[Chương trình 1]

```
public interface CharIterator {
    public boolean hasNext();
    public char next();
}
```

[Chương trình 2]

```
import java.util.NoSuchElementException;

public class CharIteratorFactory {
    public static CharIterator getCharIterator(String data) {
        if (data == null)
            throw new NullPointerException();
        // Tạo và trả về một thể hiện của CharIterator
        // trả về lần lượt các kí tự từ dữ liệu String.
        return [          A          ];
    }

    public static CharIterator getCharIterator(char[][] data) {
        if (data == null)
            throw new NullPointerException();
        // Tạo và trả về một thể hiện của CharIterator
        // trả về lần lượt các kí tự từ một mảng kí tự 2 chiều.
        return [          B          ];
    }
}
```

```
class StringCharIterator implements CharIterator {
    private String data;
    private int index = 0;

    StringCharIterator(String data) {
        this.data = data;
    }
    // Kiểm tra kí tự tiếp theo có tồn tại trong data hay không.
    public boolean hasNext() {
        return [          C          ];
    }

    public char next() {
        // Nếu kí tự tiếp theo không tồn tại, ném ra ngoại lệ NoSuchElementException.
        if (index >= data.length())
            throw new NoSuchElementException();
        // Trả về kí tự tiếp theo cho data và cập nhật giá trị chỉ số.
        return [          D          ];
    }
}


class Char2DArrayCharIterator implements CharIterator {
    private char[][] data;
    private int index1 = 0, index2 = 0;

    Char2DArrayCharIterator(char[][] data) {
        this.data = data;
    }
    public boolean hasNext() {
        // Nếu phần tử của data[index1][index2] tồn tại, trả về true.
        // Nếu không, tìm phần tử tiếp theo.
        // Nếu phần tử tiếp theo không tồn tại, trả về false.
        for (; index1 < data.length; index1++) {
            if (data[index1] != null
                    && index2 < data[index1].length) {
                return true;
            }
            [          E          ];
        }
        return false;
    }
    public char next() {
        // Gọi phương thức hasNext để kiểm tra phần tử tiếp theo. Nếu tồn tại,
        // phần tử đó được trả về, và cập nhật giá trị chỉ số.
        // Nếu không có phần tử nào tồn tại, ngoại lệ NoSuchElementException được ném ra.
        if (hasNext()) {
            return [          F          ];
        }
        throw new NoSuchElementException();
    }
}
```

[Chương trình 3]

```
public class CharIteratorTest {
    public static void main(String[] args) {
        CharIterator itr =
            CharIteratorFactory.getCharIterator("H16");
        printIterator(itr);

        itr = CharIteratorFactory.getCharIterator(
                new char[][] {   { '2' },
                                 { '0' },
                                 null,
                                 { '0', '4' }});
        printIterator(itr);
    }
    private static void printIterator(CharIterator itr) {
        while (itr.hasNext()) {
            System.out.print("'" + itr.next() + "' ");
        }
        System.out.println();
    }
}
```

Câu hỏi con

Từ các nhóm câu trả lời sau, chọn câu trả lời đúng để chèn vào các ô trống [ ] trong chương trình trên.

Nhóm câu trả lời cho A và B:

a) `getCharIterator((char[][]) data)`
b) `getCharIterator((String) data)`
c) `new Char2DArrayCharIterator()`
d) `new Char2DArrayCharIterator(data)`
e) `new StringCharIterator()`
f) `new StringCharIterator(data)`

Nhóm câu trả lời cho C:

a) `index < data.length()`
b) `index <= data.length()`
c) `index >= data.length()`
d) `index++ < data.length()`
e) `index++ <= data.length()`
f) `index++ >= data.length()`

Nhóm câu trả lời cho D:

a) `data.charAt(++index)`
b) `data.charAt(--index)`
c) `data.charAt(index + 1)`
d) `data.charAt(index++)`
e) `data.charAt(index--)`
f) `data.charAt(index)`

Nhóm câu trả lời cho E:

a) `index1 = 0`
b) `index1 = data.length`
c) `index1 = index2`
d) `index2 = 0`
e) `index2 = data.length`
f) `index2 = index1`

Nhóm câu trả lời cho F:

a) `data[index1++][index2++]`
b) `data[index1++][index2]`
c) `data[index1][++index2]`
d) `data[index1][--index2]`
e) `data[index1][index2 + 1]`
f) `data[index1][index2++]`

Đáp án câu 5

Chương trình liệt kê các kí tự

Đáp án

A – f, B – d, C – a, D – d, E – d, F – f

Giải thích

Đây là một chương trình nhận dữ liệu của một mảng 2 chiều chứa các xâu kí tự (kiểu `String`) hoặc các kí tự, trích rút từng kí tự một sau đó hiển thị chúng. Chương trình chứa phương thức "`hasNext`" và "`next`" giúp ta trả lời nhanh chóng và dễ dàng. Cần chú ý đến sự thay đổi chỉ số của mảng trong lớp `Char2DArrayCharIterator` sử dụng mảng 2 chiều, nhưng đây là điểm duy nhất có thể gây khó khăn về mặt thuật toán. Nhìn chung, đây là một câu hỏi dễ.

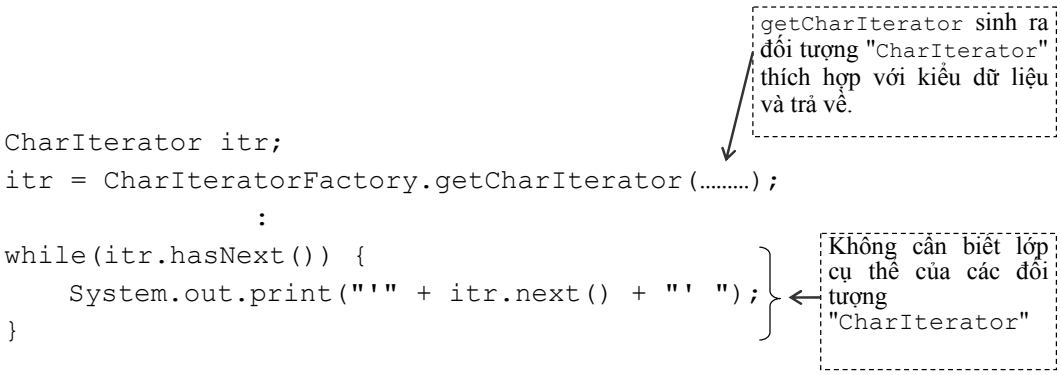
[Mô tả chương trình]

Đầu tiên, trong [Chương trình 1], giao diện "`CharIterator`" được định nghĩa. Sau đó, trong [Chương trình 2], lớp "`CharIteratorFactory`", "`StringCharIterator`", và "`Char2DArrayCharIterator`" được định nghĩa.

