

MINISTRY OF SCIENCE AND TECHNOLOGY

HOÀ LẠC HIGH TECH PARK
MANAGEMENT BOARD

VIETNAM IT EXAMINATION AND
TRAINING SUPPORT CENTER (VITEC)

BỘ KHOA HỌC VÀ CÔNG NGHỆ

BAN QUẢN LÝ
KHU CÔNG NGHỆ CAO HOÀ LẠC

TRUNG TÂM SÁT HẠCH CÔNG NGHỆ
THÔNG TIN VÀ HỖ TRỢ ĐÀO TẠO (VITEC)

FUNDAMENTAL
INFORMATION TECHNOLOGY
ENGINEER
EXAMINATION

4th April 2004

SÁT HẠCH
KỸ SƯ
CÔNG NGHỆ THÔNG TIN
CƠ BẢN

Ngày 4 tháng 4 năm 2004



<http://www.vitec.org.vn>

Afternoon

Phản thi buổi chiều

**Do not open the exam booklet until
instructed to do so.**

**Inquiries about the exam questions
will not be answered.**

**Không mở đề thi trước khi được
phép.**

**Cán bộ coi thi không giải thích gì thêm
về câu hỏi.**

Spring 2004 VITEC

Fundamental IT Engineer Examination (Afternoon)

Questions must be answered in accordance with the following:

Question Nos.	Q1-Q5	Q6-Q9	Q10-Q13
Question Selection	Compulsory	Select 1 of 4	Select 1 of 4
Examination Time	13:30 ~ 16:00 (150 minutes)		

Instructions:

1. Use an HB pencil. If you need to change an answer, erase your previous answer completely and neatly. Wipe away any eraser debris.
2. Mark your examinee information and test answers in accordance with the instructions below. Your test will not be graded if you do not mark properly. Do not mark or write on the answer sheet outside of the prescribed places.

(1) Examinee Number

Write your examinee number in the space provided, and mark the appropriate space below each digit.

(2) Date of Birth

Write your date of birth (in numbers) exactly as it is printed on your examination admission card, and mark the appropriate space below each digit.

(3) Question Selection (Q6-Q9 and Q10-Q13)

Mark the (S) of the question you select to answer in the “Selection Column” on your answer sheet.

(4) Answers

Mark your answers as shown in the following sample question.

[Sample Question] <http://www.vitec.org.vn>

In which month is this Fundamental IT Engineer Examination conducted?

Answer group:

- a) March b) April c) May d) June

Since the correct answer is “b” (April), mark your answer sheet as follows:

[Sample Reply]

SQ	a	b	c	d
1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

3. “Assembly Language specifications” are provided as a reference at the end of this booklet.

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

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

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

Mùa xuân 2004 VITEC

Kỳ thi kĩ sư Công nghệ thông tin cơ bản (Buổi chiều)

Các câu hỏi phải được trả lời tuân theo hướng dẫn sau:

Số hiệu câu hỏi	Q1-Q5	Q6-Q9	Q10-Q13
Lựa chọn câu hỏi	Bắt buộc	Chọn 1 trong 4 câu	Chọn 1 trong 4 câu
Thời gian làm bài	13:30 ~ 16:00 (150 phút)		

Hướng dẫn:

1. Dùng bút chì HB. Nếu bạn cần thay đổi câu trả lời, hãy xoá sạch câu trả lời trước. Phủi hết bụi tẩy trên giấy.
2. Đánh dấu thông tin dự thi và các câu trả lời của bạn theo hướng dẫn dưới đây. Bài thi sẽ không được chấm điểm nếu không đánh dấu đúng. Không đánh dấu hoặc viết gì ngoài những chỗ đã được qui định trên phiếu trả lời.

(1) Số báo danh

Hãy viết số báo danh của bạn vào chỗ đã cho, và đánh dấu chỗ thích hợp dưới mỗi chữ số.

(2) Ngày sinh

Hãy viết ngày sinh của bạn (bằng số) chính xác như được in trong phiếu dự thi, và đánh dấu chỗ thích hợp dưới mỗi chữ số.

(3) Lựa chọn câu hỏi (Q6-Q9 and Q10-Q13)

Bôi đen ô (S) của câu hỏi mà bạn chọn trả lời trong cột “Selection column” trên phiếu trả lời.

(4) Các câu trả lời

Hãy bôi đen các câu trả lời như được nêu trong câu hỏi mẫu dưới đây.

[Câu hỏi mẫu]

Kì thi sát hạch kĩ sư CNTT cơ bản này được tiến hành vào tháng nào?

Nhóm câu trả lời:

a) Tháng Ba b) Tháng Tư c) Tháng Năm d) Tháng Sáu

Vì câu trả lời đúng là “b)” (Tháng Tư), hãy đánh dấu vào phiếu trả lời như sau:

[Trả lời mẫu]

SQ	a	b	c	d
1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

3. Các đặc tả hợp ngữ được cung cấp làm tài liệu tham khảo tại cuối tập đề thi này.

**Không mở đề thi trước khi được phép.
Cán bộ coi thi không giải thích gì thêm về câu hỏi.**

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

Tên công ty và tên sản phẩm xuất hiện trong các câu hỏi sát hạch là thương hiệu hay thương hiệu đã đăng ký của các công ty đó. Chú ý rằng các kí hiệu ® và ™ không được dùng bên trong.

Questions 1 through 5 are compulsory. Answer every questions.

- Q1.** Read the following text regarding the execution of an instruction, then answer the Subquestion.

There is a computer with a main memory capacity of 65,536 words, wherein one word consists of 16 bits. It has four general purpose registers (0 through 3) and a program register (“PR” below). The format of instruction words (2 words in length) is as follows:

OP	R	X	I	D	adr
----	---	---	---	---	-----

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

OP: Specifies an instruction code in 8 bits. In this example, the following three instruction codes are used.

20_{16} : Sets the effective address in the general purpose register specified by R.

21_{16} : Sets the contents of the word indicated by the effective address in the general purpose register specified by R.

FF_{16} : Ends execution.

R: Specifies a general purpose register number (0 through 3) in two bits.

X: Specifies an index register number (1 through 3) in two bits. The general purpose register at the specified number is used as an index register. However, if “0” is specified, no index register-based modification is made.

I: Specifies “1” in a single bit for indirect address specification; otherwise, specifies “0”.

D: Three extension bits which are always “0”.

adr: Specifies an address in 16 bits.

The instruction word effective address is calculated as shown in the following table.

Table Relationships Between X, I, and Effective Addresses

X	I	Effective address
0	0	adr
1 through 3	0	adr + (X)
0	1	(adr)
1 through 3	1	(adr + (X))

Note (): The parentheses are used to denote information stored in the register or address inside the parentheses.

Subquestion

From the answer group below, select the correct answers to be inserted in the blanks a through d in the following text.

When the contents of the general purpose registers and the main memory had the values shown in the figure below (values are in HEX notation), 0100_{16} was set in PR and the program was executed. In this example, the command at the address 0100_{16} sets the contents 0113_{16} of the address $011B_{16}$ (underlined) to general purpose register 0.

After the execution ends, a is set to general purpose register 0, b to general purpose register 1, c to general purpose register 2, and d to general purpose register 3.

<http://www.vitec.org.vn>

General purpose register 0 : 0003 1 : 0000 2 : 0000 3 : 0000

Program register PR : 0100

Main memory

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	<u>0113</u>	0114	0115	0116	0117
0120	0118	0119	011A	011B	011C	011D	011E	011F

Fig. Values in General Purpose Registers, PR, and Main Memory

Answer group:

- | | | |
|----------------|----------------|----------------|
| 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} |

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

- Q2.** Read the following text regarding the use of regular expressions, then answer subquestions 1 through 3.

There is a ledger used to manage file information such as text and graphics, as shown in the example in Table 1. File information consists of four parameters (file name, type code, version code, date created), and each item is a character string. A search is performed on each of the items in this ledger. The search conditions are specified as regular expressions.

Table 1 An Example of the Ledger

File name	Type code	Version code	Date created
LOGO-T01	JPG	V/R100	2002-01-15
TITLE-A1	GIF	V/R100	2002-01-15
REP-JP01	HTML	V/R203	2002-01-22
OPINION3	TXT	V/R103	2003-02-05

- (1) The character strings for all of the items consist of upper-case letters, numbers, hyphens, and forward slashes, specified in the ASCII coded character sets for information interchange.
- (2) Table 2 below presents the meta-characters used in the regular expression of search conditions. A meta-character is a character used to represent the syntax of an expression.

Table 2 Meta-Characters Used in Regular Expressions

Meta-character	Meaning	Example expression	Explanation of example
.	Represents any single character.	M..N	4-character character string starting with “M” and ending with “N”
(Character string)	A character string surrounded by left and right parentheses is treated as a single pattern.	(MO)	Pattern consisting of the character string “MO”
+	Represents one or more repetitions of the immediately preceding character or pattern.	ABX+	AB and one or more repetitions of X (e.g., ABX, ABXX, ABXXX)
		(MO)+	One or more repetitions of the pattern “MO” (e.g., MO, MOMO, MOMOMO)
*	Represents 0 or more repetitions of the immediately preceding character or pattern.	ABX*	AB and zero or more repetitions of X (e.g., AB, ABX, ABXX)
?	Indicates that the immediately preceding character or pattern appears zero times or one time.	ABCD?	ABC or ABCD
\	The subsequent character is treated not as a meta-character, but as the character itself.	AB*	The character string AB*
	Represents the selection of a character or pattern.	A B	A or B
		(AB) (CDE)	The pattern “AB” or “CDE”
[m–n]	Represents the selection of any single character from a group of consecutive characters going from “m” to “n”.	[3–5]	Any single character (3, 4, or 5) in the group of consecutive characters
		[W–Z]	Any single character (W, X, Y, or Z) in the group of consecutive characters

Subquestion 1

File version codes are registered in the ledger in the syntax shown below.

A version code starts with the three characters “V/R”, followed by a single-digit number (1 through 9) representing the version number and ending with a two-digit number (00 through 99) representing the version branch number. For example, if the version number is 1 and the branch number is 03, then the version code is “V/R103”.

From the following answer group, select the answer which is incorrect as a version code search using regular expressions.

Answer group:

- a) Specify “01” to extract files whose branch code is 01.
- b) Specify “R.. 1” to extract files whose branch code is 01.
- c) Specify “R[1-3]” to extract files whose version number is 3 or less.
- d) Specify “302” to extract a file whose version number and branch code are 3 and 02, respectively.
- e) Specify “V/R302” to extract a file whose version number and branch code are 3 and 02, respectively.

Subquestion 2

Type codes (three-digit or four-digit character strings) indicating file types are registered in the ledger. There are eight different type codes, as shown below.

GIF	HTM	HTML	JPG	JPEG	JPN	MPEG	TXT
http://www.vitec.org.vn							

Files whose type codes are JPG or JPEG need to be extracted. From the following answer group, select the answer which is the correct regular expression to be used for searching for the type codes.

Answer group:

- a) JPEG?
- b) JPEG*
- c) JPE?G
- d) JP+G
- e) JP?G
- f) JP.G

Subquestion 3

File creation dates are registered in the ledger. A creation date is a character string containing a four-digit number (0001 through 9999) representing the calendar year, followed by a two-digit number (01 through 12) representing the month, followed by a two-digit number (01 through 31) representing the day, with hyphens connecting these numbers together. For example, the date March 20, 2003 is represented by “2003-03-20”.

A file creation date search was performed using the regular expression shown below. From the following answer group, select all_dates which are extracted by a search using this regular expression.

..(0|1|2)\-+.1

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Answer group:

- | | | |
|---------------|---------------|---------------|
| a) 2001-02-10 | b) 2001-12-11 | c) 2002-02-21 |
| d) 2002-11-10 | e) 2002-12-01 | f) 2003-01-10 |



<http://www.vitec.org.vn>

- Q3.** Read the following text regarding LAN access control, then answer the subquestion.

CSMA/CD (Carrier Sense Multiple Access/Collision Detect) is an access control system used in bus LANs, which use coaxial cable, and in star LANs, which use twisted-pair cable and hubs.

With CSMA/CD, the following procedure is used during frame transmission for carrier sensing and collision detection between multiple devices connected to a transmission path.

[Description of Frame Transmission Procedure]

Step 0: Wait until there is a frame to be transmitted. When a frame to be transmitted occurs, go to step 1.

Step 1: If a carrier from another connected device is sensed on the transmission path,

go to step 2; otherwise, go to step 3.

Step 2: When the time period determined using a random number passes, return to step 1.

Step 3: Start frame transmission and go immediately to step 4.

Step 4: If there is no collision during frame transmission, the transmission is deemed successful. Return to step 0. If a collision is sensed during frame transmission, go to step 5. This check to determine whether there is a collision during frame transmission is performed because there is a possibility that the start of frame transmission from another connected device cannot be sensed in step 1 due to signal propagation delay on the transmission path.

Step 5: Switch the frame being transmitted to a signal notifying other devices of the occurrence of a collision. After transmitting this for a set length of time, return to step 2.

This signal allows other connected devices to know that a collision has occurred.

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks through in the following text.

There is a LAN connected as shown in the diagram below. The transmission path distance between interconnect device X and interconnect device Y is 230 meters. The signal propagation speed on this transmission path is 230 meters per microsecond.

Frame transmission from interconnect device X has now started. Prior to the elapse of microseconds from the start of this transmission, a frame being transmitted from interconnect device Y occurs. When this happens, interconnect device Y passes through step 1 and goes to . Subsequently, interconnect device Y senses a collision.

The time period required for interconnect device X to sense the collision is, at the longest, approximately microseconds after the start of frame transmission from interconnect device X, depending on the time difference compared to the start of transmission from interconnect device Y, which starts transmission later. If transmission from interconnect device X ends prior to the elapse of this time period, then interconnect device X will go from to and be unable to detect the collision. In this case, assuming interconnect devices can be attached at any position between the terminators at either end, it is clear that with this sensing method, the time required to transmit a single frame must be at least as long as the signal round-trip time between the terminators on the transmission path.

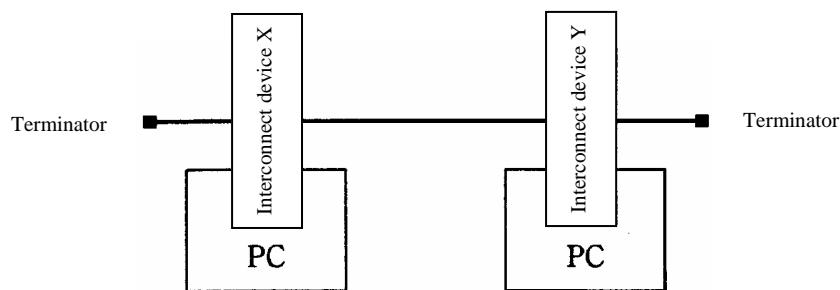


Fig. LAN Connections with the CSMA/CD Method

Answer group for a and c:

- | | | |
|-----------|--------|---------|
| a) 0.0043 | b) 0.2 | c) 1 |
| d) 2 | e) 10 | f) 4300 |

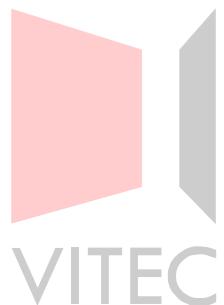
Answer group for b, d, and e:

a) Step 0
d) Step 3

b) Step 1
e) Step 4

c) Step 2
f) Step 5

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

- Q4.** Read the following program description, pseudo-language syntax description, and program, then answer the subquestion.

[Program Description]

`calc` is a subprogram that uses a stack to calculate numerical expressions expressed in Reverse Polish Notation.

- (1) Numerical expressions expressed in Reverse Polish Notation are stored, one character at a time, in the individual elements $\text{Ex}[0], \dots, \text{Ex}[\text{Lp}]$ ($\text{Lp} \geq 3$) of a character-type one-dimensional array `Ex`.
- (2) A numerical expression consists of a positive or negative integer and one or more arithmetic operation symbols. Note that if the integer is positive, a plus sign is not added to it.
- (3) A single blank space is set in front of integers except the first integer.
- (4) Calculations are done using real numbers. The subprogram `Abort()` is called to abort the program if either of the following states occurs during the calculations.
 - ① Zero division is done.
 - ② Something outside the stack is referenced.
- (5) `calc` uses the subprogram `Push`, which adds real numbers to the stack, and the subprogram `Pop`, which removes real numbers from the stack. Tables 1 through 3 below show the argument specifications for each subprogram. In addition, the function `ToReal`, which converts a single numeric character to a real number, is also used.
- (6) The following are defined as global variables: `Stack`, a real type one-dimensional array; `MAX`, a constant which represents the largest element number in `Stack`; and `Sp`, a variable which indicates the location in the stack that is being manipulated. The initial value of `Sp` is “0”.
- (7) Numerical expressions expressed in Reverse Polish Notation are assumed to be correct.

Example: Numerical expression
in infix notation

	0	1	2	3	4	5	6	7	8	9	
$(1 + 2) * (3 - 4)$	<code>Ex</code>	1	△	2	+	△	3	△	4	-	*
$12 \not\propto (-345)$	<code>Ex</code>	0	1	2	3	4	5	6	7		$\text{Lp} = 9$

	0	1	2	3	4	5	6	7	8	9	
$12 \not\propto (-345)$	<code>Ex</code>	1	2	△	-	3	4	5	/		$\text{Lp} = 7$

Note: The triangles denote blank spaces.