"`CharIteratorFactory`" là một lớp được khai báo `public`, nhưng 2 lớp còn lại không phải là lớp `public`. Những lớp không phải `public` chỉ có thể được sử dụng trong cùng gói (package); chúng không thể được sử dụng trực tiếp từ các gói khác (chương trình này không được khai báo như một `package`, nhưng kể cả trong trường hợp này nó vẫn tạo thành một gói gọi là "gói không được đặt tên"). Tuy nhiên, các đối tượng được tạo ra bởi các lớp có thể được gọi bởi các phương thức từ các gói khác thông qua các biến đối tượng kiểu `CharIterator`. Trong trường hợp này, các phương thức được ghi đè trong lớp `StringCharIterator` hoặc lớp `Char2DArrayCharIterator` sẽ được gọi. Nói cách khác, kể cả khi không khai báo các lớp cụ thể như là 2 lớp "`public`", các hàm định nghĩa trong các lớp này vẫn có thể được sử dụng thông qua lớp `CharIteratorFactory` (sinh đối tượng) và giao diện `CharIterator` (sử dụng phương thức), nên các lớp này cố ý được khai báo không public.

Trong lập trình Java, ta thường giấu một vài lớp nào đó có chủ ý trong khi sử dụng các chức năng của các lớp đó một cách gián tiếp. Điều này làm yếu đi sự ghép nối giữa các lớp (ghép nối lỏng) và nâng cao khả năng bảo trì và mở rộng của chương trình.

Trong đoạn chương trình bên dưới, phương thức `static` "`getCharIterator`" được định nghĩa chồng (overloaded) trong lớp "`CharIteratorFactory`" giúp các đối tượng thích hợp với kiểu dữ liệu của chúng được sinh ra và trả về. Hơn nữa, chương trình được thiết kế sao cho lớp "`CharIteratorTest`", sử dụng các đối tượng này trong [Chương trình 3], có thể hoạt động mà không cần biết đến sự tồn tại của các lớp như `StringCharIterator` và `Char2DArrayCharIterator`.



Ô trống A và B

Đây là các câu hỏi về các giá trị trả về của 2 phương thức `static` "`getCharIterator`" được định nghĩa chồng trong lớp "`CharIteratorFactory`". Giá trị trả về cho mỗi phương thức thuộc kiểu "`CharIterator`", nên một đối tượng của lớp thực thi giao diện "`CharIterator`" phải được tạo ra và trả về (đối tượng không thể được tạo ra từ giao diện "`CharIterator`"). Đối với lớp của các đối tượng cần sinh ra, ta có thể nghĩ đến 2 lớp: "`StringCharIterator`" và "`Char2DArrayCharIterator`". Xét phương thức khởi tạo của mỗi lớp, ta thấy phương thức khởi tạo của lớp "`StringCharIterator`" nhận đối tượng "`String`" là đối số trong khi đó phương thức khởi tạo của lớp "`Char2DArrayCharIterator`" nhận mảng 2 chiều kiểu `char`.

Với những lí do trên, lựa chọn duy nhất cho việc tạo và trả về các đối tượng một cách chính xác là (f) "`new StringCharIterator(data)`" cho ô trống A và (d) "`new Char2DArrayCharIterator(data)`" cho ô trống B. Biến "`data`" trong ô trống A thuộc kiểu `String`, và biến "`data`" trong ô trống B là một mảng 2 chiều kiểu `char` (`char[][]`), nên đó là các câu trả lời đúng do kiểu dữ liệu.

Với các lựa chọn trả lời khác, vì "`String`" không thể chuyển thành "`char[][]`" và "`char[][]`" không thể chuyển thành "`String`", nên (a) hoặc (b) đều không chính xác do "`getCharIterator`" không thể gọi phương thức khác nữa. Thêm vào đó, thật vô nghĩa khi gọi đệ qui chính chúng (gọi liên tục không ngừng, cuối cùng đưa ra kết quả lỗi). (c) và (e) sai bởi vì một phương thức khởi tạo không nhận đối số không được định nghĩa trong lớp "`StringCharIterator`" hoặc "`Char2DArrayCharIterator`".

Ô trống C

Đây là câu hỏi liên quan đến phương thức "`hasNext`" trong lớp "`StringCharIterator`". Trong lớp này, các đối tượng `String` được nhận thông qua một phương thức khởi tạo và được lưu trữ trong trường `private` "`data`", từ đó các kí tự được lấy và trả về lần lượt từng kí tự một. Có một trường khác "`index`" trong lớp "`StringCharIterator`", và giá trị khởi tạo của nó bằng 0. Ta có thể đoán đây chính là vị trí kí tự hiện tại trong xâu kí tự (vị trí đầu của xâu kí tự là 0, và vị trí cuối là "`data.length( ) - 1`"). Như đã được viết trong [Mô tả chương trình], phương thức "`hasNext`" là một phương thức trả về "sự tồn tại của kí tự tiếp theo" bằng giá trị `boolean (true/false)`. Cho nên, ta cần thiết lập sao cho trả về "`true`" nếu `index` nhỏ hơn độ dài của chuỗi kí tự "`data.length()`" và trả về "`false`" nếu `index` lớn hơn hoặc bằng `data.length()`. Do đó, câu trả lời là (a) "`index < data.length()`".

Thêm thông tin cho bạn, lựa chọn (d) "`index++ < data.length()`" cũng có chứa bước tăng, và cũng phụ thuộc vào câu trả lời cho ô trống D, câu trả lời này có thể giúp chương trình "`CharIteratorTest`" thực hiện một cách đúng đắn (nếu ô trống D là "`data.charAt(index - 1)`" – không có trong nhóm câu trả lời). Tuy nhiên, kể cả trong trường hợp đó, "`index`" không được tăng trong "`hasNext`". Giá trị của "`index`" chỉ nên được tăng sau khi đã thu được kí tự, và "`hasNext`" phải trả về kết quả như nhau dù nó gọi bao nhiêu lần cho tới khi phương thức "`next`" được gọi. Thêm vào đó, khi "`hasNext`" được gọi nhiều lần, không nên để cho bất kì một kí tự nào bị bỏ sót và không được trích rút bởi "`next`". Điều này không liên quan trực tiếp đến việc trả lời các câu hỏi tiếp theo, nhưng trong lập trình thực tế khi tạo "`Iterator`", bạn nên nhớ điều đó.