Table 1 Calc Argument Specifications

Variable name	Input/output	Meaning
Ex []	Input	Character type one-dimensional array storing a numerical expression expressed in Reverse Polish Notation
Lp	Input	The final element number in the character type one-dimensional array Ex ($L_p \geq 3$)
Ret	Output	Calculation results

Table 2 Push Argument Specifications

Variable name	Input/output	Meaning
T	Input	Real number added to stack

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

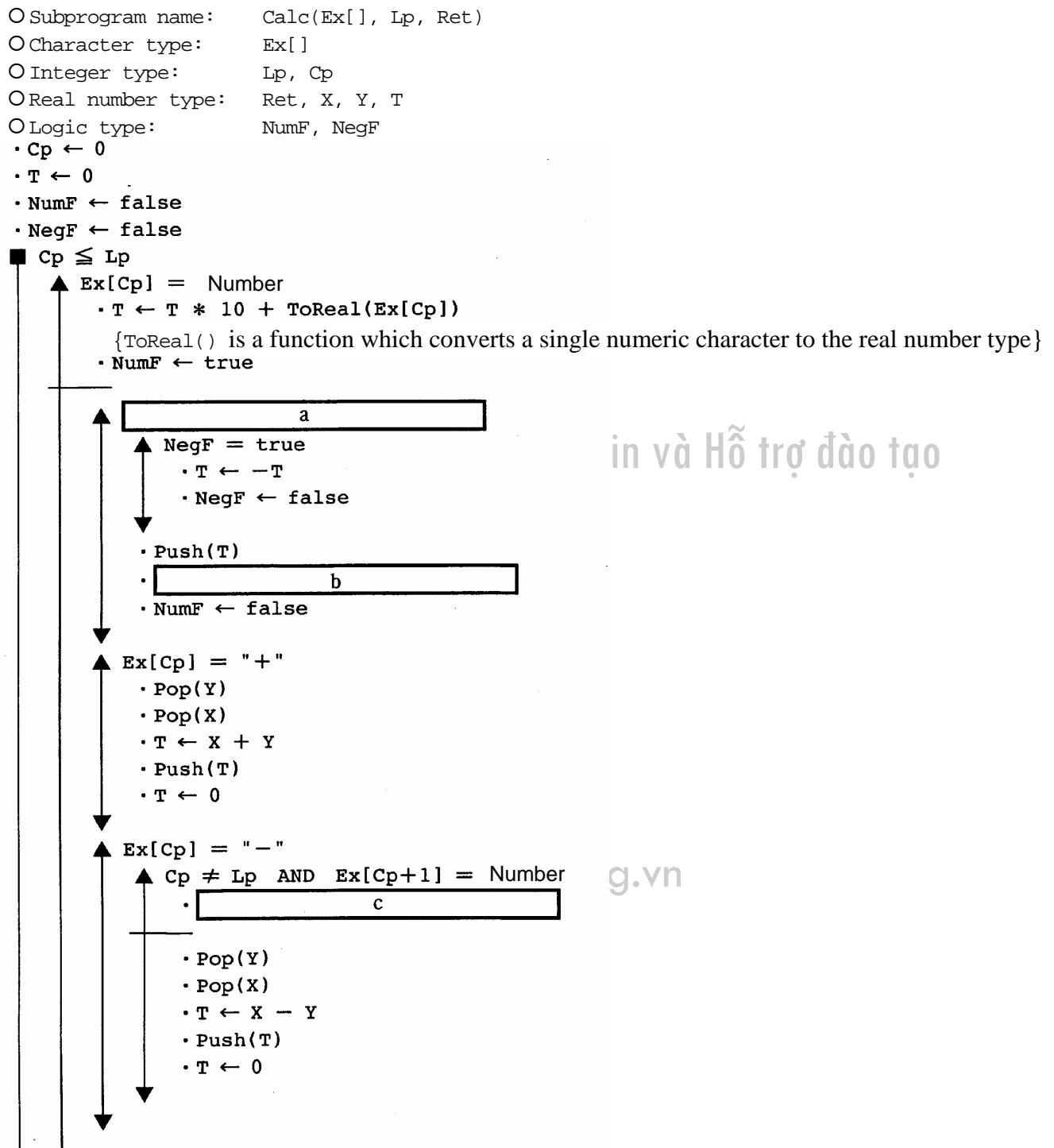
Table 3 Pop Argument Specifications

Variable name	Input/output	Meaning
T	Output	Real number taken from stack

[Description of Pseudo-Language Syntax]

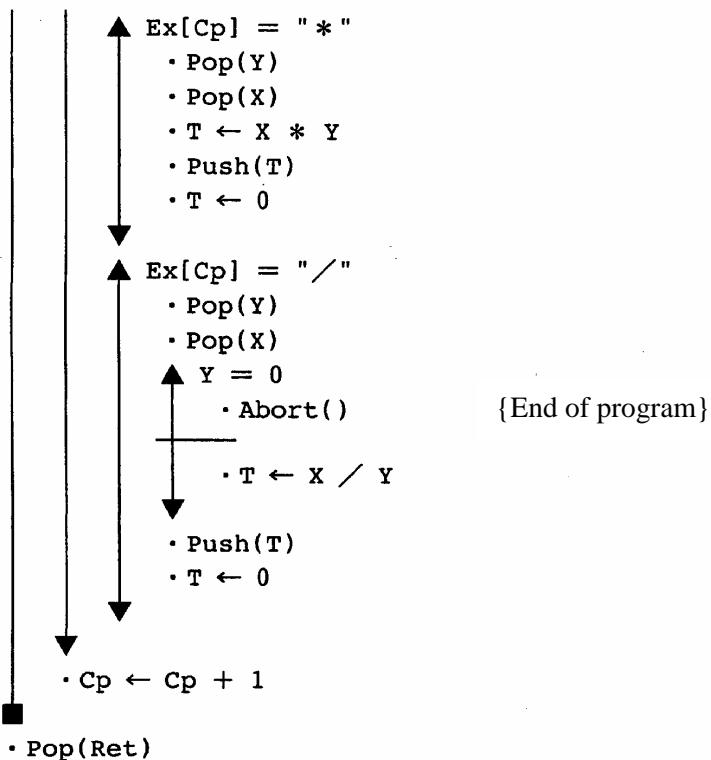
Syntax	Description
○	Declares names, types, etc. of procedures, variables, etc.
• Variable ← Expression	Assigns expression value to Variable.
{Text}	Text is a comment.
↑ Conditional expression • Process 1 — • Process 2 ↓	Denotes a selection process. If the conditional expression is TRUE, then Process 1 is executed; if it is FALSE, then Process 2 is executed.
■ Conditional expression • Process	Denotes a loop with the termination condition at the top. If the conditional expression is TRUE, then the process is executed.

[Program]

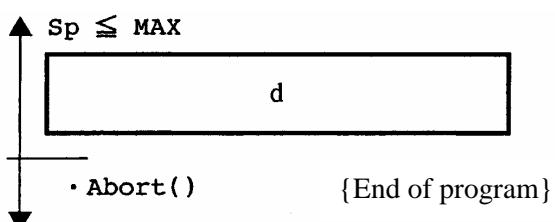


in và Hỗ trợ đào tạo

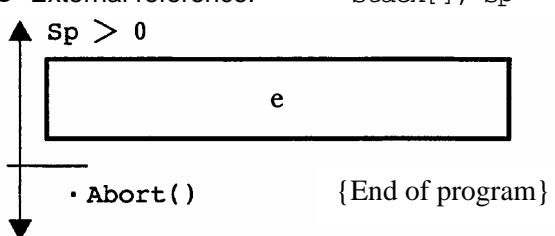
g.vn



- Subprogram name: Push(T)
- Real number type: T
- External reference: Stack[], Sp



- Subprogram name: Pop(T)
- Real number type: T
- External reference: Stack[], Sp



in và Hỗ trợ đào tạo

g.vn

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks
[a] through [e] in the above program.

Answer group for a:

- a) `Ex[Cp] = " "`
- b) `NumF = false`
- c) `NumF = true`
- d) `T = 0`
- e) `T ≠ 0`

Answer group for b and c:

- a) `NegF ← false`
- b) `NegF ← true`
- c) `NumF ← false`
- d) `NumF ← true`
- e) `T ← 0`

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Answer group for d and e:

a) • `Sp ← Sp + 1`
• `Stack[Sp] ← T`

b) • `Sp ← Sp + 1`
• `T ← Stack[Sp]`

c) • `Sp ← Sp - 1`
• `Stack[Sp] ← T`

d) • `Sp ← Sp - 1`
• `T ← Stack[Sp]`

e) • `Stack[Sp] ← T`
• `Sp ← Sp + 1`

f) • `Stack[Sp] ← T`
• `Sp ← Sp - 1`

g) • `T ← Stack[Sp]`
• `Sp ← Sp + 1`

h) • `T ← Stack[Sp]`
• `Sp ← Sp - 1`

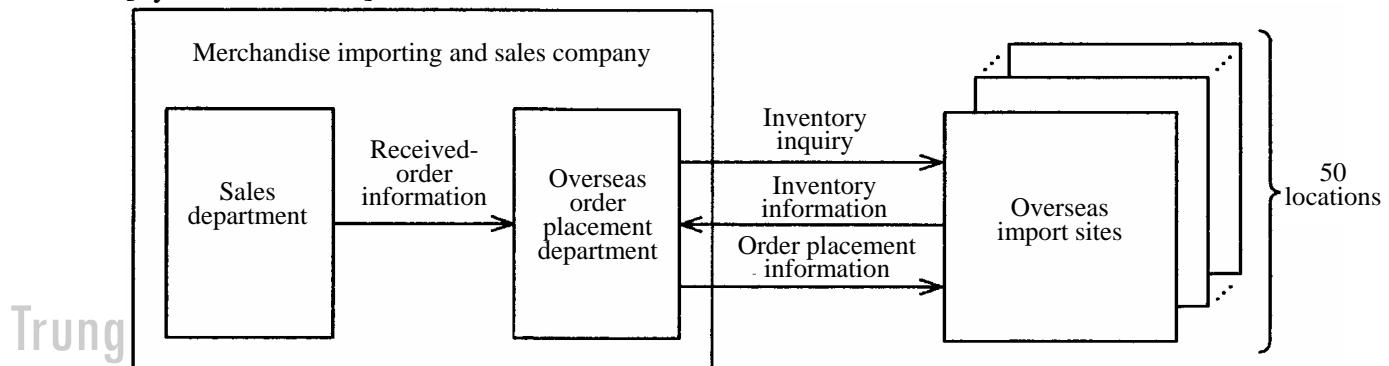


<http://www.vitec.org.vn>

- Q5.** Read the following text regarding program design, then answer subquestions 1 and 2.

An import assignment system is being designed for the overseas order placement department of a merchandise importing and sales company. An overview of this system is presented below.

[System Overview]



- (1) A merchandise code for ordered merchandise, the quantity ordered, and the delivery date desired by the sales department are received online from the sales department. The format of received-order information is shown below.

Merchandise code	Quantity ordered	Desired delivery date		
		Year	Month	Date

- (2) Based on the merchandise code in the received-order information, the import assignment system of the overseas order placement department sends the merchandise code to the overseas import sites carrying that merchandise, and queries the in-stock quantity. There are 50 import sites located throughout the world, and they are connected online with the overseas order placement department. The products carried vary depending on the particular import site. In some cases, multiple import sites carry the same merchandise.

- (3) In response, the import sites promptly send the overseas order placement department inventory information on the queried merchandise. If there is no inventory in stock, then a response is sent with the in-stock quantity set to zero. The overseas order placement department writes the received inventory information to an inventory information file. The inventory information has the following format.

Site code	Merchandise code	In-stock quantity
-----------	------------------	-------------------

(4) There are two different means for delivering merchandise from the import sites to the sales department: ship delivery and air delivery. The number of days required for delivery to the sales department (called the “number of days for delivery”) from a given import site varies depending on whether sea or air shipment is used. The number of days for delivery also varies depending on the particular import site.

The numbers of days for delivery for each import site are stored in a site master file. The site master file has the following format.

Site code	Import site name	Number of days for delivery by ship	Number of days for delivery by air	Other information
-----------	------------------	-------------------------------------	------------------------------------	-------------------

(5) A decision regarding whether importing is possible or not is made and the quantity to import from each import site is determined based on the inventory information from the import sites and the numbers of days for delivery by ship and by air from those sites. Specifically, a check is first performed to determine which import sites have the merchandise in question in inventory, and are able to meet the desired delivery date. Next, the import sites to which orders will be placed are assigned. If the import quantity cannot be satisfied by a single site, then multiple import sites are assigned. The handling procedure for doing this is shown in the flowchart below.

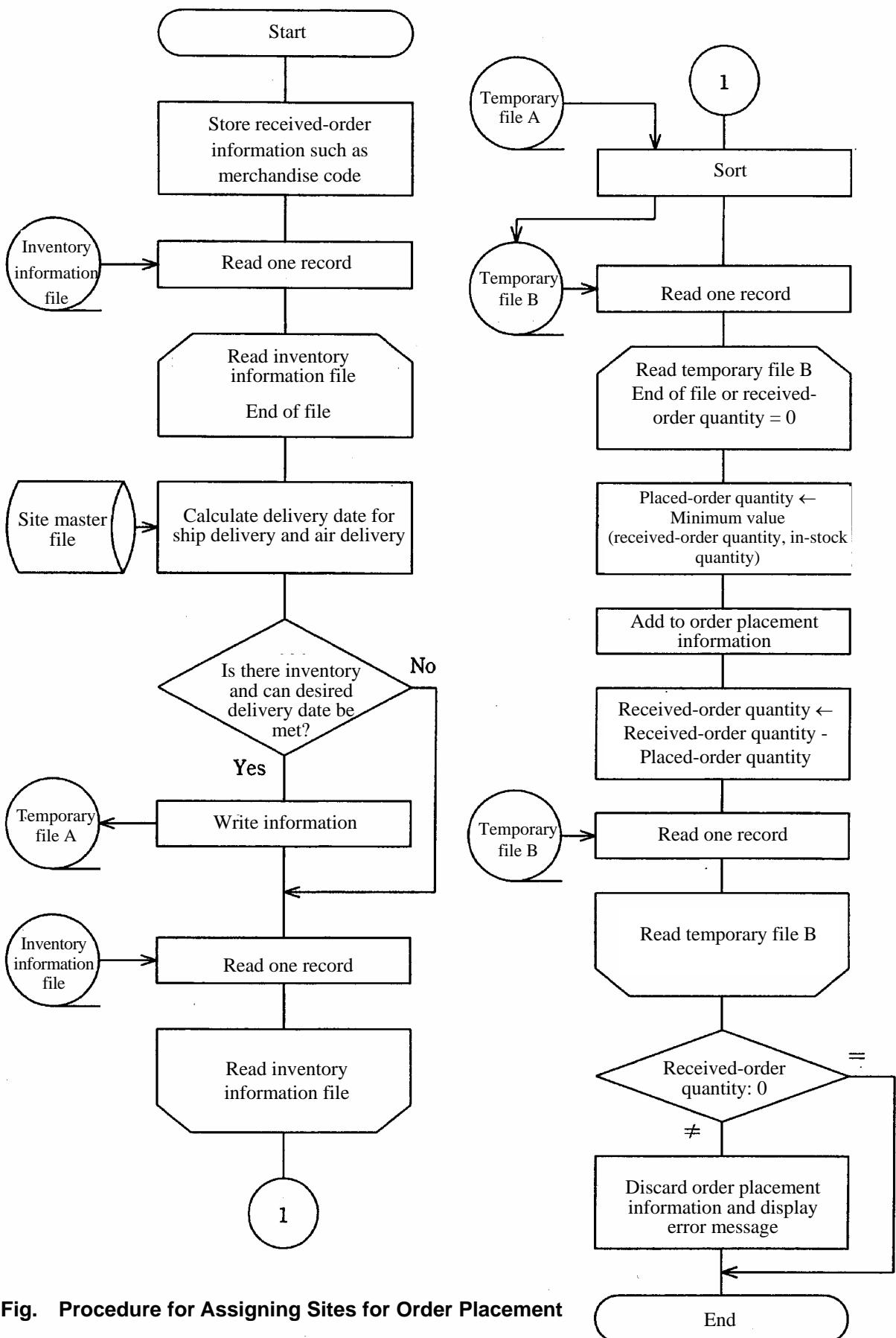
Import sites are assigned according to priority rankings ① through ③.

- ① Higher priority is given to import sites which can satisfy the desired delivery date via ship delivery as opposed to air delivery.
- ② Priority is given to import sites with larger quantities of the merchandise in question.
- ③ Priority is given to import sites with smaller site code numbers.

If it is not possible to obtain the required import quantity by the desired delivery date, an error message to that effect is outputted.

(6) Once the import sites, the order quantities to be placed with them, and the delivery means have been determined, the order placement information is sent to the relevant import sites. The order placement information format is shown below. In the format, the delivery means is stored as either ship delivery or air delivery.

Site code	Merchandise code	Quantity ordered	Delivery means
-----------	------------------	------------------	----------------



Subquestion 1

From the following answer group, select three answers which are correct as data items that must be included in temporary file A shown in the above flowchart.

Answer group:

- a) Desired delivery date
- b) Site code
- c) In-stock quantity
- d) Merchandise code
- e) Delivery means
- f) Number of days for delivery
- g) Placed-order quantity

Subquestion 2

This system needs to be tested. First, consider the following test data as received-order information.

Merchandise code	Quantity ordered	Desired delivery date		
		Year	Month	Date
12345	200	2003	05	10

Assume that a comparison of the desired delivery date and the order reception date shows a margin of 20 days for the number of days for delivery. As shown in the following table, seven different patterns have been set as test data for the in-stock quantities and numbers of days for delivery at the import sites carrying this merchandise. From the answer group below, select the correct answers to be inserted in the blanks **a** through **c** in the table so as to satisfy the expected results indicated therein.

Table Test Pattern Details and Expected Results

Pattern	Site code	In-stock quantity	Number of days for delivery by air	Number of days for delivery by ship	Expected results
1	01	500	10	20	Place order for 200 units by ship at import site with site code 01.
2	02	300	8	21	Place order for 200 units by air at import site with site code 02.
3	03	500	21	21	Error message is displayed.
4	04	100	6	19	Error message is displayed.
	05	50	7	21	
5	06	70	7	20	Place order for 70 units by ship at import site with site code 06; 70 units by air at import site with site code 07; and 60 units by air at import site with site code 08.
	07	70	12	30	
	08	a	8	21	
6	09	150	2	21	Place order for 150 units by air at import site with site code 09; and 50 units by air at import site with site code 10.
	10	b	3	22	
	11	120	4	23	
7	12	c	20	40	Place order for 30 units by air at import site with site code 12; 160 units by ship at import site with site code 13; and 10 units by air at import site with site code 14.
	13	160	10	20	
	14	30	15	25	

Answer group:

a) 20

b) 30

c) 50

d) 70

e) 100

f) 120

g) 160

Select one of the following four questions (Q6, Q7, Q8, or Q9). Be sure to mark the (S) in the Selection Column on your answer sheet for the question that you answered. If you select more than one question, only the first answer will be graded.

- Q6.** Read the following description of a C program and the program itself, then answer subquestions 1 and 2.

[Program Description]

- (1) This program generates graphics with regularity using a two-dimensional array. The program outputs the fractal graphic (called a Sierpinski gasket) shown in the figure below from the initial values shown in row 0 of the following table. In this example, the array has 33 rows and columns.



Fig. Output Results

		Table Generated State Values											32	
		Column	0	1	2	3	4	5	6	7	8	9	...	32
Row		0	0	1	0	0	0	0	0	0	0	0	...	0
Generated state values	1	0	1	1	0	0	0	0	0	0	0	0	...	0
	2	0	1	0	1	0	0	0	0	0	0	0	...	0
	3	0	1	1	1	1	0	0	0	0	0	0	...	0
	4	0	1	0	0	0	1	0	0	0	0	0	...	0
	5	0	1	1	0	0	1	1	0	0	0	0	...	0
	6	0	1	0	1	0	1	0	1	0	0	0	...	0
	7	0	1	1	1	1	1	1	1	1	0	0	...	0
	8	0	1	0	0	0	0	0	0	0	0	1	...	0
	9	0	1	1	0	0	0	0	0	0	0	1	...	0
	:	:	:	:	:	:	:	:	:	:	:	:	...	:
	31	0	1	1	1	1	1	1	1	1	1	1	...	1
	32	0	1	0	0	0	0	0	0	0	0	0	...	0

Trung tam Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

(2) Values indicating the state (0 or 1) are stored in the two-dimensional array s .

The state values in row 1 and the rows which follow it are generated according to the following rules from the initial values stored in row 0, as shown in the table above.