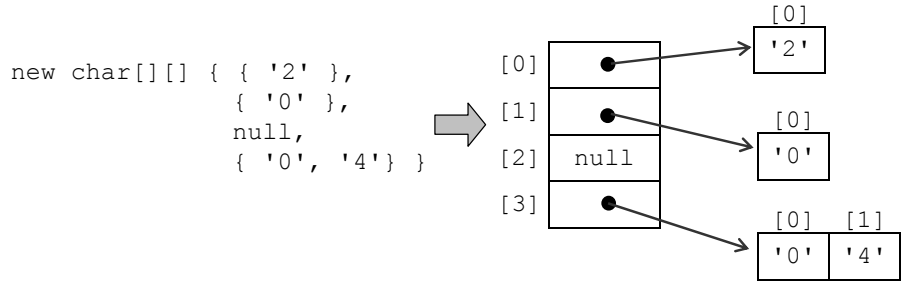
Ô trống D

Câu hỏi này là về phương thức "`next`" trong lớp "`StringCharIterator`". Nói đến phương thức này, [Mô tả chương trình] có viết "Nếu kí tự tiếp theo tồn tại thì nó được trả về. Nếu không, …" nên chức năng của phương thức là trích rút và trả về lần lượt mỗi lần 1 kí tự từ đối số là chuỗi kí tự. Đối với điều này, như đã thấy trong các giải thích cho ô trống C ở trên, ô trống D phải chứa một quá trình nhận kí tự tại vị trí được chỉ ra bởi "`index`" từ chuỗi kí tự "`data`" và sau đó là lệnh tăng giá trị của "`index`". Do đó, câu trả lời đúng là (d) "`data.charAt(index++)`".

Ô trống E

Câu hỏi này là về phương thức "`hasNext`" trong lớp "`Char2DArrayCharIterator`". Trong lớp này, một phương thức khởi tạo được sử dụng để thu nhận một mảng kí tự 2 chiều, sau đó lưu trữ trong trường `private` "`data`". Hai trường `private` khác, "`index1`" và "`index2`", được định nghĩa với giá trị ban đầu là 0. Từ những chú thích bên trong phương thức "`hasNext`" và từ nhóm trả lời cho ô trống F, có thể dễ dàng nhận ra chúng là phần tử mang chỉ số 1 và 2 của mảng 2 chiều "`data`".

Một mảng 2 chiều trong Java có cấu trúc như một "mảng của mảng". Mảng 2 chiều nhận được từ phương thức "`main`" của lớp "`CharIteratorTest`" được cấu trúc như sau:



Trong mảng này, `data[2]` là `null` trong khi `data[0]`, `data[1]`, và `data[3]` chứa một tham chiếu tới mỗi miền của mảng. Kích cỡ của các miền này, theo thứ tự, `data[0].length` (=1), `data[1].length` (=1), và `data[3].length` (=2). Hơn nữa, nhờ "`data.length`" ta có thể thu được kích thước của "`data`" như một "mảng của mảng" (=4). Điểm mấu chốt của câu hỏi ở đây là làm cách nào để sử dụng nhóm các mảng có kích thước khác nhau này và đôi khi là `null`.

Trong phương thức "`hasNext`", câu lệnh "`for`" được sử dụng. Nhìn qua có thể khiến bạn nghĩ rằng một quá trình lặp đi lặp lại được thực hiện, nhưng chương trình có câu lệnh "`if`" tiếp

theo.

```
if(data[index1] != null
  && index2 < data[index1].length) {
  return true;
}
```

Chương trình được thiết kế để thoát ngay khỏi phương thức này với "`return true;`" khi thỏa mãn điều kiện "`data[index1] != null && index2 < data[index1].length`". Do đó, vòng lặp "`for`" thực tế có thể thực hiện được một lần trong khi thực thi và chương trình có thể thường kết thúc mà không có thêm bất kì vòng lặp nào. Khi điều kiện chưa thỏa mãn, nói cách khác, nếu "`data[index1]`" là `null` hoặc "`index2`" đã bằng "`data[index1].length`", ô trống E được thực thi, "`index1`" được tăng, và vòng lặp tiếp theo của "`for`" được thực hiện (chú ý ở đây, "`index1++`" trong câu lệnh "`for`" được thực hiện trước khi chương trình kiểm tra điều kiện "`index1 < data.length`"). Điều này cho thấy vòng lặp "`for`" có mục đích chuyển tới mảng tiếp theo khi mảng trong câu hỏi ("`data[index1]`") là `null` hoặc khi duyệt đến phần tử cuối cùng của mảng. Để đến mảng tiếp theo, "`index1`" cần tăng và "`index2`" trở về 0. Ở đây chưa có câu lệnh nào gán "`index2`" bằng 0, đây là điều cần đưa vào ô trống E. Do đó, câu trả lời chính xác cho ô trống E là (d) "`index2 = 0`".

Lúc này, khi điều kiện "`index1 < data.length`" của vòng lặp "`for`" không còn, không còn thêm mảng nào, nên phương thức "`hasNext`" trả về "`false`" nhờ câu lệnh cuối "`return false;`".

Ô trống F

Câu hỏi này về phương thức "`next`" trong lớp "`Char2DArrayCharIterator`". Giống như phương thức "`next`" trong lớp "`StringCharIterator`", phương thức này trả về các kí tự lưu trong mảng 2 chiều "`data`" từng kí tự mỗi lần và phải tăng chỉ số mảng để chuẩn bị cho lời gọi kết tiếp. Biến được tăng là "`index2`", nên câu trả lời đúng là (f) "`data[index1][index2++]`".

Trong phương thức này, "`index2`" được tăng, và có thể "`index2`" đi đến cuối mảng mà không có quá trình xử lí nào được thực hiện. Tuy nhiên, như đã thấy trong chú thích cho ô trống E, khi "`index2`" tới cuối mảng, có một xử lí trong phương thức "`hasNext`". Trong phương thức "`next`", phương thức "`hasNext`" được thực thi trước khi trả về một kí tự, nên khi "`index2`" tới cuối của một mảng, kí tự được trả về sau khi "`index2`" về 0 và "`index1`" được tăng.