- ① The state value of row i , column j is represented by $s[i][j]$.
- ② All of the values in column 0 are 0.
- ③ The values of $s[i+1][j]$ (where $j \geq 1$) are generated according to the following table, based on the initial values of $s[i][j-1]$ and $s[i][j]$.

State values serving as base		Generated state value $s[i+1][j]$
$s[i][j-1]$	$s[i][j]$	
1	1	0
1	0	1
0	1	1
0	0	0

(3) The values of column 0, row 32 in the two-dimensional array s are not output.

(Program)

```
#include <stdio.h>
#define ALIVE    1
#define DEAD     0
#define SZ       33

int stschk( int, int );

main()
{
    int s[SZ][SZ], i, j;
    for ( i=0; i<SZ; i++ ) s[i][0] = DEAD;
    for ( j=2; j<SZ; j++ ) s[0][j] = DEAD;
    s[0][1] = ALIVE;
    for ( i=0; i<SZ-1; i++ ) {
        for ( j=1; j<SZ; j++ ) {
            [a] = stschk( [b], [c] );
            if ( [b] == ALIVE ) printf( "*" );
            else                  printf( " " );
        }
        printf( "\n" );
    }
}

int stschk( int s1, int s2 )
{
    if ((( s1==DEAD )&&( s2==ALIVE ))|||  

        (( s1==ALIVE )&&( s2==DEAD ))) return ALIVE;  

    else return DEAD;
}
```

Trung

đào tạo

Subquestion 1

From the answer group below, select the correct answers to be inserted in the blanks

[a] through [c] in the above program.

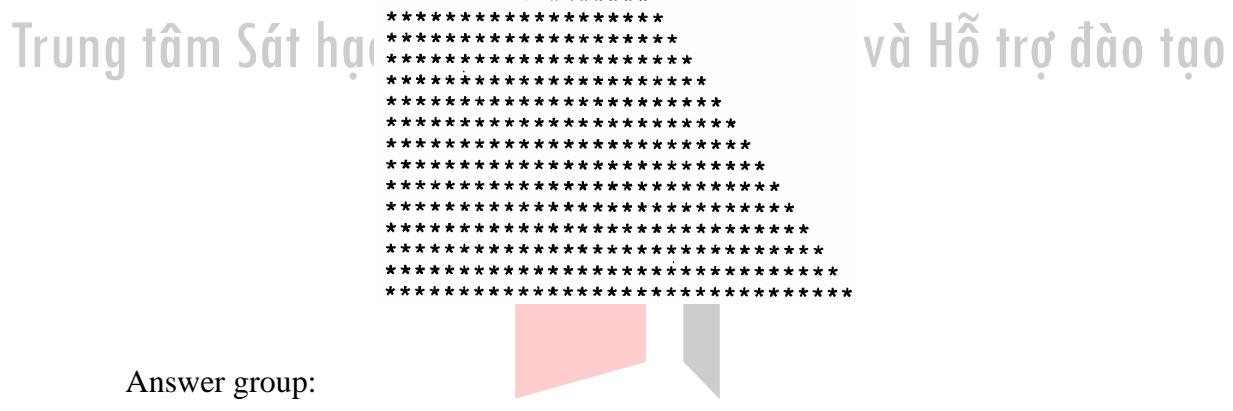
<http://www.vitec.org.vn>

Answer group:

- | | | |
|----------------|--------------|----------------|
| a) s[i-1][j-1] | b) s[i-1][j] | c) s[i-1][j+1] |
| d) s[i][j-1] | e) s[i][j] | f) s[i][j+1] |
| g) s[i+1][j-1] | h) s[i+1][j] | i) s[i+1][j+1] |

Subquestion 2

A graphic such as that shown below was output when the `if` statement for the function `stschk` was changed. From the answer group below, select the `if` statement leading to this result.



The graphic consists of a diamond shape formed by stars (*). The pattern is symmetrical, with the highest density of stars at the center and tapering off towards the edges. A watermark with the text "Trung tâm Sát hạch và Hỗ trợ đào tạo" and "VITEC" is overlaid on the graphic.

Answer group:

- a) `if ((s1==ALIVE) || (s2==ALIVE)) return ALIVE;`
`else return DEAD;`
- b) `if ((s1==ALIVE) && (s2==ALIVE)) return ALIVE;`
`else return DEAD;`
- c) `if ((s1==DEAD) || (s2==DEAD)) return DEAD;`
`else return ALIVE;`
- d) `if (((s1==ALIVE)&&(s2==DEAD))||`
`((s1==DEAD)&&(s2==ALIVE))) return ALIVE;`
`else return DEAD;`

- Q7.** Read the following description of a COBOL program and the program itself, then answer the subquestion.

[Program Description]

This program collects record counts, itemized by fault code, from a repair status file in which PC repair statuses are recorded. In addition, it prints the record counts, itemized by fault code, as well as the corresponding percentages.

- (1) The record format in the repair status file is as follows.

Order number 6 digits	Date in 8 digits	Fault code 3 digits	Other information 26 digits
--------------------------	---------------------	------------------------	--------------------------------

- (2) The print format is as follows.

FAULT CODE	COUNT	PERCENTAGE
XXX	Z,ZZ9	ZZ9.9
XXX	Z,ZZ9	ZZ9.9
XXX	Z,ZZ9	ZZ9.9
:	:	:
XXX	Z,ZZ9	ZZ9.9

- ① The record counts, itemized by fault code, are 9,999 or less.
- ② The percentage values are obtained by dividing the record counts for each fault code by the total record count in the repair status file. Values are truncated to one decimal place.
- ③ A header is printed once at the beginning.

[Program]

```

DATA DIVISION.
FILE SECTION.
FD REPAIR-FILE.
01 REPAIR-REC.
  02 ORDER-REP          PIC X(06).
  02 DATE-REP          PIC X(08).
  02 FAULT-REP         PIC X(03).
  02 EST-REP           PIC X(26).
FD PRINT-FILE.
01 PRINT-REC          PIC X(29).

```

```

SD  SORT-FILE.
01  SORT-REC          PIC  X(03).
WORKING-STORAGE SECTION.
01  END-ST            PIC  9(01).
01  COUNT-T           PIC  9(07).
01  FAULT-WRK         PIC  X(03).
01  COUNT-WRK         PIC  9(04).
01  TOP-HEADER        PIC  X(29) VALUE
                      "FAULT CODE  COUNT  PERCENTAGE".
01  DATA-PRN.
    02  FAULT-PRN       PIC  X(03).
    02                   PIC  X(09) VALUE SPACE.
    02  COUNT-PRN        PIC  Z,ZZ9.
    02                   PIC  X(07) VALUE SPACE.
    02  PER-PRN          PIC  ZZ9.9.

PROCEDURE DIVISION.
SORT-RTN.
    SORT SORT-FILE
    ON ASCENDING KEY SORT-REC
        INPUT PROCEDURE IS SELECT-RTN
        OUTPUT PROCEDURE IS TOTAL-RTN.

STOP RUN.

SELECT-RTN.
    OPEN INPUT REPAIR-FILE.
    INITIALIZE END-ST COUNT-T.
    PERFORM UNTIL END-ST = 1
        READ REPAIR-FILE AT END
        MOVE 1 TO END-ST
        NOT AT END
            RELEASE SORT-REC FROM FAULT-REP
            COMPUTE COUNT-T = COUNT-T + 1
        END-READ
    END-PERFORM.
    CLOSE REPAIR-FILE.

TOTAL-RTN.
    OPEN OUTPUT PRINT-FILE.
    WRITE PRINT-REC FROM TOP-HEADER AFTER PAGE.
    INITIALIZE END-ST.
    RETURN SORT-FILE AT END
    MOVE 1 TO END-ST
    NOT AT END
        PERFORM INIT-RTN
    END-RETURN.
    PERFORM UNTIL END-ST = 1
        RETURN SORT-FILE AT END
        PERFORM PRINT-RTN
        MOVE 1 TO END-ST
        NOT AT END
            IF [ ] a
                COMPUTE COUNT-WRK = COUNT-WRK + 1
            ELSE
                PERFORM PRINT-RTN
            [ ] b
        END-IF

```

```

        END-RETURN
        END-PERFORM.
        CLOSE PRINT-FILE.
PRINT-RTN.
        MOVE COUNT-WRK TO COUNT-PRN.
 .
        MOVE FAULT-WRK TO FAULT-PRN.
 .
INIT-RTN.
        MOVE 1 TO COUNT-WRK.
 .

```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks
 through in the above program.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Answer group for a:

- | | |
|------------------------------|-----------------------------|
| a) FAULT-WRK = FAULT-REP | b) FAULT-WRK = SORT-REC |
| c) FAULT-WRK > FAULT-REP | d) FAULT-WRK > SORT-REC |
| e) FAULT-WRK NOT = FAULT-REP | f) FAULT-WRK NOT = SORT-REC |

Answer group for b and e:

- | |
|--------------------------------|
| a) MOVE FAULT-REP TO FAULT-WRK |
| b) MOVE FAULT-WRK TO FAULT-PRN |
| c) MOVE FAULT-WRK TO FAULT-REP |
| d) MOVE FAULT-WRK TO SORT-REC |
| e) MOVE SORT-REC TO FAULT-PRN |
| f) MOVE SORT-REC TO FAULT-WRK |
| g) PERFORM INIT-RTN |
| h) PERFORM PRINT-RTN |
| i) PERFORM SELECT-RTN |
| j) PERFORM TOTAL-RTN |

Answer group for c:

- a) COMPUTE PER-PRN = 100 * COUNT-T / COUNT-WRK
- b) COMPUTE PER-PRN = 100 * COUNT-WRK / COUNT-T
- c) COMPUTE PER-PRN = COUNT-T / COUNT-WRK
- d) COMPUTE PER-PRN = COUNT-T / COUNT-WRK / 100
- e) COMPUTE PER-PRN = COUNT-WRK / COUNT-T
- f) COMPUTE PER-PRN = COUNT-WRK / COUNT-T / 100
- g) MOVE COUNT-T TO PER-PRN
- h) MOVE COUNT-WRK TO PER-PRN

Answer group for d:

- a) WRITE PRINT-REC
- b) WRITE PRINT-REC AFTER 1
- c) WRITE PRINT-REC FROM DATA-PRN AFTER 1
- d) WRITE PRINT-REC FROM TOP-HEADER AFTER 1



<http://www.vitec.org.vn>

- Q8.** Read the following description of a Java program and the program itself, then answer the subquestion.

[Program Description]

This program calculates the area of a figure and outputs the result. The figure is a triangle, rectangle, or square, and is defined in the program as a figure object with the following attributes.

Triangle: Length of three sides

Rectangle: Length of two sides (vertical and horizontal)

Square: Length of one side

This program is comprised of the following five classes.

AreaTest

This class has the method `main`, and performs the following processes:

- (1) It constructs triangles, rectangles, and square objects, and sets them in array `figures`.
- (2) It obtains the area of each figure and outputs the result. In this case, it is assumed that numerical values resulting in correct figures are provided.

Figure

This class is an abstract class of figures. It declares the abstract method `getArea`, which calculates the area and returns the result.

Triangle

This class is a triangle class. It defines the method `toString`, which returns an attribute as a character string; and the method `getArea`, which calculates the area of a triangle using Heron's Formula and returns the result.

Rectangle

This class is a rectangle class. It defines the method `toString`, which returns an attribute as a character string; and the method `getArea`, which calculates the area of a rectangle and returns the result.

Square

This class is a square class. It defines the method `toString`, which returns an attribute as a character string.

The Program 1 execution results are shown here.

```
Triangle : sides = 2.0, 3.0, 3.0 : area = 2.8284271247461903
Rectangle : height = 5.0, width = 8.0 : area = 40.0
Square : width = 5.0 : area = 25.0
```

Fig. Execution Results

[Program 1]

```
public class AreaTest {
    public static void main(String args[]) {
        Figure[] figures = {
            new Triangle(2, 3, 3),
            new Rectangle(5, 8),
            new Square(5)};
        for (int i = 0; i < figures.length; i++) {
            System.out.println(figures[i] +
                "area = " + figures[i].getArea());
        }
    }
}
```

đào tạo

[Program 2]

```
public abstract class Figure {
    public abstract double getArea();
}
```

[Program 3]

```
public class Triangle extends [REDACTED] a {
    double la;
    double lb;
    double lc;
    public Triangle(double la, double lb, double lc) {
        this.la = la;
        this.lb = lb;
        this.lc = lc;
    }
    public String toString() {
        return "Triangle : sides = " + la + ", " + lb + ", " +
            lc + " : ";
    }
}
```

```

public double getArea() {
    double s = (la + lb + lc) / 2.0;
    return Math.sqrt(s * (s - la) * (s - lb) * (s - lc));
}
}

```

[Program 4]

```

public class Rectangle extends [REDACTED] b {
    double height;
    double width;
    public Rectangle(double height, double width) {
        this.height = height;
        this.width = width;
    }
    public String toString() {
        return "Rectangle : height = " + height +
               ", width = " + width + " : ";
    }
    public double getArea() {
        return [REDACTED] c [REDACTED];
    }
}

```

t_{ào} t_{ạo}

[Program 5]

```

public class Square extends [REDACTED] d {
    public Square(double width) {
        [REDACTED] e [REDACTED];
    }
    public String toString() {
        return "Square : width = " + width + " : ";
    }
}

```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks [REDACTED] a through [REDACTED] e in the above programs. The same answer may be selected more than once.

Answer group for a, b, and d:

- | | | |
|--------------|-----------|------------|
| a) abstract | b) Figure | c) getArea |
| d) Rectangle | e) Square | f) super |

Answer group for c:

- a) height
- b) height * height
- c) height * width
- d) width
- e) width * width

Answer group for e:

- a) super(height)
- b) super(height, height)
- c) super(width)
- d) super(width, height)
- e) super(width, width)
- f) this.height = height
- g) this.height = width
- h) this.width = height
- i) this.width = width

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

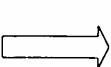
- Q9.** Read the following description of an assembler program and the program itself, then answer subquestions 1 and 2.

[Program Description]

Sixteen connected words are treated as a matrix consisting of four rows and four columns. The subprogram ROTATE is a program which rotates the contents of matrix 90° clockwise and stores it in matrix Y.

Word 0	A	B	C	D
Word 4	E	F	G	H
Word 8	I	J	K	L
Word 12	M	N	O	P

Matrix X



M	I	E	A
N	J	F	B
O	K	G	C
P	L	H	D

Matrix Y

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

- (1) The leading address of matrix X is set in GR1, and is transferred from a main program.
- (2) The leading address of matrix Y is set in GR2, and is transferred from a main program.
- (3) It is assumed that the areas where matrix X and matrix Y are stored do not overlap.
- (4) The original contents of the general purpose register are restored when returning from a subprogram.

VITEC

<http://www.vitec.org.vn>

[Program]

(Line number)

```
1 ROTATE START ;  
2 RPUSH ;  
3 LAD GR3,16 ;  
4 LOOP0 LAD GR4,4 ;  
5 LOOP1 LD GR5,0,GR1 ; } Move one word from matrix X  
6 ST GR5,3,GR2 ; } to matrix Y  
7 SUBA GR3,=1 ;  
8 JZE FIN ;  
9 LAD GR1,1,GR1 ; Update matrix X pointer  
10 SUBA GR4,=1 ;  
11 JZE NXTROW ; End of processing for one row?  
12 [ ] a ;  
13 JUMP LOOP1 ;  
14 NXTROW [ ] b ; Go to next row  
15 JUMP LOOP0 ; ; tin và Hỗ trợ đào tạo  
16 FIN RPOP ;  
17 RET ;  
18 END ;
```

Tru

tin và Hỗ trợ đào tạo

Subquestion 1

From the answer group below, select the correct answers to be inserted in the blanks

a

b

Answer group:

- | | |
|--------------------|--------------------|
| a) LAD GR2,-16,GR2 | b) LAD GR2,-15,GR2 |
| c) LAD GR2,-14,GR2 | d) LAD GR2,-13,GR2 |
| e) LAD GR2,1,GR2 | f) LAD GR2,2,GR2 |
| g) LAD GR2,3,GR2 | h) LAD GR2,4,GR2 |
| i) LAD GR2,6,GR2 | j) LAD GR2,8,GR2 |

VITEC

<http://www.vitec.org.vn>

Subquestion 2

From the answer groups below, select the correct answers to be inserted in the blanks
[c] and [d] in the following text.

In order to change the contents of matrix X so that it is rotated 90° counterclockwise and stored in matrix Y, line number 5 should be changed to [c] and line number 9 should be changed to [d].

Word 0	A	B	C	D
Word 4	E	F	G	H
Word 8	I	J	K	L
Word 12	M	N	O	P

→

D	H	L	P
C	G	K	O
B	F	J	N
A	E	I	M

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Answer group for c:

- | | |
|----------------------------|----------------------------|
| a) LOOP1 LD GR5 , 3 , GR1 | b) LOOP1 LD GR5 , 4 , GR1 |
| c) LOOP1 LD GR5 , 7 , GR1 | d) LOOP1 LD GR5 , 8 , GR1 |
| e) LOOP1 LD GR5 , 15 , GR1 | f) LOOP1 LD GR5 , 16 , GR1 |

Answer group for d:

- | | |
|-----------------------|-----------------------|
| a) LAD GR1 , -4 , GR1 | b) LAD GR1 , -3 , GR1 |
| c) LAD GR1 , -2 , GR1 | d) LAD GR1 , -1 , GR1 |
| e) LAD GR1 , 2 , GR1 | f) LAD GR1 , 3 , GR1 |
| g) LAD GR1 , 4 , GR1 | h) LAD GR1 , 5 , GR1 |

Select one of the following four questions (Q10, Q11, Q12, or Q13). Be sure to mark the in the Selection Column on your answer sheet for the question that you answered. If you select more than one question, only the first answer will be graded.

- Q10.** Read the following description of a C program and the program itself, then answer the subquestion.

[Program Description]

A program is to be created to determine a temperature correction coefficient for correcting the temperature of a certain piezoelectric sensor (a device which generates voltage according to the magnitude of pressure applied). The output characteristics of this piezoelectric sensor vary according to the ambient temperature, so the output value must be corrected. The results shown in the following table were obtained through an experimental study of changes in the piezoelectric sensor output value relative to the ambient temperature. This table shows ratios, with the output value at 0°C equal to 1.00.

Table Ambient Temperatures and Sensor Output Value Ratios

Temperature	Sensor output value ratio
-40°C	0.20
-20°C	0.60
-10°C	0.80
0°C	1.00
10°C	1.17
30°C	1.50
50°C	1.80

In order to correct the piezoelectric sensor output value to a value which is not dependent on the ambient temperature during measurement, a temperature correction coefficient K is to be determined based on the data in the table. The temperature is corrected by multiplying K by the piezoelectric sensor output value.