Câu hỏi 6 Chương trình Java

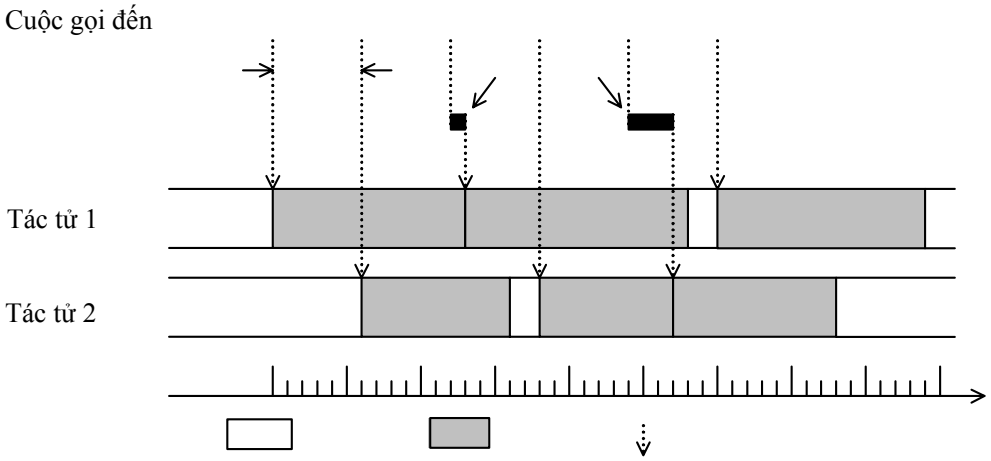
Q6. Đọc mô tả chương trình và chương trình Java dưới đây, rồi trả lời câu hỏi con.

[Mô tả chương trình]

Chương trình này là một trình giả lập mô phỏng hoạt động của tổng đài (call center) của Công ty A. Công ty A sử dụng trình giả lập này để ước lượng thời gian cần thiết cho một người dùng kết nối tới một nhân viên trực tổng đài (operator) khi gọi đến tổng đài. Một cuộc gọi tới tổng đài luôn chỉ được trả lời bởi một nhân viên trực tổng đài.

Giả định cho trình giả lập này chỉ cần một nhân viên trực tổng đài rỗi thì không người gọi nào phải đợi. Ví dụ, giả sử có 6 cuộc gọi tới tổng đài mỗi cuộc gọi cách nhau 60 giây và có 2 nhân viên trực tổng đài đang rỗi. Thời gian của mỗi cuộc gọi là 130 giây, 100 giây, 150 giây, 90 giây, 110 giây, và 140 giây. Nghĩa là thời gian đợi của người thứ 3 là 10 giây, người thứ 5 là 30 giây, và những người khác là 0 giây. (Xem hình 1)

Chú ý rằng trình giả lập mô phỏng 1 giây trong thế giới thực là 0.1 giây trong trình giả lập.



Hình 1: Thời gian đợi của người dùng

Chương trình bao gồm các lớp sau:

(1) `CallCenter`

Lớp này biểu diễn tổng đài điện thoại. Nó thực hiện mô phỏng sau khi định rõ số nhân viên trực tổng đài, danh sách thời gian gọi, và các khoảng thời gian tạo ra cuộc gọi trong đối số của phương thức khởi tạo. Nó bao gồm các phương thức và các lớp nội bộ sau:

Phone call 1 2 3

60 giây

`public static void main(String[] args)`

Phương thức này khởi động trình giả lập để chạy thử.

`Call answer()`

Phương thức này được gọi bởi một nhân viên trực tổng đài để trả lời một cuộc gọi. Nó trả về một đối tượng `Call` cho nhân viên trực tổng đài. Phương thức này khiến cho tiến trình nhân viên trực tổng đài đợi đến khi một cuộc gọi có thể được chỉ định cho nhân viên trực tổng đài. Nếu vẫn không có cuộc gọi được chỉ định sau khi tất cả các cuộc gọi tạo ra bởi lớp này đều đã được chỉ định, nó trả về `null`.

`Operator`

Lớp tiến trình này giả lập nhân viên trực tổng đài. Mỗi nhân viên trực tổng đài được xử lí trong một tiến trình riêng. Tiến trình này trả lời một cuộc gọi tới tổng đài và nói chuyện. Nó lặp lại quá trình này cho đến khi tất cả các cuộc gọi khởi tạo bởi `CallCenter` đều được chỉ định.

(2) `Call`

Lớp này mô tả một cuộc gọi từ người gọi. Thời gian nói chuyện được xác định trong đối số của phương thức khởi tạo. Lớp này bao gồm các phương thức sau.

`public void talk()`

Phương thức này giả lập trạng thái trong quá trình cuộc nói chuyện giữa một nhân viên trực tổng đài và người gọi. Phương thức này hiển thị thời gian chờ để người gọi được chỉ định tới nhân viên trực tổng đài tính theo giây (làm tròn), và sau đó dừng tiến trình nhân viên trực tổng đài chỉ trong thời gian xác định là thời gian nói chuyện.

Hình 2 chỉ ra kết quả chương trình khi ví dụ của Hình 1 được giả lập.
Chú ý (s) biểu diễn giây.

```
0(s)
0(s)
10(s)
0(s)
30(s)
0(s)
```

Hình 2 Kết quả giả lập

`java.util.Vector` trong chương trình là một lớp mô tả một mảng có số phần tử thay đổi, và lớp này bao gồm các phương thức sau:

`public boolean add(Object obj)`

Nó thêm phần tử `obj` vào cuối mảng.

`public Object remove(int index)`

Nó xóa phần tử của mảng tại vị trí xác định bởi `index` và trả về phần tử bị xóa. Vị trí của phần tử đầu tiên là 0. Các phần tử sau `index` được dịch lên phía trước.

`public boolean isEmpty()`

Nếu không có phần tử nào, phương thức này trả về `true`. Ngược lại, nó trả về `false`.

[Chương trình 1]

```
import java.util.Vector;

public class CallCenter {
    private final Vector waitingList = new Vector();
    private boolean running; // true khi cuộc gọi được khởi tạo

    public static void main(String[] args) {
        int op = 2;       // Số nhân viên trực tổng đài
        // Thời gian nói chuyện (giây)
        long[] duration = {130, 100, 150, 90, 110, 140};
        long interval = 60;  // Khoảng thời gian khởi tạo cuộc gọi (giây)
        new CallCenter(op, duration, interval);
    }

    public CallCenter(int op, long[] duration, long interval) {
        running = true;
        // Tạo ra tiến trình nhân viên trực tổng đài và khởi động nó
        for (int i = 0; i < op; i++) new Operator().start();
        long nextCallTime = System.currentTimeMillis();
        for (int i = 0; i < duration.length; i++) {
            // Khởi tạo một cuộc gọi và thêm vào danh sách
            synchronized (waitingList) {
                waitingList.add(new Call(duration[i]));
                waitingList.notify();
            }
            // Đợi đến khi cuộc gọi tiếp theo được tạo ra
            nextCallTime += interval * 100;  // Thao tác nhanh gấp 10 lần
            long sleeping = [          A          ];
            try {
                if (sleeping > 0) Thread.sleep(sleeping);
            } catch (InterruptedException ie) {}
        }
        // Kết thúc mọi tiến trình nhân viên trực tổng đài
        running = false;
        [          B          ] {
            waitingList.notifyAll();
        }
    }
```

```
    Call answer() {
        synchronized (waitingList) {
            while (waitingList.isEmpty() [C]) {
                try {
                    [D];
                } catch (InterruptedException ie) {}
            }
            if (waitingList.isEmpty()) return null;
            return (Call)waitingList.remove(0);
        }
    }

    class Operator  extends Thread {
        public void run() {
            Call call;
            [E] call.talk();
        }
    }
}
```

[Chương trình 2]

```
public class Call {
    private final long start, duration;
    public Call(long duration) {
        this.duration = duration;
        start = System.currentTimeMillis();
    }
    public void talk() {
        long elapsed = System.currentTimeMillis() - start;
        // Hiển thị thời gian trôi qua sau khi làm tròn
        System.out.println([F] + "(s)");
        try {
            Thread. sleep(duration * 100);      // Thao tác nhanh gấp 10 lần
        } catch (InterruptedException ie) {}
    }
}
```

Câu hỏi con 1

Trong các nhóm câu trả lời dưới đây, hãy chọn đáp án đúng để chèn vào các ô trống ☐ trong chương trình trên.

Nhóm câu trả lời cho A:

a) `nextCallTime`
b) `nextCallTime - duration[i] * 100`
c) `nextCallTime - System.currentTimeMillis()`
d) `System.currentTimeMillis() - nextCallTime`

Nhóm câu trả lời cho B:

a) `for (int i = 0; i < op; i++)`
b) `if (waitingList != null)`
c) `synchronized (this)`
d) `synchronized (waitingList)`

Nhóm câu trả lời cho C:

a) `&& !running`
b) `&& running`
c) `== !running`
d) `|| running`

Nhóm câu trả lời cho D:

a) `notify()`
b) `wait()`
c) `waitingList.notify()`
d) `waitingList.wait()`

Nhóm câu trả lời cho E:

a) `if ((call = answer()) != null)`
b) `if (answer() != null)`
c) `while ((call = answer()) != null)`
d) `while (answer() != null)`

Nhóm câu trả lời cho F:

a) `(elapsed + 50) / 100`
b) `(elapsed + 500) / 100`
c) `(elapsed - 50) / 100`
d) `(elapsed - 500) / 100`

Câu hỏi con 2

Một thể hiện `CallCenter` được tạo ra như dưới đây, trong nhóm câu trả lời sau, hãy chọn số lượng nhân viên trực tổng đài đang bận sau 8 giây trôi qua trong trình giả lập (80 giây trong thời gian thực).

```
new CallCenter(3, new long[]{70, 90, 100, 110}, 30);
```

Nhóm câu trả lời:

a) 0 b) 1 c) 2 d) 3

Đáp án câu 6

Mô phỏng tổng đài

Đáp án đúng

[Câu hỏi con 1] A – c, B – d, C – b, D – d, E – c, F – a

[Câu hỏi con 2] c

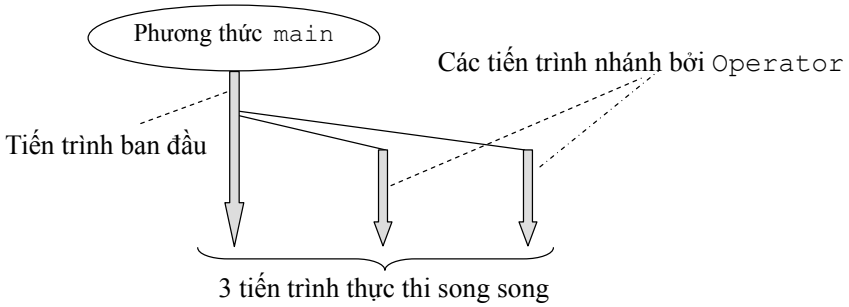
Giải thích

Đây là một chương trình giả lập cho phép ước lượng thời gian chờ từ khi có một cuộc gọi tới tổng đài cho đến lúc người gọi được kết nối với một nhân viên trực tổng đài. Đây là lần đầu tiên một chương trình sử dụng chức năng đa tiến trình của Java xuất hiện trong kì thi. Câu hỏi không chỉ hỏi về mặt cú pháp của chức năng tiến trình của Java mà còn yêu cầu kiến thức sâu về cách sắp xếp hài hòa giữa các tiến trình. Về mặt cú pháp, các phương thức như "`wait`" và "`notify`" cũng như các lớp nội bộ được sử dụng. Nhìn chung, đây là một câu hỏi khó.

[Mô tả chương trình]

Đầu tiên ta sẽ thiết lập một cái nhìn tổng quan về [Mô tả chương trình] và các nội dung của chương trình. Chương trình bao gồm lớp `CallCenter`, lớp `Operator` (là một lớp nội bộ của lớp `CallCenter`), và lớp `Call`. Vì lớp `Operator` kế thừa lớp `Thread`, các tiến trình mới có thể được khởi động bằng cách tạo ra các thể hiện và thực thi phương thức "`start`".

Khi chương trình được thực thi từ phương thức "`main`" trong lớp "`CallCenter`", phương thức "`main`" tạo ra các đối tượng "`CallCenter`". Những đối tượng này không cần lưu trong các biến riêng biệt để sử dụng, nhưng thông qua quá trình tạo ra các đối tượng, phương thức khởi tạo cho lớp "`CallCenter`" được gọi đến. Phương thức khởi tạo cho lớp "`CallCenter`" này ban đầu tạo ra các thể hiện trong lớp "`Operator`" và gọi phương thức "`start`". Nó bắt đầu một tiến trình mới. Số các tiến trình được tạo ra được quyết định bởi đối số "`op`" của phương thức khởi tạo. Ở đây, vì `op = 2` cộng với tiến trình ban đầu, như vậy có tổng cộng 3 tiến trình được thực thi song song.



Một tiến trình được tạo ra bởi đối tượng `Operator` tự gọi phương thức "`run`" của chính nó. Khi xem xét phương thức "`run`" của lớp `Operator`, ta chỉ thấy ô trống E và lời gọi phương thức "`talk`" của đối tượng "`Call`" là "`call.talk`". Tuy nhiên, từ nhóm câu trả lời cho E, ta thấy ô trống E chứa lời gọi phương thức "`answer`" của lớp "`CallCenter`". Lớp "`Operator`" là một lớp nội bộ của lớp "`CallCenter`" nên các đối tượng " `Operator`" có thể tham chiếu và sử dụng

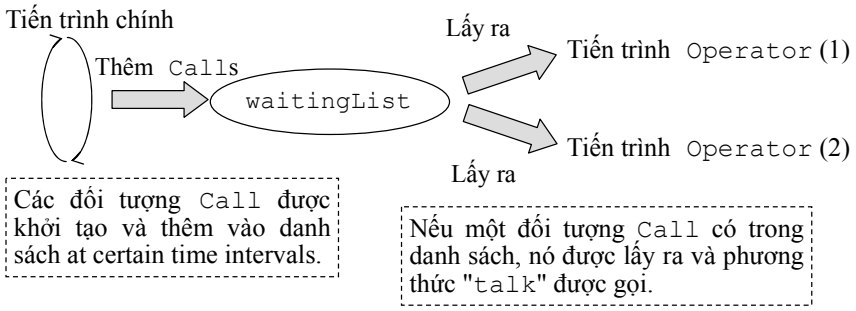
các biến thể hiện `private` và các phương thức thể hiện của các đối tượng `CallCenter`.