- (1) A temperature correction table is created from the data in the above table. The temperature correction table is established as a structure array.

```
typedef struct {
    int     Temp;   /* Temperature */  

    double  Ratio;  /* Sensor output value ratio (actual measurement value) */  

    double  Step;   /* Increment per 1°C */  

} CURVE;
```

- (2) The function `SetupCurve`, which initializes the temperature correction table, and the function `GetK`, which determines the temperature correction coefficient K, are created.
- (3) In order to determine the temperature correction coefficient K for the temperature Degree, `GetK` searches for the temperature correction table using a binary search method. The ambient temperature is assumed to be no lower than -40°C and no higher than 50°C . If there is no corresponding temperature value in the temperature correction table, the sensor output value ratio corresponding to the temperature Degree is determined by linear interpolation, and the inverse of this value is returned as K.
- (4) The main program is a test program which determines and displays the temperature correction coefficient K from -40°C to 50°C in increments of 1°C , in order to verify the operations of these two functions. An example of this display is shown below.

Temperature	Temperature correction coefficient
-40	5.00
-39	4.55
-38	4.17
-37	3.85
-36	3.57
-35	3.33
-34	3.13
-33	2.94
-32	2.78
-31	2.63
-30	2.50
-29	2.38
-28	2.27
-27	2.17
-26	2.08
-25	2.00
:	

[Program]

```
#include <stdio.h>

typedef struct {
    int      Temp      ; /* Temperature                               */
    double   Ratio     ; /* Sensor output value ratio (actual measurement value) */
    double   Step      ; /* Increment per 1°C                         */
} CURVE ;

void    SetupCurve( CURVE * , int );
double  GetK ( int , CURVE * , int );
#define   ITEMS   7

main()
{
    int      Degree ;
    double   k ;
    CURVE  Curve[ ITEMS ] = {
        { -40 , 0.20 , 0.0 },{ -20 , 0.60 , 0.0 },
        { -10 , 0.80 , 0.0 },{  0 , 1.00 , 0.0 },
        { 10 , 1.17 , 0.0 },{ 30 , 1.50 , 0.0 },
        { 50 , 1.80 , 0.0 } };

    SetupCurve( Curve , ITEMS );
    printf(" Temperature Temperature correction coefficient \n" );
    for( Degree = -40 ; Degree <= 50 ; Degree++ ) {
        k = GetK( Degree , Curve , ITEMS );
        printf( " %3d      %4.2f \n" , Degree , k );
    }
}

void SetupCurve( CURVE *p , int Points )
{
    /* Initialize temperature correction table */
    int i ;
    for( i = 0 ; i < Points - 1 ; i++ , [ a ] ){
        p->Step = ( [ b ] ) / ( (p+1)->Temp - p->Temp );
    }
}

double GetK( int Temp , CURVE *p , int Points )
{
    /* Return temperature correction coefficient K as return value */
    int i,j,n ;
    i = 0 ; j = Points - 1 ;
    if ( ( Temp < p->Temp ) || ( Temp > (p+j)->Temp ) )
        return 0.0; /* Return 0.0 if outside temperature range */
```

Trí

lào tạo

```

/* If this matches temperature of final element in temperature correction table */
if ( Temp == ( p+j )->Temp )
    return 1.0 / ( p+j )->Ratio ;

/* Search for temperature correction table using binary search method */

while( 1 ) {
    n = [c] ;
    if ( ( Temp >= ( p+n )->Temp ) &&
        ( Temp < ( p+n+1 )->Temp ) ) break ;

    if ( Temp < ( p+n )->Temp ) j = n ;
    else [d] ;
}

p += n ;
return 1.0 / ( p->Ratio + p->Step * ( [e] ) );
}

```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks
[**a**] through [**e**] in the above program.

Answer group for a:

- | | | |
|-----------|-----------|--------------|
| a) p-- | b) p++ | c) p->Temp++ |
| d) p += 3 | e) p += 7 | |

Answer group for b:

- | | |
|----------------------------|--------------------------------|
| a) (p-1)->Ratio - p->Ratio | b) (p-1)->Ratio - (p+1)->Ratio |
| c) p->Ratio - (p-1)->Ratio | d) p->Ratio - (p->Ratio - 1) |
| e) (p+1)->Ratio - p->Ratio | |

Answer group for c:

- | | |
|------------------|------------------|
| a) (i + j) / 2 | b) (i * j) / 2 |
| c) (i % j) / 2 | d) (i + j) * 2 |
| e) (i % j) * 2 | |

Answer group for d:

- | | | |
|------------|----------------|----------------|
| a) $i = 0$ | b) $i = n - 1$ | c) $i = n + 1$ |
| d) $j = n$ | e) $j = n + 1$ | |

Answer group for e:

- | | | |
|-------------------|-------------------|-------------------|
| a) Temp - p->Step | b) Temp - p->Temp | c) Temp + p->Temp |
| d) p->Temp - n | e) p->Temp - Temp | |

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

- Q11.** Read the following description of a COBOL program and the program itself, then answer the subquestion.

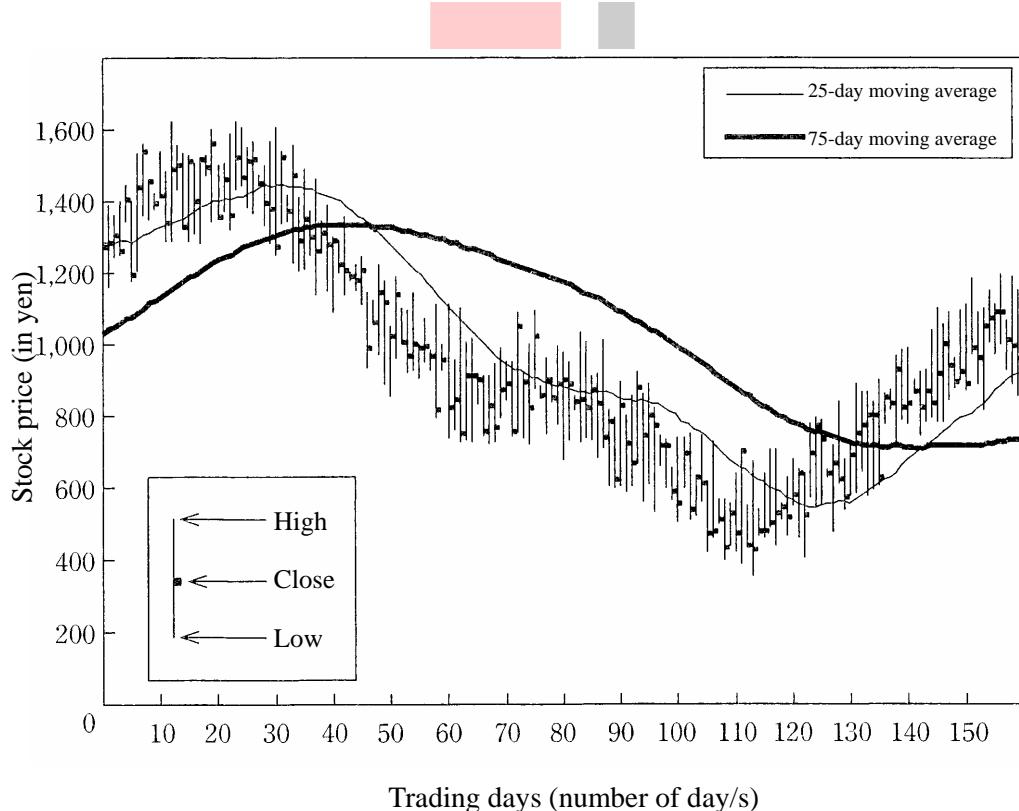
This program simulates stock trades (purchases and sales) according to the rules described below, using past stock-price information. It reads a file containing the daily prices for a certain stock issue, writes trade records based on these rules to an output file, and lastly displays the profit/loss which is the result of the series of trades.

[Program Description]

- (1) The stock trading rules are as follows.

- ① A 25-day moving average and a 75-day moving average are used. In this case, an n-day moving average is a line graph which determines the mean value for the stock price over the preceding n trading days, moving along the time axis. If the price on trading day t is P_t , the n-day moving average $MA_t(n)$ at trading day t is determined by the following formula.

$$MA_t(n) = \frac{P_t + P_{t-1} + \dots + P_{t-n+1}}{n}$$



- ② The stock's closing price (the price at which the stock last traded on a given day) is used to calculate the moving average.
- ③ When the 25-day moving average crosses the 75-day moving average from below to the upside, the stock is purchased on the next trading day. Specifically, if $s < t$ and $MA_s(25) < MA_s(75)$ is established, once this inequality changes to $MA_t(25) < MA_t(75)$ for the first time, the stock is purchased on trading day $t + 1$.
- ④ When the 25-day moving average crosses the 75-day moving average from above to the downside, the stock is sold on the next trading day. Specifically, if $s < t$ and $MA_s(25) > MA_s(75)$ is established, once this inequality changes to $MA_t(25) < MA_t(75)$ for the first time, the stock is sold on trading day $t + 1$. However, the stock cannot be sold if it has not already been purchased.
- ⑤ It is assumed that the stock can be purchased and sold at the mean value of the high (the highest traded price) and low (the lowest traded price) on trading day t .
- ⑥ It is assumed that there are no other costs related to trading.
- ⑦ It is assumed that one share of stock is traded per time.

- (2) The input file (**INFILE**) is a sequential file in which stock-price data for at least 75 trading days on a given stock issue are stored in a time series. The record format is as follows.

Date 8 digits	Open 9 digits	Close 9 digits	High 9 digits	Low 9 digits
------------------	------------------	-------------------	------------------	-----------------

- (3) The output file (**OUTFILE**) is a sequential file containing records of trades occurring when the trading rules in (1) are applied to the stock-price data in the input file. The record format is as follows.

Date 8 digits	Flag 1 digit	Trade price 10 digits
------------------	-----------------	--------------------------

- ① The record output to the output file (**OUTFILE**) contains the date corresponding to a day (trading day t) when a trading decision is made, as well as a flag and the trade price.
- ② The flag contains an “S” for a sale, and a “B” for a buy.
- ③ The trade price contains a value with one digit below the decimal point.

- (4) The profit/loss for the series of trades is displayed as a numerical value with a plus or minus sign, and a single digit beyond the decimal point.

Example :

Total Revenue:	-114.5
----------------	--------

[Program]

```

DATA DIVISION.
FILE SECTION.
FD INFILE.
01 I-MARKET-PRICE-REC.
  05 I-YYYYMMDD-MARKET      PIC X(8).
  05 I-OPENING-PRICE        PIC 9(9).
  05 I-CLOSING-PRICE        PIC 9(9).
  05 I-HIGHEST-PRICE        PIC 9(9).
  05 I-LOWEST-PRICE         PIC 9(9).

FD OUTFILE.
01 O-EXCHANGE-REC.
  05 O-YYYYMMDD-EXCHANGE    PIC X(8).
  05 O-ACTION               PIC X.
  05 O-PRICE                PIC 9(9)V9.

WORKING-STORAGE SECTION.
01 SERIES-SIZE            PIC 9(4) VALUE 100.
01 RANGE-SHORT             PIC 9(4) VALUE 25.
01 RANGE-LONG              PIC 9(4) VALUE 75.
01 W-MARKET-PRICE-REC.
  05 W-YYYYMMDD-MARKET      PIC X(8).
  05 W-OPENING-PRICE        PIC 9(9).
  05 W-CLOSING-PRICE        PIC 9(9).
  05 W-HIGHEST-PRICE        PIC 9(9).
  05 W-LOWEST-PRICE         PIC 9(9).

01 MARKET-PRICE-SERIES.
  05 TBL-MARKET-PRICE-REC OCCURS 100.
    10 TBL-YYYYMMDD-MARKET    PIC X(8).
    10 TBL-OPENING-PRICE      PIC 9(9).
    10 TBL-CLOSING-PRICE      PIC 9(9).
    10 TBL-HIGHEST-PRICE      PIC 9(9).
    10 TBL-LOWEST-PRICE       PIC 9(9).

01 SERIES-TOP              PIC 9(4).
01 TAIL-LONG                PIC 9(4).
01 TAIL-SHORT               PIC 9(4).
01 SUM-SHORT                 PIC 9(12).
01 SUM-LONG                  PIC 9(12).
01 MA-SHORT                  PIC 9(9)V9.
01 MA-LONG                   PIC 9(9)V9.
01 MA-COMPARE-CURRENT      PIC X.
  88 LONG-LT-SHORT-CUR       VALUE "S".
  88 SHORT-LT-LONG-CUR       VALUE "L".
  88 SHORT-EQ-LONG-CUR       VALUE "E".

01 MA-COMPARE-PREVIOUS     PIC X.
  88 LONG-LT-SHORT-PRE       VALUE "S".

```

Trun

Hỗ trợ đào tạo

```

88  SHORT-LT-LONG-PRE      VALUE "L".
88  SHORT-EQ-LONG-PRE      VALUE "E".
01  INFILE-EOF-FLG         PIC X.
  88  INFILE-EOF            VALUE "Y".
  88  INFILE-NOT-EOF        VALUE "N".
01  TRADING-PRICE          PIC 9(9)V9.
01  REVENUE                PIC S9(9)V9.
01  REVENUE-ED              PIC +++++++9.9.
01  INDX                   PIC 9(9).
01  TRADING-ACTION         PIC S9.
  88  FLG-NO-TRADE         VALUE 0.
  88  FLG-SELL              VALUE 1.
  88  FLG-BUY                VALUE -1.
01  STOCK-HOLDING          PIC X.
  88  HAS-STOCK             VALUE "Y".
  88  HAS-NO-STOCK          VALUE "N".
PROCEDURE DIVISION.
MAIN-PARAGRAPH.
  OPEN INPUT INFILE  OUTPUT OUTFILE.
  PERFORM INIT-MOVING-AVARAGE.
  IF MA-SHORT = MA-LONG THEN
    SET SHORT-EQ-LONG-CUR TO TRUE
  END-IF.
  PERFORM SET-CURRENT-MA-COMPARISON.
  MOVE ZERO TO REVENUE.
  SET HAS-NO-STOCK TO TRUE.
  SET FLG-NO-TRADE TO TRUE.
  SET INFILE-NOT-EOF TO TRUE.
  PERFORM UNTIL INFILE-EOF
    IF FLG-BUY OR FLG-SELL THEN
      PERFORM TRADE-STOCK-AND-MAKE-RECORD
    END-IF
    READ INFILE INTO W-MARKET-PRICE-REC
    AT END
      SET INFILE-EOF TO TRUE
    NOT AT END
      PERFORM CALC-NEXT-MOVING-AVERAGE
      PERFORM DECIDE-TRADING
    END-READ
  END-PERFORM.
  MOVE REVENUE TO REVENUE-ED.
  DISPLAY "Total Revenue: " REVENUE-ED.
  CLOSE INFILE OUTFILE.
  STOP RUN.

INIT-MOVING-AVARAGE.
  PERFORM VARYING SERIES-TOP FROM 1 BY 1
    UNTIL RANGE-LONG < SERIES-TOP
    READ INFILE INTO TBL-MARKET-PRICE-REC(SERIES-TOP)
  END-PERFORM.
  COMPUTE TAIL-SHORT = SERIES-TOP - RANGE-SHORT.
  COMPUTE TAIL-LONG = SERIES-TOP - RANGE-LONG.
  SUBTRACT 1 FROM SERIES-TOP.
  MOVE ZERO TO SUM-SHORT.
  PERFORM VARYING INDX [REDACTED] a

```

Trí

rợ đào tạo

```

        ADD TBL-CLOSING-PRICE(INDX) TO SUM-SHORT
END-PERFORM.
MOVE SUM-SHORT TO SUM-LONG.
PERFORM VARYING INDX [ ] b
        ADD TBL-CLOSING-PRICE(INDX) TO SUM-LONG
END-PERFORM.
COMPUTE MA-SHORT ROUNDED = SUM-SHORT / RANGE-SHORT.
COMPUTE MA-LONG ROUNDED = SUM-LONG / RANGE-LONG.
SET-CURRENT-MA-COMPARISON.
EVALUATE TRUE
    WHEN MA-SHORT = MA-LONG
        CONTINUE
    WHEN MA-SHORT < MA-LONG
        SET SHORT-LT-LONG-CUR TO TRUE
    WHEN MA-LONG < MA-SHORT
        SET LONG-LT-SHORT-CUR TO TRUE
    END-EVALUATE.
TRADE-STOCK-AND-MAKE-RECORD.
EVALUATE TRUE
    WHEN FLG-BUY
        MOVE "B" TO O-ACTION
        SET HAS-STOCK TO TRUE
    WHEN FLG-SELL AND [ ] c
        MOVE "S" TO O-ACTION
        SET HAS-NO-STOCK TO TRUE
    END-EVALUATE.
IF O-ACTION = "B" OR O-ACTION = "S"
THEN
    COMPUTE TRADING-PRICE ROUNDED
        = ( TBL-HIGHEST-PRICE(SERIES-TOP)
            + TBL-LOWEST-PRICE(SERIES-TOP) ) / 2
    COMPUTE REVENUE = REVENUE + TRADING-PRICE * [ ] d
    MOVE TBL-YYYYMMDD-MARKET(SERIES-TOP) TO
        O-YYYYMMDD-EXCHANGE
    MOVE TRADING-PRICE TO O-PRICE
    WRITE O-EXCHANGE-REC
END-IF.
CALC-NEXT-MOVING-AVERAGE.
ADD W-CLOSING-PRICE TO SUM-SHORT SUM-LONG.
SUBTRACT TBL-CLOSING-PRICE(TAIL-SHORT) FROM SUM-SHORT.
SUBTRACT TBL-CLOSING-PRICE(TAIL-LONG) FROM SUM-LONG.
COMPUTE MA-SHORT ROUNDED = SUM-SHORT / RANGE-SHORT.
COMPUTE MA-LONG ROUNDED = SUM-LONG / RANGE-LONG.
ADD 1 TO SERIES-TOP TAIL-SHORT TAIL-LONG.
EVALUATE TRUE
    WHEN SERIES-SIZE < SERIES-TOP
        MOVE 1 TO SERIES-TOP
    WHEN SERIES-SIZE < TAIL-SHORT
        MOVE 1 TO TAIL-SHORT
    WHEN SERIES-SIZE < TAIL-LONG
        MOVE 1 TO TAIL-LONG
END-EVALUATE.
MOVE W-MARKET-PRICE-REC TO TBL-MARKET-PRICE-REC(SERIES-TOP).

```

DECIDE-TRADING.

MOVE MA-COMPARE-CURRENT TO MA-COMPARE-PREVIOUS.

PERFORM SET-CURRENT-MA-COMPARISON.

SET FLG-NO-TRADE TO TRUE.

EVALUATE TRUE

WHEN e AND LONG-LT-SHORT-CUR

SET FLG-BUY TO TRUE

WHEN LONG-LT-SHORT-PRE AND SHORT-LT-LONG-CUR

SET FLG-SELL TO TRUE

END-EVALUATE.

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks

a

e

in the above program.