Xem xét nội dung phương thức "`answer`" của lớp `CallCenter`, ta chú ý điều sau: Phương thức này kiểm tra "`waitingList`" có rỗng hay không bằng cách sử dụng vòng lặp "`while`". Tại cuối phương thức, câu lệnh "`return (Call)waitingList.remove(0);`" được sử dụng để xóa bỏ đối tượng "`Call`" đầu tiên lưu vào "`waitingList`" khỏi danh sách này; cùng lúc đó, đối tượng này được trả về. Phần này chứa ô trống C và D, nên khó có thể hiểu được chi tiết của phương thức, nhưng ít nhất có thể thấy, nếu danh sách chứa một đối tượng "`Call`" thì đối tượng ấy được trả về.

Ở đây, một lần nữa phải nhớ rằng sau khi tiến trình khác tách biệt với tiến trình `main` (tiến trình bắt đầu từ phương thức "main") được tạo ra, mọi xử lí sau phương thức "`run`" trong lớp "`Operator`" đều được thực thi.

Tiến trình `main` được xử lí liên tục sau khi các đối tượng "`Operator`" được tạo ra và phương thức "`start`" được gọi, tức là, kể cả khi phương thức "`run`" được gọi và thực hiện phần xử lí của nó. Theo dấu các quá trình tiếp theo phương thức khởi tạo của lớp "`CallCenter`", ta thấy các đối tượng "`Call`" được tạo ra liên tục bởi vòng lặp "`for`" và được thêm vào "`waitingList`". Gần đến khi kết thúc lặp, quá trình thực thi bị hoãn lại bởi phương thức "`Thread.sleep`". Dựa trên đề bài và phần chú thích "Đợi đến khi lời gọi tiếp được tạo ra" trong chương trình, lời gọi phương thức "`Thread.sleep`" xuất hiện để tạo ra các cuộc gọi (các đối tượng "`Call`") theo các khoảng thời gian nào đó.

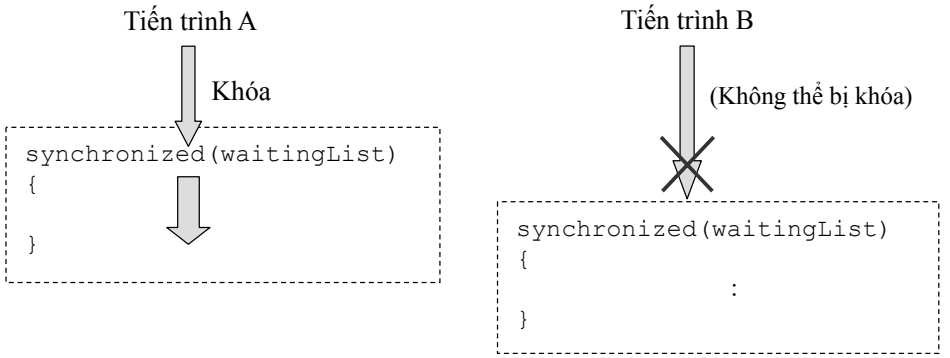
Từ những nội dung trên, ta hiểu rằng trong chương trình này tiến trình chính thêm các đối tượng "`Call`" vào "`waitingList`" theo các khoảng thời gian nào đó trong khi các tiến trình của "`Operator`" nhận các đối tượng "`Call`" từ "`waitingList`" bằng phương thức "`answer`" và gọi phương thức "`talk`".



Khi bạn hiểu được hoạt động của chương trình như thế, chỉ còn một điều bạn cần xem xét, đó là làm cách nào kiểm soát được các xung đột có thể xảy ra giữa các tiến trình. Trong chương trình, mỗi tiến trình đều tham chiếu đến "`waitingList`", nên bạn cần xem xét đến vấn đề xung đột giữa các tiến trình đối với "`waitingList`" này. Ví dụ, giả sử một tiến trình kiểm tra danh sách là không rỗng và chuẩn bị lấy ra một phần tử. Nhưng chỉ trong một khoảng thời gian ngắn giữa thời điểm kiểm tra rằng danh sách không rỗng và thời điểm sắp lấy ra phần tử, một tiến trình khác có thể xóa bỏ phần tử, khiến cho danh sách rỗng. Điều này dẫn tới lỗi. Trong lập trình đa luồng, một vấn đề có thể nảy sinh khi nhiều các tiến trình cố gắng giành quyền truy cập tới cùng một thông tin.

Để giải quyết vấn đề này, trong Java, bạn có thể sử dụng cách điều khiển đồng bộ giữa các tiến trình bằng cách sử dụng một "câu lệnh `synchronized`". Một câu lệnh `synchronized` khóa một đối tượng nào đó có hiệu lực từ thời điểm tiến trình đi vào khối `synchronized` cho đến khi rời khỏi khối. Trong khi một tiến trình có một đối tượng bị khóa, tiến trình khác không thể khóa cùng đối tượng đó bởi câu lệnh `synchronized`; tiến trình này chuyển sang trạng thái tạm nghỉ

cho đến khi tiến trình đầu giải phóng khóa trên đối tượng.



Trong khi một tiến trình có "waitingList" bị khóa với câu lệnh "synchronized", tiến trình khác không thể vào "synchronized"

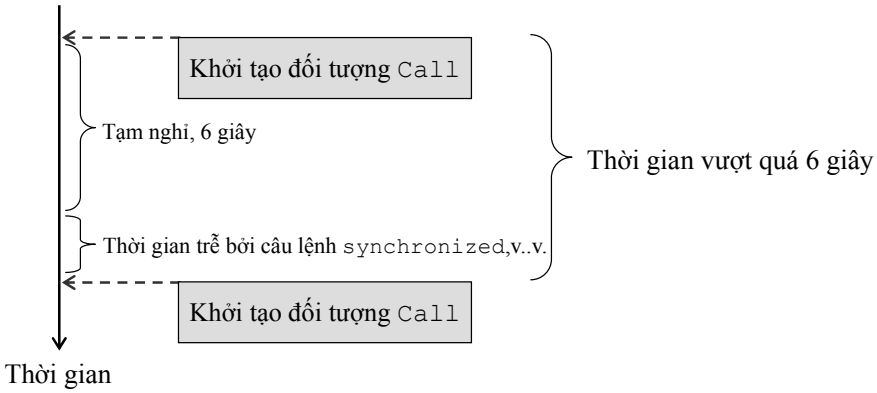
Ví dụ, với phương thức "answer" trong lớp "CallCenter", kiểm soát đồng bộ phải được thực hiện bằng một câu lệnh "synchronized" trong trường hợp trạng thái của "waitingList" được xác nhận bởi phương thức "isEmpty" và sau đó được cập nhật bởi một phương thức ví dụ như "remove".

[Câu hỏi con 1]

Ô trống A:

Bạn được hỏi nên gán giá trị nào cho biến "sleeping" (kiểu long). Vì giá trị này được chỉ định là đối số cho "thread.sleep", nó chính là thời gian nghỉ cho tiến trình. Như đã giải thích trong [Mô tả chương trình], vòng lặp "for" ở đây có thể là một vòng lặp cho tiến trình main để sinh ra các cuộc gọi (các đối tượng "Call) theo các khoảng thời gian nào đó. Ví dụ, các cuộc gọi cách nhau 60 giây có thể được giả lập nếu cứ 6 giây ta tạo ra các đối tượng Call bởi trình giả lập (một khoảng thời gian thực 1 giây được giả lập bằng 0.1 giây trong trình giả lập). Đối số của phương thức "sleep" được xác định bằng mili-giây, nên đối số là 6000 cho khoảng thời gian 6 giây. "interval * 100" có trong chương trình là cho đối số này. Ở đây, biến "interval" là một đối số của phương thức khởi tạo và có giá trị 60. 100 nghĩa là "100 mili-giây" (bình thường, 1 giây = 1000 mili-giây, nhưng 1 giây được giả lập bởi 0.1 giây [= 100 mili-giây], số nhân là 100). Tuy nhiên, biểu thức "interval * 100" này được dùng trong câu lệnh "nextCallTime += interval * 100" như một lượng thêm vào "nextCallTime". Điều này không tìm thấy trong các lựa chọn của nhóm câu trả lời.

Làm thế nào để hiểu được điều này? Đây là một vấn đề, nhưng để ý rằng đó có thể là thời gian đợi gây ra bởi câu lệnh "synchronized" ở đầu của vòng lặp "for" và sẽ mất thời gian để tạo ra các đối tượng Call và thêm chúng vào "waitingList". Giả sử rằng 6000 (= 6 giây) được đưa vào làm đối số của phương thức "sleep". Nếu câu lệnh "synchronized" kích hoạt điều khiển đồng bộ, do đó sinh ra thời gian chờ, sẽ mất một khoảng thời gian nhiều hơn 6 giây từ khi tạo ra một đối tượng "Call" đến khi tạo ra đối tượng "Call" tiếp theo. Nếu quá trình này lặp đi lặp lại nhiều lần nhiều lần, sai số sẽ tích lại, cuối cùng đưa ra các thời điểm tạo ra đối tượng Call không đúng.



Để tránh điều này, chương trình đưa ra một biến gọi là "nextCallTime", dùng để chỉ ra thời điểm khi một đối tượng "Call" tiếp theo được tạo ra. Khi đối tượng "Call" đầu tiên được tạo ra, "nextCallTime" được khởi tạo bằng thời điểm hiện tại (phương thức "System.currentTimeMillis" lấy về thời gian hiện tại theo mili-giây), và thời gian cho việc tạo ra đối tượng tiếp theo được tính toán tại 6 giây sau thời điểm hiện tại đó. Khi sinh các đối tượng "Call" cũng có thể mất thêm thời gian, nên thời gian nghỉ bởi phương thức "sleep" là khác nhau giữa thời điểm khi một đối tượng được sinh tiếp theo và thời gian hiện tại. Vì thế, đáp án cho ô trống A là (c), "nextCallTime – System.currentMillis()".

Ô trống B:

Trong khối có chứa ô trống B, ta đọc thấy "waitingList.notifyAll();". Phương thức "notifyAll" là một trong những phương thức được định nghĩa trong lớp "Object". Nó thông báo tới tất cả tiến trình được đặt trong trạng thái tạm nghỉ bởi phương thức "wait" để lại tiếp tục tiến trình. Giống như "synchronized", "wait" và "notifyAll" được sử dụng khi các tiến trình được đồng bộ với nhau. "notifyAll" (or "notify") thông báo tới tất cả các tiến trình đang đợi bởi phương thức "wait" trong một điều kiện nào đó để chúng lại tiếp tục hoạt động của mình; đó là cách "notifyAll" thực hiện đồng bộ hóa. Giống một "synchronized", các phương thức này được dùng để khóa và giải phóng các đối tượng nào đó. Nói cách khác, chúng gọi "wait" cho các đối tượng trong "waitingList", và để thông báo những tiến trình tạm nghỉ, chúng lại gọi phương thức "notifyAll" cho các đối tượng trong "waitingList". Ngoài ra, những phương thức này phải được sử dụng trong khi "waitingList" bị khóa bởi câu lệnh "synchronized". Ngược lại, xảy ra một ngoại lệ "IllegalMonitorStateException". Vì thế, một câu lệnh "synchronized" cho "waitingList" phải được đưa vào ô trống B, và đáp án là (d).

Ô trống C và D:

Những ô trống này liên quan đến phương thức "answer". Như đã giải thích trong [Mô tả chương trình], phương thức này được gọi bởi tiến trình "Operator" và, nếu một đối tượng Call tồn tại trong "waitingList" thì nó trả về đối tượng đó. Ô trống C là một câu lệnh được thêm vào cho điều kiện cho vòng lặp "while" này. Ô trống D là quá trình được thực hiện khi điều kiện thỏa mãn. Trước hết xét D. Đây là cái cần thực thi lặp đi lặp lại nhiều lần khi "waitingList" là rỗng, nên nó có thể là nơi mà các tiến trình được đặt ở trạng thái tạm nghỉ bởi phương thức "wait". Điều đó giải thích cho phần sau trong phương thức khởi tạo của lớp "CallCenter".

```
                :
synchronized(waitingList) {
   waitingList.add(new Call(duration[i]));
   waitingList.notify();
}
                :
```

Nói cách khác, tiến trình "Operator" gọi phương thức "answer" để kiểm tra "waitingList" có rỗng hay không, và nếu có thì đợi cho đến khi tiến trình main thêm một đối tượng "Call" vào "waitingList". Trong lúc đó, sau khi thêm một đối tượng vào "waitingList", tiến trình main gọi "notify" để thông báo rằng một đối tượng đã được thêm và đánh thức (tiếp tục) các tiến trình đang ở trạng thái tạm nghỉ. Phương thức "notify" khác với "notifyAll" ở chỗ chỉ một tiến trình được thông báo kể cả khi có nhiều tiến trình. Phương thức "notify" được gọi cho các đối tượng "waitingList", nên phương thức "wait" cũng cần được gọi cho "waitingList". Do đó, đáp án đúng cho ô trống D là (d).