Answer group for a and b:

- a) FROM 1 BY 1 UNTIL SERIES-SIZE < INDX
- b) FROM SERIES-TOP - 1 BY -1 UNTIL INDX < TAIL-SHORT
- c) FROM SERIES-TOP BY -1 UNTIL INDX <= TAIL-SHORT
- d) FROM TAIL-LONG BY 1 UNTIL TAIL-SHORT <= INDX
- e) FROM TAIL-SHORT BY 1 UNTIL SERIES-TOP < INDX

Answer group for c through e:

- | | |
|-----------------------|----------------------|
| a) FLG-BUY | b) FLG-SELL |
| c) HAS-STOCK | d) LONG-LT-SHORT-CUR |
| e) MA-COMPARE-CURRENT | f) SHORT-LT-LONG-PRE |
| g) TRADING-ACTION | |

- Q12.** Read the following description of a Java program and the program itself, then answer subquestions 1 and 2.

[Program Description]

The library at a certain school has 30 self-learning seats which are administrated according to the following rules.

(1) Administration rules

- ① A student desiring to use a seat submits a request at the front desk and uses the assigned seat. When the student finishes using the seat, he or she notifies the front desk.
- ② If there is a student desiring a seat but no seat is available, the student who has currently been using a seat for the longest time in excess of one hour must vacate that seat for the new student desiring a seat. If there are no vacant seats and no student has been using a seat for more than one hour, then the student desiring a seat cannot use a seat.
- ③ A single student cannot use multiple seats at the same time.

(2) A program was designed as follows to assist in managing the self-learning seats based on the above administration rules.

- ① The class `ListElement` is defined for implementing the bi-directional list shown in the diagram below. The bi-directional list is ring-shaped, and is designed so that the terminal end does not receive special treatment in element insertion and deletion processes. `ListElement` expresses both a list head (beginning) and elements. The following methods are implemented in `ListElement`.

`nextElement`

Returns the next element of this `ListElement` instance.

`previousElement`

Returns the previous element of this `ListElement` instance.

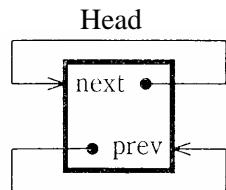
`insertBefore`

Inserts this `ListElement` instance before the element specified in the argument.

`remove`

Removes this `ListElement` instance from the list.

Empty list



List containing elements

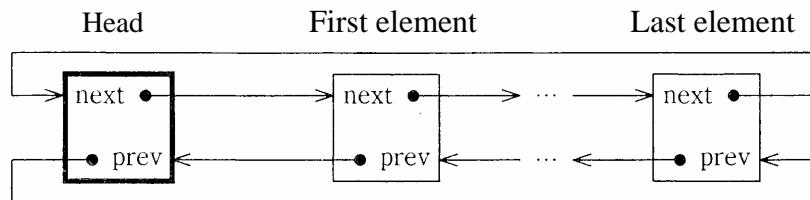


Fig. Bi-Directional List States for Empty List and List Containing Elements

- ② Seats are indicated by the class `Seat`, which extends `ListElement`. A seat number is assigned to each seat.
- ③ The class `SeatManager` is used to implement methods for carrying out the administration rules.
- ④ The lists of free seats and occupied seats are, respectively, `freeSeats` and `occupiedSeats`. Seat instances become elements of each list.
- ⑤ Occupied seats are registered in `occupiedSeats`. The list is managed so that the seats are sorted from the one with the shortest usage time to the one with the longest usage time.
- ⑥ `SeatManager` implements the following public methods.

<http://www.vitec.org.vn>

checkin

Confirms that a student desiring a seat who is specified in an argument is not already using a seat. If there is a free seat in `freeSeats` (free seats list), that seat is assigned to the student desiring a seat. Otherwise, a seat which has been used for more than one hour is assigned. The assigned seat is registered as the first element in `occupiedSeats` (occupied seats list), and that `Seat` instance is returned. If there are no available seats, `null` is returned.

checkout

Deletes the seat used by the user specified in the argument from occupiedSeats, moves it back to freeSeats, and returns true. If the specified user is not found in occupiedSeats, false is returned.

First, the class `ListElement` and the class `Seat` were implemented and a unit test was performed to confirm that operations were correct. Next, a test was performed with the class `SeatManager` implemented. In this test, a problem was discovered, wherein a single student could use multiple seats at the same time. In other respects, normal operations were confirmed.

[Program 1]

```
public class ListElement {  
    private ListElement prev, next;  
    public ListElement() {  
        prev = next = this;  
    }  
    public ListElement nextElement() { return next; }  
    public ListElement previousElement() { return prev; }  
    public void insertBefore(ListElement element) {  
        next = element;  
        prev = element.prev;  
        next.prev = prev.next = a;  
    }  
    public void remove() {  
        b = next;  
        c = prev;  
        prev = next = this;  
    }  
}
```

[Program 2]

```
public class Seat extends ListElement {  
    private String userID;      // User  
    private long checkinTime;   // Check-in time  
    private int seatNumber;     // Seat number  
  
    public Seat(int seatNumber) {  
        this.seatNumber = seatNumber;  
    }  
    public int getSeatNumber() {  
        return seatNumber;  
    }  
}
```

```

public String getUserId() {
    return userID;
}
public void setUserId(String userID) {
    this.userID = userID;
}
public boolean isUsedBy(String userID) {
    return this.userID.equals(userID);
}
public long getCheckinTime() {
    return checkinTime;
}
public void setCheckinTime(long time) {
    checkinTime = time;
}
}

```

[Program 3]

Trí ùo tào

```

public class SeatManager {
    private static final int NSEATS = 30; // Number of seats
    // Maximum usage time (ms)
    private static final int MAXTIME = 60 * 60 * 1000;
    // Free seats list
    private ListElement freeSeats = new ListElement();
    // Occupied seats list
    private ListElement occupiedSeats = new ListElement();

    public SeatManager() {
        for (int i = 1; i <= NSEATS; i++) {
            Seat seat = new Seat(i);
            seat.insertBefore(freeSeats);
        }
    }

    // If there is a free seat on the free seats list, the Seat instance is deleted from the free seats list
    // and the instance is returned. If there are no free seats, null is returned.
    private Seat getFreeSeat() {
        ListElement le = freeSeats.nextElement();
        if (le != freeSeats) {
            le.remove();
            return (Seat) le;
        }
        return null;
    }

    // The occupied seats list is checked. If there is a user whose seat usage time exceeds the maximum usage time,
    // a message to that effect is outputted and checkout is called.
    private void vacateExpiredSeat(long time) {
        ListElement le = [REDACTED] d;
    }
}

```

```

        if (le != occupiedSeats) {
            Seat seat = (Seat) le;
            if ((seat.getCheckinTime() + MAXTIME) < time) {
                System.out.println("Seat#" +
                    seat.getSeatNumber() + " " +
                    seat.getUserID() +
                    " must check out.");
                checkout(seat.getUserID());
            }
        }
    }

//The seat used by the user specified in the occupied seats list is searched for. If the user is found,
//the corresponding seat is returned. Otherwise, null is returned.
private Seat findUser(String userID) {
    ListElement le = [REDACTED];
    while (le != occupiedSeats) {
        Seat seat = (Seat) le;
        if (seat.isUsedBy(userID)) {
            return seat;
        }
        le = le.nextElement();
    }
    return null;
}

public Seat checkin(String userID) {
    long now = System.currentTimeMillis();
    Seat seat = getFreeSeat();
    if (seat == null) {
        vacateExpiredSeat(now);
        seat = getFreeSeat();
    }
    if (seat != null) {
        seat.setCheckinTime(now);
        seat.setUserID(userID);
        seat.insertBefore(occupiedSeats.nextElement());
    }
    return seat;
}

public boolean checkout(String userID) {
    Seat seat = findUser(userID);
    if (seat != null) {
        seat.remove();
        seat.setUserID(null);
        seat.insertBefore(freeSeats);
        return true;
    }
    return false;
}
}

```

True

to tạo

Subquestion 1

From the answer groups below, select the correct answers to be inserted in the blanks through in the above programs.

Answer group for a through c:

- a) element
- b) element.next
- c) element.prev
- d) new ListElement()
- e) next
- f) next.prev
- g) null
- h) prev
- i) prev.next
- j) this

Answer group for d and e:

- a) freeSeats
- b) freeSeats.nextElement()
- c) freeSeats.previousElement()
- d) occupiedSeats
- e) occupiedSeats.nextElement()
- f) occupiedSeats.previousElement()

Subquestion 2

Program 3 needs to be fixed so that a single student cannot use multiple seats at the same time. From the answer group below, select the method that is suitable for making this adjustment. Note that this question assumes that the correct answers have been entered for all of the blanks through in the above programs.

<http://www.vitec.org.vn>

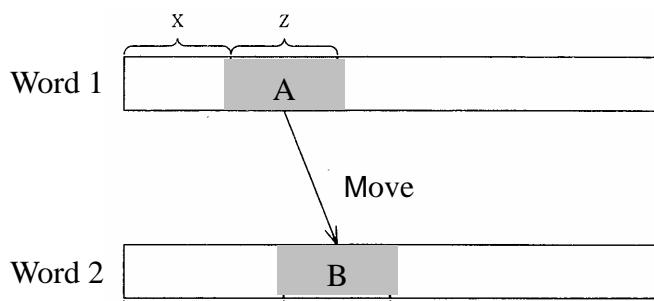
Answer group:

- a) In the method checkin, occupiedSeats is checked prior to assigning a seat. If the student desiring a seat is already using another seat, an error occurs and the seat assignment process is not performed.
- b) In the method checkout, seats which are no longer being used are not correctly removed from occupiedSeats. Therefore, correctly delete seats which are no longer being used from occupiedSeats.
- c) In the method vacateExpiredSeat, the method checkout is called to force a user who has exceeded the maximum usage time (MAXTIME) to checkout. Immediately thereafter, a process is added to delete the seat used by that user from occupiedSeats.

- Q13.** Read the following description of an assembler program and the program itself, then answer Subquestions 1 through 3.

[Program Description]

The subprogram BMOVE moves bit string A in Word 1 to position B in Word 2 as shown below. x, y, and z all indicate numbers of bits.



- (1) The information shown below is set in GR1 through GR5 and called from a main program.

GR1: Word 1 address

GR2: Word 2 address

GR3: x

GR4: y

GR5: z

- (2) The following is assumed: $x \geq 0, y \geq 0, z \geq 1$.
- (3) The following is assumed: $x + z \leq 16, y + z \leq 16$.
- (4) Word 1 and Word 2 are different words.
- (5) The original contents of the general purpose register are restored when returning from a subprogram.

[Program]

(Line number)

```
1 BMOVE START ;  
2 PUSH 0,GR6 ; } Save register contents  
3 PUSH 0,GR7 ; }  
4 LD GR6,=#8000 ; } Line up z "1" bits  
5 [ ] a ;  
6 LD GR7,GR6 ;  
7 SRL GR7,0,GR4 ; }  
8 XOR GR7,=#FFFF ; } Set part B in Word 2 to "0"  
9 AND GR7,0,GR2 ; }  
10 [ ] b ; } Remove bit string A of Word 1  
11 AND GR6,0,GR1 ; }  
12 SLL GR6,0,GR3 ; } Shift the removed bit string to position B  
13 [ ] c ; }  
14 OR GR6,GR7 ;  
15 ST GR6,0,GR2 ; } Restore the original register contents  
16 POP GR7 ;  
17 POP GR6 ;  
18 RET ; ; thông tin và Hỗ trợ đào tạo  
19 END ;
```

Trung

Subquestion 1

From the answer group below, select the correct answers to be inserted in the blanks
[] through [] in the above program.

Answer group:

- | | |
|-------------------|-------------------|
| a) SLA GR6,0,GR3 | b) SLA GR6,0,GR4 |
| c) SLL GR6,0,GR3 | d) SLL GR6,0,GR4 |
| e) SRA GR6,-1,GR3 | f) SRA GR6,-1,GR5 |
| g) SRL GR6,-1,GR5 | h) SRL GR6,0,GR3 |
| i) SRL GR6,0,GR4 | |

Subquestion 2

From the answer group below, select the answer which correctly indicates the contents of GR6 immediately after the execution of the instruction (SLL) in line number 12, when the contents of Word 1, Word 2, x, y, and z transferred from the main program are as shown below.

Word 1	0010	0110	1101	0111
Word 2	1100	1101	0111	1010

$$x = 9$$

$$y = 3$$

$$z = 5$$

Answer group:

Irung

ào tạo

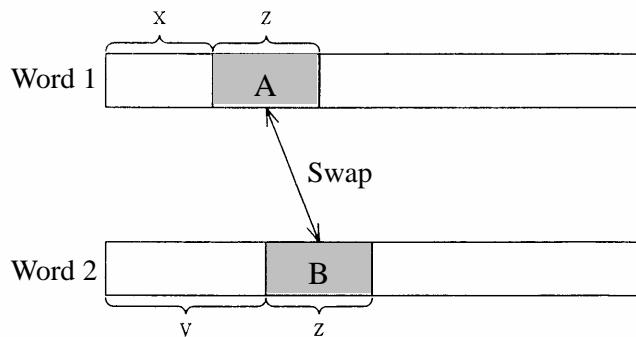
- | | | | | |
|----|------|------|------|------|
| a) | 0000 | 0000 | 0001 | 0101 |
|----|------|------|------|------|
- | | | | | |
|----|------|------|------|------|
| b) | 0000 | 0000 | 0101 | 0100 |
|----|------|------|------|------|
- | | | | | |
|----|------|------|------|------|
| c) | 0001 | 0101 | 0000 | 0000 |
|----|------|------|------|------|
- | | | | | |
|----|------|------|------|------|
| d) | 1010 | 1000 | 0000 | 0000 |
|----|------|------|------|------|



<http://www.vitec.org.vn>

Subquestion 3

A program BSWAP, which swaps bit string A in Word 1 with bit string B in Word 2 using the subprogram BMOVE, was created. From the answer group below, select the correct answers to be inserted in the blanks **d** and **e** as follows. In this case, the individual parameters are set in GR1 through GR5 and called as in BMOVE.



Trung ti

lõi trợ đào tạo

[Line number]

```
1 BSWAP  START          ;  
2      RPUSH           ; Save register  
3      LD    GR6, 0, GR2  ;  
4      ST    GR6, WORD   ; Save Word 2 to WORD  
5      CALL  BMOVE        ;  
6      LD    GR2, GR1     ; Set address of Word 1 as address of Word 2  
7      d                ; Set address of WORD as address of Word 1  
8      PUSH  0, GR4       ; } Swap x and y  
9      e                ; }  
10     POP   GR3          ;  
11     CALL  BMOVE        ;  
12     RPOP             ; Restore register  
13     RET               ;  
14 WORD  DS   1           ;  
15 END               ;
```

Answer group:

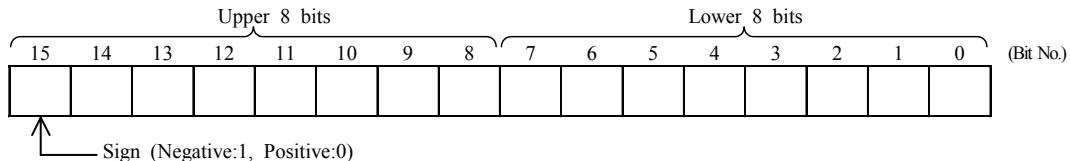
- | | | | |
|--------|-----------|---------|-----------|
| a) LAD | GR1, WORD | b) LAD | GR4, WORD |
| c) LD | GR1, WORD | d) LD | GR3, GR4 |
| e) LD | GR4, GR3 | f) LD | GR4, WORD |
| g) POP | GR4 | h) PUSH | 0, GR3 |

Assembly Language Specifications

1. COMET II Hardware Specifications

1.1 Hardware Specifications

- (1) One word is 16 bits, and the bit format is as follows:



- (2) Main storage capacity is 65,536 words with address numbers 0 through 65,535.
(3) Numeric values are expressed as 16-bit binary numbers. Negative numbers are expressed in complements of two.
(4) Control is sequential. COMET II utilizes a one-word or two-word instruction word.
(5) The COMET II has four types of registers: GR (16 bits), SP (16 bits), PR (16 bits) and FR (3 bits).
There are eight GR (General Register) registers, GR0 through GR7. These eight registers are used for arithmetic, logical, compare and shift operations. Of these, GR1 through GR7 are also used as index registers to modify addresses.
The stack pointer stores the address currently at the top of the stack.
The PR (Program Register) stores the first address of the next instruction.
The FR (Flag Register) consists of three bits: OF (Overflow Flag), SF (Sign Flag) and ZF (Zero Flag). The following values are set, depending on the result generated by certain operation instructions. These values are referenced by conditional branch instructions.

OF	When the result of an arithmetic operation instruction is out of the range of -32,768 to 32,767, the value is 1, and in other cases, the value is 0. When the result of a logical operation instruction is out of the range of 0 to 65,535, the value is 1, and in other cases, the value is 0.
SF	When the sign of the operation result is negative (bit number 15 = 1), the value is 1, and in other cases, the value is 0.
ZF	When the operation result is 0 (all bits are 0), the value is 1, and in other cases, the value is 0.

- (6) Logical addition or logical subtraction: Treats the data to be added or subtracted as unsigned data, and performs addition or subtraction.

1.2 Instructions

Formats and functions of instructions are described in the following chart. When an instruction code has two types of operands, the upper operand shows the instruction between registers and the lower operand shows the instruction between register and main storage.

Instruction	Format		Description of instructions	FR setting
	Opcode	Operand		

(1) Load, store, load address instruction

Load	LD	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r2)$ $r \leftarrow (\text{effective address})$	O*1
Store	ST	$r,adr [x]$	Effective address $\leftarrow (r)$	-
Load Address	LAD	$r,adr [x]$	$r \leftarrow \text{effective address}$	-

(2) Arithmetic and logical operation instructions

ADD Arithmetic	ADDA	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) + (r2)$ $r \leftarrow (r) + (\text{effective address})$	O
ADD Logical	ADDL	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) +_L (r2)$ $r \leftarrow (r) +_L (\text{effective address})$	
SUBtract Arithmetic	SUBA	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) - (r2)$ $r \leftarrow (r) - (\text{effective address})$	
SUBtract Logical	SUBL	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) -_L (r2)$ $r \leftarrow (r) -_L (\text{effective address})$	
AND	AND	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) \text{ AND } (r2)$ $r \leftarrow (r) \text{ AND } (\text{effective address})$	O*1
OR	OR	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) \text{ OR } (r2)$ $r \leftarrow (r) \text{ OR } (\text{effective address})$	
eXclusive OR	XOR	$\frac{r1,r2}{r,adr [x]}$	$r1 \leftarrow (r1) \text{ XOR } (r2)$ $r \leftarrow (r) \text{ XOR } (\text{effective address})$	

(3) Compare operation instructions

ComPare Arithmetic	CPA	$\frac{r1,r2}{r,adr [x]}$	Performs an arithmetic compare or logical compare operation on (r1) and (r2) or (r) and (effective address), and sets FR as follows, according to the result of the compare operation.	O*1
ComPare Logical	CPL	$\frac{r1,r2}{r,adr [x]}$	Compare result	
			FR value	
			SF	
			ZF	
			(r1) > (r2)	
			0	
			0	
			0	
			1	
			1	
			0	

(4) Shift operation instructions

Shift Left Arithmetic	SLA r,adr [,x]	Shifts (r) (excluding the sign bit) left or right by the number of bits specified by the effective address. When a left shift is performed, those bits that are left vacant by the shift operation are filled with zeroes. When a right shift is performed, those bits that are left vacant by the shift operation are filled with the same value as the sign bit.	O*2
Shift Right Arithmetic	SRA r,adr [,x]	Shifts (r) (including the sign bit) left or right by the number of bits specified by the effective address. Those bits that are left vacant by the shift operation are filled with zeroes.	
Shift Left Logical	SLL r,adr [,x]	Shifts (r) (including the sign bit) left or right by the number of bits specified by the effective address.	
Shift Right Logical	SRL r,adr [,x]	Shifts (r) (including the sign bit) left or right by the number of bits specified by the effective address. Those bits that are left vacant by the shift operation are filled with zeroes.	

(5) Branch instructions

Jump on PLus	JPL adr [,x]	Branches to the effective address, depending on the value of FR. If control does not branch to a new address, execution continues with the next instruction.	-
Jump on MInus	JMI adr [,x]		
Jump on Non Zero	JNZ adr [,x]		
Jump on ZEro	JZE adr [,x]		
Jump on OVerflow	JOV adr [,x]		
unconditional JUMP	JUMP adr [,x]	Branches unconditionally to the effective address.	

(6) Stack operation instructions

PUSH	PUSH adr [,x]	SP \leftarrow (SP) $-_L$ 1, (SP) \leftarrow effective address	-
POP	POP r	r \leftarrow ((SP)), SP \leftarrow (SP) $_L +$ 1	

(7) Call and return instructions

CALL subroutine	CALL adr [,x]	SP \leftarrow (SP) $-_L$ 1, (SP) \leftarrow (PR), PR \leftarrow effective address	-
RET from subroutine	RET	PR \leftarrow ((SP)), SP \leftarrow (SP) $_L +$ 1	

(8) Other

SuperVisor Call	SVC adr [,x]	Determine based on the effective address as the argument. After the execution, GR and FR are undefined.	-
No OPeration	NOP	N/A	

Note)	r, r1, r2	All of these represent GR. Values from GR0 to GR7 can be specified.
	adr	This represents the address. A value from 0 to 65,535 can be specified.
x		This represents GR used as the index register. A value from GR1 to GR7 can be specified.
[]		Square brackets ([]) indicate that the specification contained in the brackets may be omitted.
()		The contents of the register or address contained in the parentheses ().
Effective address		A value produced by adding, through "logical addition," adr and the contents of x, or the address pointed at by that value.
←		This means that the operation result is stored in the left part register or address.
+L, -L		Logical addition and logical subtraction.
Effective address for FR setting	O : Setting is performed. O*1 : Setting is performed, but 0 is set to OF. O*2 : Setting is performed, but the bit value sent from the register is set to OF. - : The value before execution is stored.	

1.3 Character Set

- (1) A JIS X0201 Romaji/katakana character set that uses 8-bit codes is used.
- (2) Part of the character set is shown in the right table. Eight bits are used to represent one character; the upper four bits indicate the column in the table, and the lower four bits indicate the row. For example, the hexadecimal codes for the space character, "4," "H," and "¥" are 20, 34, 48 and 5C, respectively. The characters that correspond to the hexadecimal codes 21 to 7E (and A1 to DF omitted in this table) are called "graphics characters." Graphics characters can be displayed (printed) as characters on an output device.
- (3) If any characters not listed in this table and the bit configuration for those characters are needed, they are given in the problem.

Column Row	02	03	04	05	06	07
0	Spac	0	@	P	'	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	\$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(8	H	X	h	x
9)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	[k	{
12	,	<	L	¥	l	
13	-	=	M]	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

2 Specifications of the CASL II Assembly Language

2.1 Specifications of the language

- (1) CASL II is an assembly language for the COMET II.
- (2) A program consists of instruction lines and comment lines.
- (3) One instruction is described in one instruction line, and cannot continue to the next line.
- (4) Instruction lines and comment lines are written from the first character of the line in the following description formats:

Line type		Description format
Instruction line	With operand	[label] {blank} {instruction code} {blank} {operand} [{blank} [comment]]
	Without operand	[label] {blank} {instruction code} [{blank} [{;} [comment]]]
Comment line		[blank] {;} [comment]

(Note) [] Square brackets ([]) indicate that the specification contained in the brackets may be omitted.

{ } Braces ({ }) indicate that the specification contained in the braces is mandatory.
Label Label is the name used to refer to the address of (the first word of) the instruction from other instructions and programs. A label must be 1 to 8 characters in length, and the leading character must be an uppercase alphabetic letter. Either uppercase alphabetic letters or numeric characters can be used for the subsequent characters. Reserved words, GR0 through GR7, are not available.

Blank One or more space characters.

Instruction The description format is defined by instruction.

code Operand The description format is defined by instruction.

Comment Optional information such as memorandums that can be written in any characters allowed by the processing system.

2.2 Instruction Types

CASL II consists of four assembler instructions (START, END, DS and DC), four macro instructions (IN, OUT, RPUSH and RPOP) and machine language instructions (COMET II instructions). The specifications are as follows:

Instruction type	Label	Instruction code	Operand	Function
Assembler instruction	Label	START	[Execution start address]	Defines the top of a program. Defines the starting address for execution of a program. Defines the entry name for reference in other programs.
		END		Defines the end of a program.
	[label]	DS	Word length	Allocates an area.
	[label]	DC	Constant[, constant]...	Defines a constant.
Macro instruction	[label]	IN	Input area, input character length area	Input character data from input devices.
	[label]	OUT	Output area, output character length area	Output character data from output devices.
	[label]	RPUSH		Stores the contents of GR in the stack
	[label]	RPOP		Stores the contents of stack in GR
Machine language instruction	[label]	(See "1.2 Instructions")		

2.3 Assembler Instructions

Assembler instructions are used for assembler control, etc.

(1)	START	[Execution start address]
-----	-------	---------------------------

The START instruction defines the top of a program.

The label name that is defined within this program specifies the execution start address. If the label is specified, execution begins from the address, and if the label is omitted, execution begins from the next instruction of the START instruction.

The label for this instruction can be referred to from other programs as the entry name.

(2)	END	
-----	-----	--

The END instruction defines the end of a program.

(3)	DS	Word length
-----	----	-------------

The DS instruction allocates an area of the specified word length.

The word length is specified by a decimal constant (≥ 0). If "0" is specified for the word length of an area, the area is not allocated, but the label is valid.

(4)	DC	Constant[, constant] ...
-----	----	--------------------------

The DC instruction stores the data that has been specified as a constant in (consecutive) words.

There are four types of constants: decimal constants, hexadecimal constants, character constants and address constants.

Type of constant	Format	Description of instruction
Decimal constant	n	This instruction stores the decimal value specified by "n" as one word of binary data. If "n" is outside of the range of -32,768 to 32,767, only the lower 16 bits of n are stored.
Hexadecimal constant	#h	Assume "h" is a four-digit hexadecimal number. (Hexadecimal notation uses 0 through 9 and A through F.) This instruction stores the hexadecimal value specified by "h" as one word of binary data. (0000 \leq h \leq FFFF)
Character constant	'character string'	This instruction allocates a continuous area for the number of characters (> 0) in the character string. The first character is stored in bits 8 through 15 of the first word, the second character is stored in bits 8 through 15 of the second word, and so on, so that the character data is stored sequentially in memory. Bits 0 through 7 of each word are filled with zeroes. Spaces and any of the graphics characters can be written in a character string. Apostrophes ('') must be written twice consecutively.
Address constant	Label	This instruction stores an address corresponding to the label name as one word of binary data.

2.4 Macro Instructions

<http://www.vitec.org.vn>

Macro instructions use a pre-defined group of instructions and operand data to generate a group of instructions that performs a desired function (the word length is undefined).

(1)	IN	Input area, input character length area
-----	----	---

The IN instruction reads one record of character data from a previously assigned input device.

The input area operand should be the label of a 256-word work area, and the input data is input in this area beginning at the starting address, one character per word. No record delimiter code (such as a line return code, when using a keyboard) is stored. The storage format is the same as character constants with the DC instruction. If the input data is less than 256 characters long, the previous data is left as is in the remaining portion of the input area. If the input data exceeds 256 characters, the excess characters are ignored.

The input character length area should be the label of the one-word work area, and the character length that was input (≥ 0) is stored as binary data. If the end-of-file indicator is encountered, -1 is stored.

When the IN instruction is executed, the contents of GR registers are saved but the contents of FR are undefined.

(2)	OUT	Output area, output character length area
-----	-----	---

The OUT instruction writes character data as one record of data to the previously assigned output device.

The output area operand should be the label of the area where the data to be output is stored, one

character per word. The storage format is the same as character constants with the DC instruction. Bits 0 through 7 do not have to be zeroes because the OS ignores them.

The output character length area should be the label of the one-word work area, and the character length that is to be output (≥ 0) is stored as binary data.

When the OUT instruction is executed, the contents of the GR registers are saved but the contents of FR are undefined.

(3)	RPUSH	
-----	-------	--

The RPUSH instruction stores the contents of GR in the stack in order of GR1, GR2, ... and GR7.

(4)	RPOP	
-----	------	--

The RPOP instruction takes out of the contents of the stack sequentially, and stores in GR in order of GR7, GR6, ... and GR1.

2.5 Machine Language Instructions

Operands of machine language instructions are described in the following formats:

- r, r1, r2 GR is specified using a symbol from GR0 to GR7.
x GR used as the index register can be specified by a symbol from GR1 to GR7.
adr The address is specified by a decimal constant, a hexadecimal constant, an address constant or a literal.

A literal can be described by attaching the equal sign (=) before a decimal constant, a hexadecimal constant or a character constant. CASL II generates a DC instruction by specifying the constant after the equal sign as the operand, and sets the address to the adr value.

2.6 Other

- (1) The relative positions of the instruction words and areas generated by the assembler conform to the order of the descriptions in the assembly language program. All DC instructions generated from literals are located just before the END instruction.
(2) The instruction words and areas that are generated occupy a continuous area in the main memory.

3. Guide to Program Execution

3.1 OS

The following arrangements exist regarding program execution.

- (1) The assembler interprets undefined labels (of those labels written in the operand column, any that are not defined within the program) as entry names (START instruction labels) for other programs. In this case, the assembler refrains from determining the address and entrusts that task to the OS. Before executing the program, the OS performs link processing with entry names for other programs and determines the addresses (program linking).
(2) The program is started up by the OS. Although the area in the main memory where a program is loaded is undefined, the address value corresponding to the label in the program is corrected to the actual address by the OS.
(3) During program startup, the OS allocates enough stack area for the program, then adds one to the last address and sets that value in the SP.
(4) The OS passes control to the program by the CALL instruction. When returning control to the OS after executing the program, the RET instruction is used.
(5) The assignment of an input device to the IN instruction or of an output device to the OUT instruction is made by the user before executing the program.
(6) The OS handles the differences that may arise in input and output procedures due to the different I/O devices and media involved; I/O is performed using the system's standard format and procedures (including error handling). Therefore, the user of these IN and OUT instructions does not need to be concerned with differences among I/O devices.

3.2 Undefined Items

Ensure that any items concerning program execution that are not defined in these specifications are handled by the processing system.

Các câu hỏi từ 1 đến 5 là bắt buộc. Trả lời mọi câu hỏi.

- Q1.** Hãy đọc đoạn văn bản sau liên quan đến việc thực hiện của một lệnh, rồi trả lời Câu hỏi con.

Một máy tính có dung lượng nhớ chính là 65.536 từ, mỗi từ gồm 16 bits. Nó có bốn thanh ghi vạn năng (từ 0 đến 3) và một thanh ghi chương trình (“PR”). Khuôn dạng của các từ lệnh (chiều dài 2 từ) như sau:

OP	R	X	I	D	adr
----	---	---	---	---	-----

- OP: Biểu diễn mã lệnh gồm 8 bits. Trong ví dụ này, ba mã lệnh sau được sử dụng.
 20_{16} : Đặt địa chỉ hiệu dụng trong thanh ghi vạn năng được biểu diễn bởi R.
 21_{16} : Đặt nội dung của từ được chỉ ra bởi địa chỉ hiệu dụng trong thanh ghi vạn năng được biểu diễn bởi R.
 FF_{16} : Kết thúc thực hiện
R: Biểu diễn số hiệu của thanh ghi vạn năng (từ 0 đến 3) bằng hai bit.
X: Biểu diễn số hiệu của thanh ghi chỉ số (từ 1 đến 3) bằng hai bit. Thanh ghi vạn năng tại số hiệu này được sử dụng làm thanh ghi chỉ số. Tuy nhiên, nếu X có giá trị "0", không có sự biến đổi nào theo thanh ghi chỉ số được thực hiện.
I: Có giá trị "1" biểu diễn bằng một bit trong trường hợp địa chỉ gián tiếp, trong trường hợp khác có giá trị "0".
D: Ba bit mở rộng, luôn có giá trị "0".
adr: Biểu diễn địa chỉ bằng 16 bits.

Địa chỉ hiệu dụng của từ lệnh được tính toán theo bảng sau.

Bảng Quan hệ giữa X, I, và địa chỉ hiệu dụng

X	I	Địa chỉ hiệu dụng
0	0	adr
1 đến 3	0	adr + (X)
0	1	(adr)
1 đến 3	1	(adr + (X))

Chú ý (): Ngoặc đơn được sử dụng để biểu diễn thông tin được lưu trong thanh ghi hoặc địa chỉ nằm trong ngoặc.

Câu hỏi con

Tù nhomial câu trả lời sau, chọn các câu trả lời đúng để điền vào các chỗ trống từ **a** đến **d** trong đoạn văn bản sau.

Khi các nội dung của các thanh ghi vạn năng và bộ nhớ chính có các giá trị nêu trong hình vẽ sau (các giá trị ở dạng HEX), 0100_{16} được đặt trong PR và chương trình được thực hiện. Trong ví dụ này, lệnh tại địa chỉ 0100_{16} đặt nội dung 0113_{16} của địa chỉ $011B_{16}$ (gạch dưới) vào thanh ghi vạn năng 0.

Sau khi kết thúc thực hiện, **a** được đặt vào thanh ghi vạn năng 0, **b** được đặt vào thanh ghi vạn năng 1, **c** được đặt vào thanh ghi vạn năng 2, và **d** được đặt vào thanh ghi vạn năng 3.

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

Thanh ghi chương trình PR : 0100

Bộ nhớ chính

Địa chỉ	+ 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	<u>0113</u>	0114	0115	0116	0117
0120	0118	0119	011A	011B	011C	011D	011E	011F

Hình. Các giá trị trong các thanh ghi vạn 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} |

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

- Q2.** Hãy đọc đoạn văn bản sau liên quan đến việc sử dụng các biểu thức, rồi trả lời các Câu hỏi con từ 1 đến 3.

Có một sổ cái (ledger) dùng để quản lý thông tin của tệp văn bản và đồ họa, như trong ví dụ ở bảng 1. Thông tin tệp bao gồm bốn tham số (tên tệp, kiểu mã, mã phiên bản, ngày tạo), và mỗi mục là một xâu ký tự. Việc tìm kiếm được thực hiện trên mỗi mục trong Sổ cái. Điều kiện tìm kiếm được xác định như các biểu thức.

Bảng 1 Ví dụ của Sổ cái

Tên tệp	Kiểu mã	Phiên bản mã	Ngày tạo
LOGO-T01	JPG	V/R100	2002-01-15
TITLE-A1	GIF	V/R100	2002-01-15
REP-JP01	HTML	V/R203	2002-01-22
OPINION3	TXT	V/R103	2003-02-05

- (1) Các xâu ký tự của tất cả các mục bao gồm các chữ cái hoa, số, gạch dưới, gạch nghiêng được xác định bởi tập ký tự mã trao đổi thông tin ASCII .
- (2) Bảng 2 dưới đây thể hiện các siêu ký tự được sử dụng trong biểu thức của các điều kiện tìm kiếm. Siêu ký tự (meta-character) là ký tự dùng để thể hiện cú pháp của một biểu thức.

Bảng 2 Các siêu ký tự dùng trong các biểu thức

Siêu ký tự	Ý nghĩa	Biểu thức ví dụ	Giải thích ví dụ
.	Thể hiện một ký tự bất kỳ	M..N	Xâu ký tự gồm 4 ký tự, bắt đầu với “M” và kết thúc với “N”
(Xâu ký tự)	Xâu ký tự được bao bởi các ngoặc đơn trái và phải, được coi như một mẫu (pattern).	(MO)	Mẫu bao gồm xâu ký tự “MO”
+	Thể hiện một hoặc lặp lại nhiều lần của ký tự hoặc mẫu đứng trước.	ABX+	AB và một hoặc lặp lại của nhiều lần của X (ví dụ, ABX, ABXX, ABXXX)
		(MO)+	Một hoặc lặp lại của nhiều lần của mẫu "MO" (ví dụ, MO, MOMO, MOMOMO)
*	Thể hiện 0 hoặc lặp lại nhiều lần của ký tự hoặc mẫu đứng trước.	ABX*	AB và 0 hoặc lặp lại nhiều lần của X (ví dụ., AB, ABX, ABXX)
?	Chỉ ra ký tự hoặc mẫu đứng trước xuất hiện 0 hoặc 1 lần.	ABCD?	ABC hoặc ABCD
\	Ký tự tiếp theo không phải là siêu ký tự, mà chỉ là ký tự thường	AB*	Xâu ký tự AB*
	Thể hiện lựa chọn ký tự hoặc mẫu	A B	A hoặc B
		(AB) (CDE)	Mẫu “AB” hoặc “CDE”
[m–n]	Thể hiện lựa chọn một ký tự bất kỳ trong nhóm các ký tự liên tiếp từ “m” đến “n”.	[3–5]	Ký tự bất kỳ (3, 4, hoặc 5) trong nhóm các ký tự liên tiếp.
		[W–Z]	Ký tự bất kỳ (W, X, Y, hoặc Z) trong nhóm các ký tự liên tiếp.

Câu hỏi con 1

Các mã phiên bản của tệp được ghi trong sổ cái theo cú pháp sau đây.

Mã phiên bản được bắt đầu với ba ký tự “V/R”, đi theo bởi một chữ số (từ 1 đến 9) thể hiện số hiệu của phiên bản và kết thúc với 2 chữ số (từ 00 đến 99) thể hiện số hiệu con của phiên bản. Ví dụ, nếu số hiệu phiên bản là 1 và số hiệu con là 03, thì mã phiên bản là “V/R103”.

Trong nhóm câu trả lời sau đây, hãy chọn câu trả lời **không đúng** cho việc tìm kiếm mã phiên bản dùng các biểu thức.

Nhóm câu trả lời:

- a) Sử dụng “01” để lấy ra các tệp có mã con là 01.
- b) Sử dụng “R.. 1” để lấy ra các tệp có mã con là 01.
- c) Sử dụng “R[1-3]” để lấy ra các tệp có số hiệu phiên bản là 3 hoặc nhỏ hơn.
- d) Sử dụng “302” để lấy ra tệp có số hiệu phiên bản và số hiệu con tương ứng là 3 và 02.
- e) Sử dụng “V/R302” để lấy ra tệp có số hiệu phiên bản và số hiệu con tương ứng là 3 và 02.