Trở lại với ô trống C. Từ các lựa chọn trong nhóm câu trả lời, bạn biết rằng điều kiện này bao gồm một biến tên là "running". Biến "running" đầu tiên được khởi tạo giá trị "true" khi bắt đầu phương thức khởi tạo của "CallCenter" và chuyển sang "false" sau khi vòng lặp "for" kết thúc cùng với chú thích "Dừng tất cả các tiến trình Operator". Ở đây, phương thức "notifyAll" cũng được gọi. Do đó, ý nghĩa của biến "running" dường như cho thấy trong khi "running" là "true" thì tiến trình main hoạt động, và các đối tượng "Call" được tạo ra và thêm vào "waitingList". Khi tiến trình main chấm dứt, biến "running" trở thành "false". Xem xét đến nhiều tiến trình được đặt trong trạng thái tạm nghỉ bởi phương thức "wait". Mọi xử lí dường như có lí nếu ta dùng "notifyAll" thay vì "notify" để mọi tiến trình đều được thông báo. Trong khi tiến trình main đang chạy, biến "running" là "true", nên ô trống C là (b) "&& running". Sau đó, khi "running" thay đổi thành "false" bởi phương thức main, vòng lặp "while" này sẽ kết thúc. Nếu lúc đó "waitingList" rỗng thì phương thức "answer" trả về giá trị "null", vì thế toàn bộ hoạt động này thỏa mãn mô tả của phương thức "answer" của câu hỏi.

Ô trống E:

Đây là trong phương thức "run" trong lớp "Operator". Ở đây, phương thức "answer" được gọi, và giá trị trả về của nó phải được lưu trong biến "call". Nếu không thì một ngoại lệ xảy ra ở câu lệnh ngay sau "call.talk();". Do đó, các lựa chọn có thể thu hẹp lại còn (a) và (c). Nếu chọn (a), sau khi thực thi một lần, phương thức "run" kết thúc, và tiến trình cũng kết thúc tại thời điểm đó. Tuy nhiên, xét bản chất của câu hỏi, tiến trình này phải tiếp tục nhận các đối tượng "Call" lặp đi lặp lại nhiều lần trong khi "answer" tiếp tục trả về các đối tượng "Call". Do đó, đáp án là (c). Nếu phương thức "answer" trả về "null" thì nó báo rằng tiến trình main đã ngừng tạo ra các đối tượng "Call"; sau đó, tiến trình "Operator" cũng kết thúc.

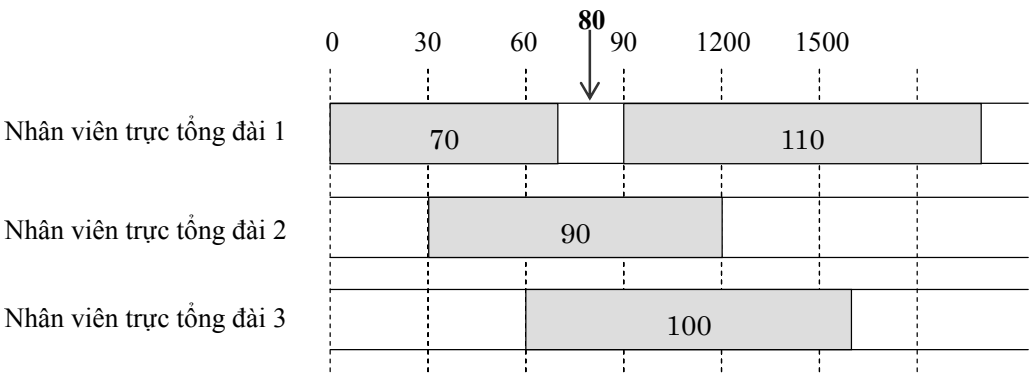
Ô trống F:

Điều được hiển thị ở đây là thời gian đợi của của người dùng để được được phân công tới một nhân viên trực tổng đài. Dòng lệnh trước ô trống F lưu kết quả của phép trừ thời điểm hiện tại cho

"`start`" trong biến "`elapsed`". Mặt khác, biến "`start`" được thiết lập bằng thời điểm hiện tại bởi phương thức khởi tạo của đối tượng, tức là thời điểm khi đối tượng được tạo ra. Coi thời điểm "`start`" này như là thời điểm khi người dùng gọi điện đến, và xem thời điểm khi phương thức "`talk`" được thực thi như là thời điểm khi nhân viên trực tổng đài trả lời cuộc gọi; khoảng chênh lệch là thời gian đợi của người gọi. Biến "`elapsed`" chứa thời gian đợi của người gọi dưới dạng mili-giây. Trong trình giả lập này, 0.1 giây (= 100 mili-giây) được coi như 1 giây của thế giới thực, nên số giây trôi qua trong thế giới thực có thể tính bằng cách chia thời gian cho 100. Tuy nhiên, chú thích trong dòng ngay trước có viết "sau khi làm tròn nó" (làm tròn đến giây gần nhất). Vì thế ta cần chọn công thức cho ta giá trị làm tròn chính xác. Trong Java, khi một số nguyên bị chia bởi một số nguyên khác, phần dư (phần thập phân) bị cắt bỏ, vì thế cộng thêm 50 vào một số để đưa ra cùng kết quả khi làm tròn đến số gần nhất khi chia cho 100. Do đó, đáp án là (a) "`(elapsed + 50) / 100`".

[Câu hỏi con 2]

Đối với các đối số của phương thức khởi tạo trong lớp "`CallCenter`", ta chỉ rõ số nhân viên trực tổng đài, danh sách thời gian gọi, và độ dài của các khoảng thời gian sau khi cuộc gọi được thiết lập. Gắn các đối số của câu hỏi theo định dạng này, cần chú ý rằng có 3 nhân viên trực tổng đài, cứ 30 giây các cuộc gọi được thiết lập, và các cuộc gọi chiếm 70, 90, 100, và 110 giây. Hình sau thể hiện kết quả giả lập dưới các điều kiện đó.



Hình trên cho biết số nhân viên trực tổng đài điện thoại sau 8 giây trôi qua trong trình giả lập (80 giây thời gian thực) là 2, nên câu trả lời đúng là (c).