Câu hỏi con 2

Các mã kiểu (các xâu ký tự có 3 hoặc 4 ký tự) chỉ ra kiểu tệp được ghi trong sổ cái. Có tám mã kiểu các nhau, như sau.

GIF	HTM	HTML	JPG	JPEG	JPN	MPEG	TXT
-----	-----	------	-----	------	-----	------	-----

<http://www.vitec.org.vn>

Các tệp có mã kiểu là JPG hoặc JPEG cần được lấy ra. Trong nhóm các câu trả lời sau, hãy chọn câu trả lời là biểu thức chính xác được dùng để tìm kiếm các mã kiểu này.

Nhóm các câu trả lời:

- a) JPEG?
- b) JPEG*
- c) JPE?G
- d) JP+G
- e) JP?G
- f) JP.G

Câu hỏi con 3

Các ngày tạo tệp được ghi lại trong số cái. Ngày tạo là một xâu ký tự chứa bốn chữ số (từ 0001 đến 9999) thể hiện năm, đi theo bởi một số có hai chữ số (từ 01 đến 12) thể hiện tháng, tiếp theo là một số có hai chữ số (từ 01 đến 31) thể hiện ngày, với dấu gạch ngang để nối các số này với nhau. Ví dụ, ngày 20 tháng 3 năm 2003 được thể hiện bởi “2003-03-20”.

Việc tìm kiếm ngày tạo tệp được thực hiện bằng cách dùng biểu thức dưới đây. Trong nhóm các câu trả lời sau, chọn ra tất cả các ngày được lấy ra bằng cách tìm kiếm sử dụng biểu thức

..(0|1|2)\-)+.1

- Nhóm câu trả lời:
- a) 2001-02-10 b) 2001-12-11 c) 2002-02-21
d) 2002-11-10 e) 2002-12-01 f) 2003-01-10



<http://www.vitec.org.vn>

- Q3.** Hãy đọc đoạn văn bản sau liên quan đến điều khiển truy cập LAN, rồi trả lời câu hỏi con.

CSMA/CD (Carrier Sense Multiple Access/Collision Detect) là hệ thống điều khiển truy cập được sử dụng trong các mạng LAN kiểu bus sử dụng cáp đồng trục (coaxial cable), và trong các mạng LAN hình sao sử dụng cáp 2 dây xoắn và hubs.

Với CSMA/CD, thủ tục sau được sử dụng trong truyền khung (frame transmission) để nhận biết tín hiệu mang và phát hiện xung đột giữa các thiết bị được nối tới đường truyền.

[Mô tả thủ tục truyền khung]

Bước 0: Chờ cho đến khi có một khung được truyền. Khi có một khung được truyền, nhảy tới bước 1.

Bước 1: Nếu tín hiệu mang của thiết bị khác trong mạng được nhận biết trên đường truyền, nhảy đến bước 2, nếu không thì nhảy tới bước 3.

Bước 2: Khi chu kỳ thời gian xác định ngẫu nhiên đã trôi qua, quay lại bước 1.

Bước 3: Bắt đầu truyền khung và nhảy thẳng tới bước 4.

Bước 4: Nếu không có xung đột khi truyền khung, việc truyền coi như thành công.

Quay lại bước 0. Nếu phát hiện xung đột trong quá trình truyền khung, nhảy tới bước 5. Việc kiểm tra này được thực hiện nhằm xác định xem có xung đột trong quá trình truyền khung hay không, bởi vì khả năng là sự bắt đầu truyền khung từ một thiết bị khác trong mạng đã không được phát hiện trong bước 1 do độ trễ lan truyền tín hiệu trên đường truyền.

Bước 5: Chuyển từ khung đang được truyền sang tín hiệu thông báo cho các thiết bị khác về sự xuất hiện xung đột. Sau khi truyền tín hiệu này trong khoảng thời gian xác định, quay lại bước 2.

Tín hiệu này cho phép các thiết bị khác trong mạng biết rằng xung đột đã xảy ra.

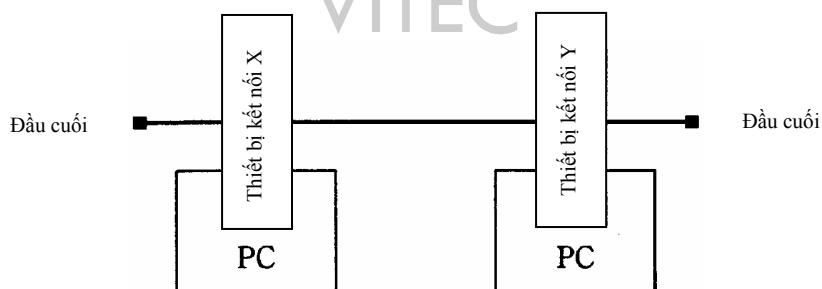
Câu hỏi con

Tùy nhóm các câu trả lời dưới đây, chọn các câu trả lời đúng để điền vào các chỗ trống từ **a** đến **e** trong đoạn văn bản sau.

Có một mạng LAN được nối như trong sơ đồ sau. Khoảng cách đường truyền giữa thiết bị kết nối X và thiết bị kết nối Y là 230 m. Tốc độ lan truyền tín hiệu trên đường truyền là 230 m trong một micro giây.

Việc truyền khung từ thiết bị X được bắt đầu. Trước khi trôi qua **a** micro giây từ lúc bắt đầu truyền, một khung đang được truyền từ thiết bị Y. Khi điều này xảy ra, thiết bị Y chuyển qua bước 1 và nhảy tới **b**. Nhờ đó, thiết bị Y nhận biết được xung đột.

Khoảng thời gian cần có tối đa cho thiết bị X để nhận biết xung đột là xấp xỉ **c** micro giây sau khi bắt đầu truyền khung từ thiết bị X, phụ thuộc vào khoảng thời gian chênh lệch so với thời điểm bắt đầu truyền từ thiết bị Y, là thiết bị bắt đầu truyền sau thiết bị X. Nếu việc truyền từ thiết bị X kết thúc trước khi khoảng thời gian này trôi qua, thì thiết bị X sẽ đi từ **d** đến **e** và sẽ không thể phát hiện ra xung đột. Trong trường hợp này, giả thiết rằng các thiết bị kết nối có thể gắn vào bất kỳ vị trí nào ở khoảng giữa các thiết bị đầu cuối lắp tại cả hai đầu, thì rõ ràng là với phương pháp nhận biết này, thời gian cần có để truyền một khung đơn (single frame) ít nhất cũng phải dài bằng thời gian tín hiệu đi một vòng giữa các thiết bị đầu cuối trên đường truyền.



Hình Kết nối mạng LAN Connections bằng phương pháp CSMA/CD

Nhóm câu trả lời cho a và c:

- | | | |
|-----------|--------|---------|
| a) 0.0043 | b) 0.2 | c) 1 |
| d) 2 | e) 10 | f) 4300 |

Nhóm câu trả lời cho b, d, và e:

a) Bước 0
d) Bước 3

b) Bước 1
e) Bước 4

c) Bước 2
f) Bước 5

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

- Q4.** Hãy đọc mô tả chương trình, mô tả cú pháp giả ngữ, và chương trình dưới đây rồi trả lời câu hỏi con.

[Mô tả chương trình]

`Calc` là chương trình con sử dụng một ngăn xếp để tính các biểu thức số được biểu diễn theo cách ghi Polish đảo (Reverse Polish Notation).

- (1) Các biểu thức số được biểu diễn theo cách ghi Polish đảo được lưu giữ, mỗi lần một ký tự, trong các phần tử riêng biệt $\text{Ex}[0], \dots, \text{Ex}[\text{Lp}]$ ($\text{Lp} \geq 3$) của mảng một chiều kiểu ký tự `Ex`.
- (2) Biểu thức số gồm số nguyên dương hoặc nguyên âm và một hoặc nhiều ký hiệu toán tử số học. Nếu số nguyên là dương, thì dấu cộng không được thêm vào nó.
- (3) Một dấu cách (trống) được đặt trước các số nguyên, trừ số nguyên đầu tiên.
- (4) Các tính toán được thực hiện bằng cách dùng các số thực. Chương trình con `Abort()` được gọi để thoát khỏi chương trình nếu một trong hai trạng thái sau xảy ra trong quá trình tính toán.
 - ① Thực hiện việc chia cho 0.
 - ② Có gì đó bên ngoài ngăn xếp được tham chiếu đến.
- (5) `Calc` sử dụng chương trình con `Push`, chương trình này thêm các số thực vào ngăn xếp, và chương trình con `Pop`, chương trình này lấy các số thực ra khỏi ngăn xếp. Các bảng 1 đến 3 dưới đây cho thấy đặc tả của các đối số đối với mỗi chương trình con. Ngoài ra, hàm `ToReal` cũng được sử dụng, hàm này chuyển đổi một ký tự số đơn thành một số thực.
- (6) Các biến sau đây được định nghĩa như các biến toàn cục (global variables): `Stack`, mảng một chiều kiểu số thực; `MAX`, một hằng diễn tả số lượng phần tử lớn nhất trong `Stack`; và `sp`, một biến cho biết vị trí đang được xử lý trong ngăn xếp. Giá trị ban đầu của `sp` là “0”.
- (7) Giả thiết rằng các biểu thức số được diễn tả theo cách ghi Polish đảo là đúng.

Ví dụ: Biểu thức số trong cách
ghi trung tố

	0	1	2	3	4	5	6	7	8	9	
(1 + 2) * (3 - 4)	Ex	1	△	2	+	△	3	△	4	-	*
12 / (- 345)	Ex	0	1	2	3	4	5	6	7		Lp = 9

	0	1	2	3	4	5	6	7	8	9	
(1 + 2) * (3 - 4)	Ex	1	△	2	+	△	3	△	4	-	*
12 / (- 345)	Ex	0	1	2	3	4	5	6	7		Lp = 7

Ghi chú: Các hình tam giác ký hiệu cho các dấu cách (trống).

Bảng 1 Đặc tả đối số của Calc

Tên biến	Vào/ra	Ý nghĩa
Ex []	Vào	Mảng một chiều kiểu ký tự chưa biểu thức số được diễn tả theo cách ghi Polish đảo
Lp	Vào	Số hiệu của phần tử cuối cùng trong mảng một chiều kiểu ký tự Ex ($L_p \geq 3$)
Ret	Ra	Kết quả tính toán

Bảng 2 Đặc tả đối số của Push

Tên biến	Vào/ra	Ý nghĩa
T	Vào	Số thực được thêm vào ngăn xếp

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

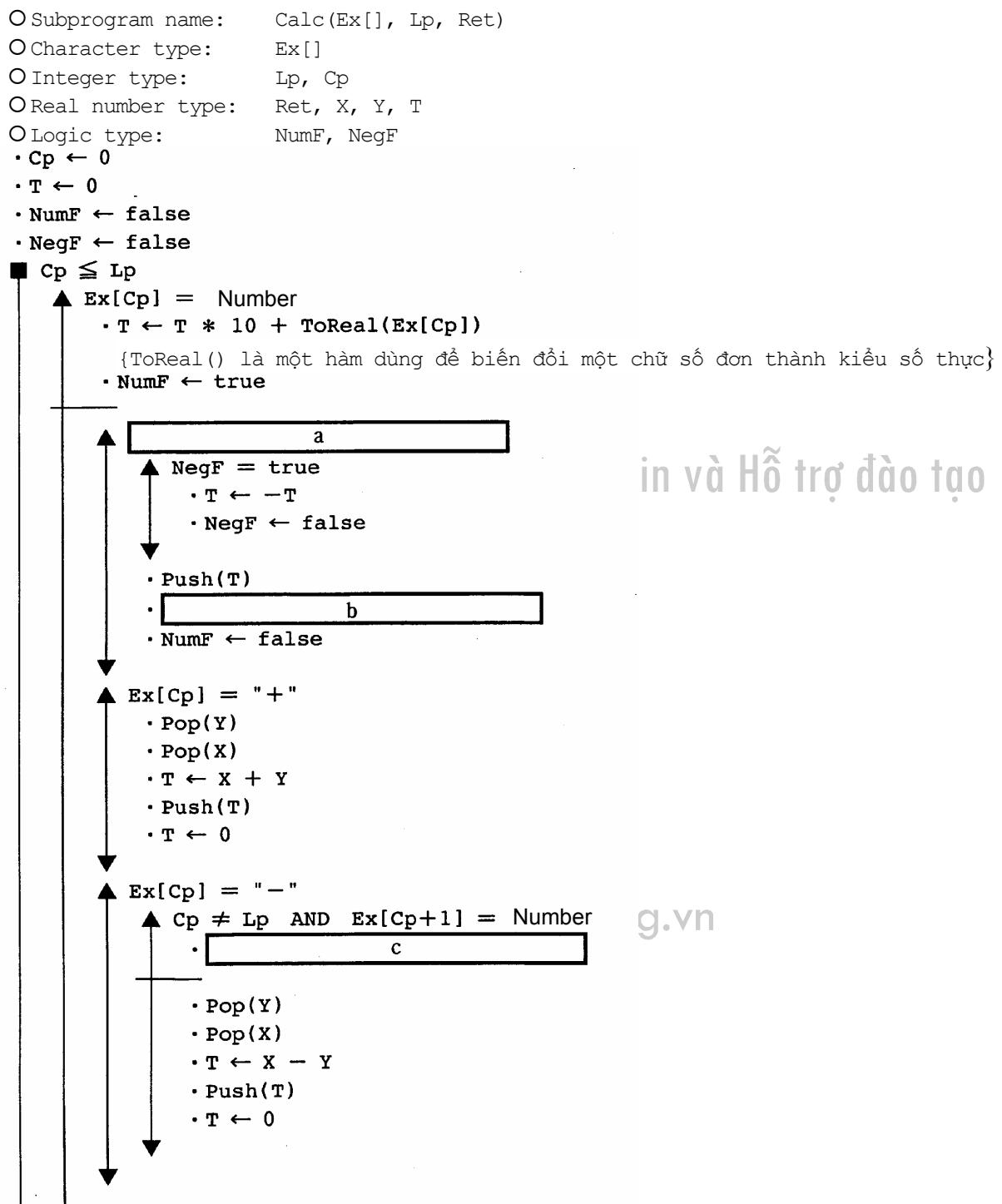
Bảng 2 Đặc tả đối số của Pop

Tên biến	Vào/ra	Ý nghĩa
T	Ra	Số thực được lấy ra khỏi ngăn xếp

[Mô tả cú pháp giả ngữ]

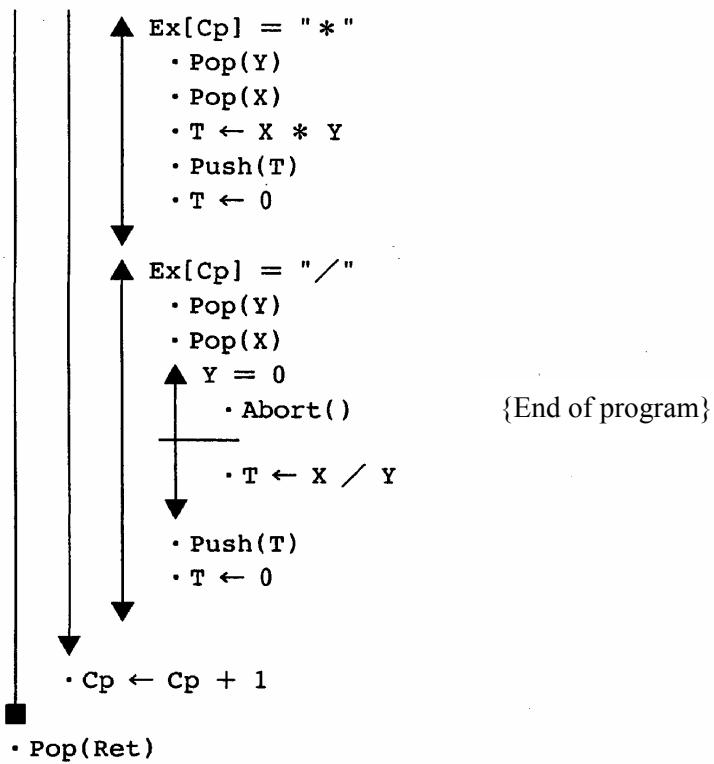
Cú pháp	Mô tả
○	Khai báo tên, kiểu, ... của các thủ tục, biến, ...
• Biến \leftarrow Biểu thức	Gán giá trị của Biểu thức vào Biến.
{Văn bản}	Văn bản là chú thích.
↑ Biểu thức điều kiện • Quy trình 1 — • Quy trình 2 ↓	Chỉ ra một quy trình chọn lựa. Nếu biểu thức điều kiện là TRUE, thì Quy trình 1 được thực hiện; nếu nó là FALSE, thì Quy trình 2 được thực hiện.
█ Biểu thức điều kiện • Quy trình	Chỉ một vòng lặp với điều kiện kết thúc đặt lên đầu. Nếu biểu thức điều kiện là TRUE, thì Quy trình được thực hiện.

[Chương trình]

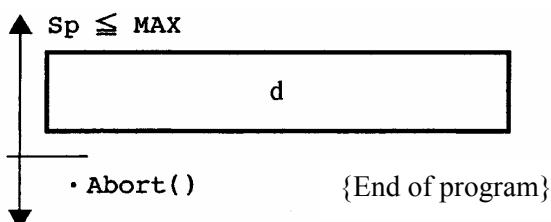


in và Hỗ trợ đào tạo

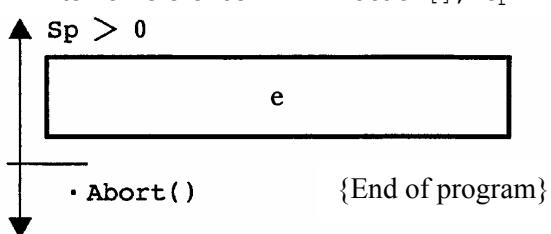
g.vn



- Subprogram name: Push(T)
- Real number type: T
- External reference: Stack[], Sp



- Subprogram name: Pop(T)
- Real number type: T
- External reference: Stack[], Sp



in và Hỗ trợ đào tạo

g.vn

Câu hỏi con

Tùy các nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào các chỗ trống từ **a** đến **e** trong chương trình trên.

Nhóm câu trả lời cho a:

- a) **Ex[Cp] = " "** b) **NumF = false** c) **NumF = true**
d) **T = 0** e) **T ≠ 0**

Nhóm câu trả lời cho b và c:

- a) **NegF ← false** b) **NegF ← true** c) **NumF ← false**
d) **NumF ← true** e) **T ← 0**

Nhóm câu trả lời cho d và e:

- a) • **Sp ← Sp + 1**
• **Stack[Sp] ← T**
- b) • **Sp ← Sp + 1**
• **T ← Stack[Sp]**
- c) • **Sp ← Sp - 1**
• **Stack[Sp] ← T**
- d) • **Sp ← Sp - 1**
• **T ← Stack[Sp]**
- e) • **Stack[Sp] ← T**
• **Sp ← Sp + 1**
- f) • **Stack[Sp] ← T**
• **Sp ← Sp - 1**
- g) • **T ← Stack[Sp]**
• **Sp ← Sp + 1**
- h) • **T ← Stack[Sp]**
• **Sp ← Sp - 1**

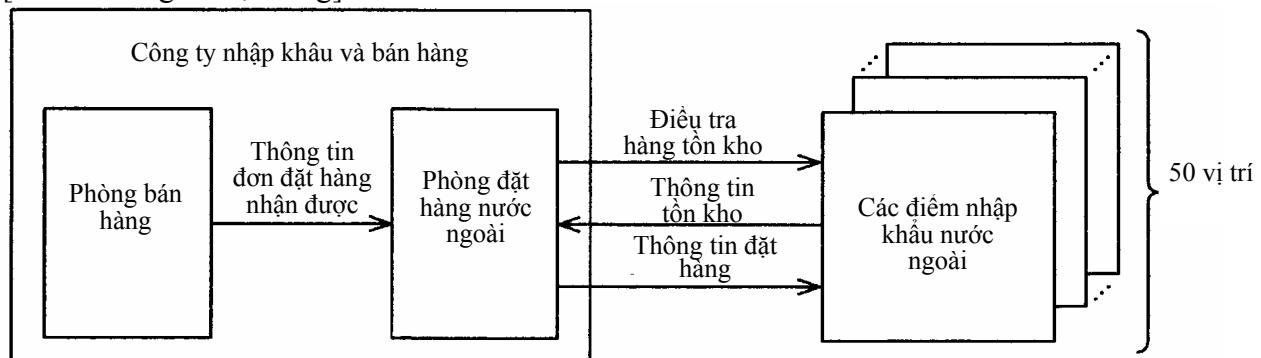


<http://www.vitec.org.vn>

- Q5.** Hãy đọc văn bản sau về thiết kế chương trình, sau đó trả lời Câu hỏi con 1 và 2.

Một hệ thống phân công nhập hàng đang được thiết kế cho phòng đặt hàng nước ngoài của một công ty nhập khẩu và bán hàng. Khái quát về hệ thống như sau.

[Mô tả chung về hệ thống]



Trung

- (1) Mã hàng được đặt, số lượng đặt, và ngày giao hàng mà phòng bán hàng mong muốn, được nhận trực tuyến từ phòng bán hàng. Thông tin đơn hàng có định dạng như sau.

Mã hàng	Số lượng yêu cầu	Ngày giao hàng mong muốn		
		Năm	Tháng	Ngày

- (2) Dựa theo mã hàng trong thông tin đơn hàng, hệ thống phân công nhập hàng của phòng đặt hàng nước ngoài gửi mã hàng tới các điểm nhập khẩu có mặt hàng đó và yêu cầu cho biết số lượng hàng còn trong kho. Có 50 điểm nhập khẩu ở trên toàn thế giới, và chúng được kết nối trực tuyến với phòng đặt hàng nước ngoài. Các sản phẩm được tích trữ tùy thuộc vào từng điểm nhập khẩu cụ thể. Trong một số trường hợp, nhiều điểm nhập khẩu tích trữ cùng một loại hàng.

- (3) Để phản hồi, các điểm nhập khẩu gửi ngay về phòng đặt hàng nước ngoài các thông tin tồn kho về mặt hàng yêu cầu. Nếu không có hàng trong kho, thì câu trả lời được gửi lại với số lượng hàng trong kho bằng 0. Phòng đặt hàng nước ngoài ghi lại các thông tin tồn kho nhận được vào một tệp thông tin tồn kho. Thông tin tồn kho có định dạng sau.

Mã điểm nhập khẩu	Mã hàng	Số lượng trong kho

(4) Có hai cách để giao hàng từ điểm nhập khẩu tới phòng bán hàng: bằng tàu thuỷ và bằng máy bay. Số ngày cần để giao hàng cho phòng bán hàng (được gọi là “số ngày để giao hàng”) từ một điểm nhập khẩu đã cho thay đổi phù thuộc vào việc vận chuyển bằng đường biển hay đường hàng không. Số ngày để giao hàng cũng thay đổi phụ thuộc vào điểm nhập khẩu cụ thể.

Số ngày để giao hàng đối với mỗi điểm nhập khẩu được lưu trong tệp chủ về các điểm nhập khẩu. Tệp này có định dạng sau.

Mã điểm nhập khẩu	Tên điểm nhập khẩu	Số ngày để giao hàng bằng tàu thuỷ	Số ngày để giao hàng bằng máy bay	Thông tin khác
-------------------	--------------------	------------------------------------	-----------------------------------	----------------

(5) Quyết định về việc có thể nhập hàng hay không được đưa ra và số lượng hàng nhập từ mỗi điểm được xác định trên cơ sở thông tin tồn kho từ các điểm nhập

khẩu và số ngày để giao hàng bằng tàu thuỷ hay bằng máy bay từ các điểm đó. Cụ thể, trước tiên phải kiểm tra để xác định những điểm nào có mặt hàng yêu cầu theo kiểm kê, và có thể đáp ứng được ngày giao hàng mong muốn. Tiếp theo, sẽ phân công các điểm nhập khẩu mà các đơn hàng sẽ được gửi tới. Nếu một điểm không đáp ứng được số lượng hàng cần nhập thì sẽ phân công cho nhiều điểm. Thủ tục xử lý để làm việc đó được mô tả trong biểu đồ luồng dưới đây.

Các điểm nhập khẩu được phân công theo các mức ưu tiên từ ① đến ③.

① Ưu tiên cao hơn cho các điểm nhập khẩu có thể đáp ứng được thời hạn giao hàng mong muốn qua đường tàu thuỷ so với các điểm đáp ứng được thời hạn giao hàng qua đường hàng không.

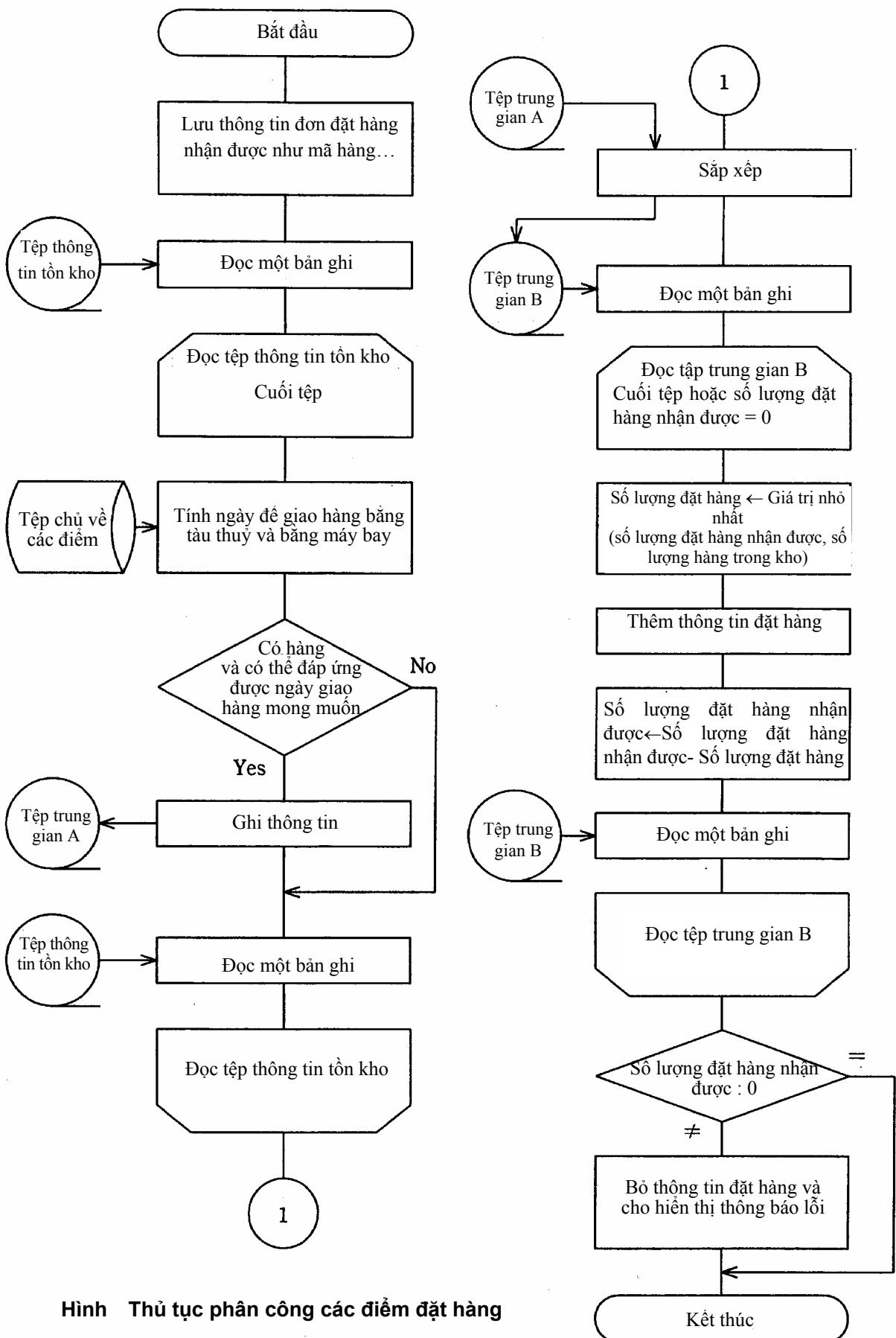
② Ưu tiên cho các điểm nhập khẩu có số lượng mặt hàng yêu cầu lớn hơn.

③ Ưu tiên cho các điểm nhập khẩu có mã điểm nhập khẩu nhỏ hơn.

Nếu không thể nhận được số hàng yêu cầu vào ngày giao hàng mong muốn, một thông báo về việc đó sẽ được đưa ra.

(6) Một khi các điểm nhập khẩu, số lượng hàng được đặt tương ứng tại các điểm đó, và phương tiện giao hàng đã được xác định, thông tin đặt hàng được gửi tới các điểm nhập khẩu liên quan. Thông tin đặt hàng có định dạng như sau. Trong định dạng, phương tiện giao hàng được lưu là một trong hai phương tiện tàu thuỷ hoặc máy bay.

Mã điểm nhập khẩu	Mã hàng	Số lượng đặt	Phương tiện giao hàng
-------------------	---------	--------------	-----------------------



Câu hỏi con 1

Tù nhom câu trả lời dưới đây, hãy chọn ba câu trả lời đúng cho các mục dữ liệu cần phải đưa vào tệp trung gian A được chỉ ra trong biểu đồ luồng ở trên .

Nhom câu trả lời:

- a) Ngày giao hàng mong muốn
- b) Mã điểm nhập khẩu
- c) Số lượng hàng trong kho
- d) Mã hàng
- e) Phương tiện giao hàng
- f) Số ngày để giao hàng
- g) Số lượng đặt hàng

Câu hỏi con 2

Hệ thống này cần được kiểm thử. Trước tiên, hãy xem các dữ liệu kiểm thử cho thông tin đơn đặt hàng nhận được.

Mã hàng	Số lượng đặt	Ngày giao hàng mong muốn		
		Năm	Tháng	Ngày
12345	200	2003	05	10

Giả thiết rằng, việc so sánh ngày giao hàng mong muốn với ngày nhận đơn hàng cho thấy có 20 ngày dự trữ cho số ngày để giao hàng. Như được chỉ ra trong bảng sau, có 7 mẫu khác nhau đã được thiết lập làm dữ liệu kiểm thử cho số lượng hàng trong kho và số ngày để giao hàng tại các điểm nhập khẩu có mặt hàng đó. Từ nhom câu trả lời sau, hãy chọn các câu trả lời đúng để điền vào các ô trống từ a đến c trong bảng sao cho thoả mãn các kết quả mong đợi được chỉ ra ở trong đó.

Bảng Chi tiết về các mẫu kiểm thử và các kết quả mong đợi

Mẫu thử	Mã điểm	Lượng hàng trong kho	Số ngày để giao hàng bằng máy bay	Số ngày để giao hàng bằng tàu thuỷ	Các kết quả mong đợi
1	01	500	10	20	Đặt 200 đơn vị hàng bằng tàu thuỷ tại điểm nhập khẩu có mã là 01.
2	02	300	8	21	Đặt 200 đơn vị hàng bằng máy bay tại điểm nhập khẩu có mã là 02.
3	03	500	21	21	Thông báo lỗi được hiển thị.
4	04	100	6	19	Thông báo lỗi được hiển thị.
	05	50	7	21	
5	06	70	7	20	Đặt 70 đơn vị hàng bằng tàu thuỷ tại điểm nhập khẩu có mã là 06; 70 đơn vị hàng bằng máy bay tại điểm nhập khẩu có mã là 07; và 60 đơn vị hàng bằng máy bay tại điểm nhập khẩu có mã là 08.
	07	70	12	30	
	08	a	8	21	
6	09	150	2	21	Đặt 150 đơn vị hàng bằng máy bay tại điểm nhập khẩu có mã là 09; và 50 đơn vị hàng bằng máy bay tại điểm nhập khẩu có mã là 10.
	10	b	3	22	
	11	120	4	23	
7	12	c	20	40	Đặt 30 đơn vị hàng bằng máy bay tại điểm nhập khẩu với mã 12; 160 đơn vị hàng bằng tàu thuỷ tại điểm nhập khẩu có mã là 13; và 10 đơn vị hàng bằng máy bay tại điểm nhập khẩu có mã là 14.
	13	160	10	20	
	14	30	15	25	

Nhóm câu trả lời:

- | | | |
|--------|--------|--------|
| a) 20 | b) 30 | c) 50 |
| d) 70 | e) 100 | f) 120 |
| g) 160 | | |

Chọn một trong bốn câu hỏi sau (Q6, Q7, Q8, hoặc Q9) để trả lời. Chú ý phải tô đen vào (S) trong cột Selection Column trên phiếu trả lời đối với câu hỏi mà bạn chọn trả lời. Nếu bạn chọn trả lời nhiều hơn một câu hỏi, thì chỉ có câu trả lời đầu tiên được chấm điểm.

- Q6.** Hãy đọc mô tả sau về một chương trình C và đọc chương trình đó, sau đó trả lời các câu hỏi con 1 và 2.

[Mô tả chương trình]

(1) Chương trình này tạo ra đồ thị có tính cân đối bằng cách sử dụng một mảng hai chiều. Chương trình cho ra đồ thị fractal (được gọi là Sierpinski gasket) được chỉ ra trong hình dưới đây từ các giá trị ban đầu trong dòng 0 của bảng dưới đây. Trong ví dụ này, mảng có 33 dòng và cột.



Hình: Kết quả đưa ra

		Cột Bảng Các giá trị của trạng thái được tạo ra											
Dòng	Các giá trị ban đầu	0	1	2	3	4	5	6	7	8	9	...	32
0	0	0	1	0	0	0	0	0	0	0	0	...	0
1	0	1	1	0	0	0	0	0	0	0	0	...	0
2	0	1	0	1	0	0	0	0	0	0	0	...	0
3	0	1	1	1	1	0	0	0	0	0	0	...	0
4	0	1	0	0	0	1	0	0	0	0	0	...	0
5	0	1	1	0	0	1	1	0	0	0	0	...	0
6	0	1	0	1	0	1	0	1	0	0	0	...	0
7	0	1	1	1	1	1	1	1	1	0	0	...	0
8	0	1	0	0	0	0	0	0	0	0	1	...	0
9	0	1	1	0	0	0	0	0	0	0	1	...	0
:	:	:	:	:	:	:	:	:	:	:	:	⋮	⋮
31	0	1	1	1	1	1	1	1	1	1	1	...	1
32	0	1	0	0	0	0	0	0	0	0	0	...	0

Trung tam Sát hạch Công nghệ thông tin và Ho trợ đào tạo

(2) Các giá trị chỉ trạng thái (0 hoặc 1) được lưu trong mảng hai chiều s. Các giá trị trạng thái trong dòng 1 và các dòng sau đó được tạo ra theo các quy tắc dưới đây từ các giá trị ban đầu được lưu trong dòng 0, như ta thấy ở bảng trên.

- ① Giá trị trạng thái của dòng i , cột j được biểu diễn bằng $s[i][j]$.
- ② Mọi giá trị trong cột 0 đều bằng 0.
- ③ Các giá trị $s[i+1][j]$ (trong đó $j \geq 1$) được tạo ra theo bảng sau, dựa trên các giá trị ban đầu của $s[i][j-1]$ and $s[i][j]$.

Các giá trị trạng thái dùng		Giá trị trạng thái được tạo ra cho
$s[i][j-1]$	$s[i][j]$	$s[i+1][j]$
1	1	0
1	0	1
0	1	1
0	0	0

(3) Các giá trị của cột 0, dòng 32 trong mảng hai chiều s không phải là đầu ra.

(Program)

```
#include <stdio.h>
#define ALIVE    1
#define DEAD     0
#define SZ       33

int stschk( int, int );

main()
{
    int s[SZ][SZ], i, j;
    for ( i=0; i<SZ; i++ ) s[i][0] = DEAD;
    for ( j=2; j<SZ; j++ ) s[0][j] = DEAD;
    s[0][1] = ALIVE;
    for ( i=0; i<SZ-1; i++ ) {
        for ( j=1; j<SZ; j++ ) {
            [a] = stschk( [b], [c] );
            if ( [b] == ALIVE ) printf( "*" );
            else                  printf( " " );
        }
        printf( "\n" );
    }
}

int stschk( int s1, int s2 )
{
    if ((( s1==DEAD )&&( s2==ALIVE ))|||  

        (( s1==ALIVE )&&( s2==DEAD ))) return ALIVE;  

    else return DEAD;
}
```

Trung

đào tạo

Câu hỏi con 1

Từ nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để chèn vào các ô trống từ **[a]** đến **[c]** trong chương trình trên.

<http://www.vitec.org.vn>

Nhóm câu trả lời:

- | | | |
|----------------|--------------|----------------|
| a) s[i-1][j-1] | b) s[i-1][j] | c) s[i-1][j+1] |
| d) s[i][j-1] | e) s[i][j] | f) s[i][j+1] |
| g) s[i+1][j-1] | h) s[i+1][j] | i) s[i+1][j+1] |

Câu hỏi con 2

Đồ thị mà ta thấy dưới đây sẽ được đưa ra nếu mệnh đề if trong hàm `stschk` bị thay đổi. Từ nhóm câu trả lời dưới đây, hãy chọn mệnh đề if dẫn đến kết quả này.

A large, dense pattern of black asterisks (*) forming a triangular shape. The pattern is composed of many smaller triangles pointing upwards, creating a larger overall triangle. The density of the asterisks increases towards the bottom right corner of the image.

Trung tâm Sát hạch và Hỗ trợ đào tạo

Nhóm câu trả lời:

- a) if ((s1==ALIVE) || (s2==ALIVE)) return ALIVE;
 else return DEAD;

b) if ((s1==ALIVE) && (s2==ALIVE)) return ALIVE;
 else return DEAD;

c) if ((s1==DEAD) || (s2==DEAD)) return DEAD;
 else return ALIVE;

d) if (((s1==ALIVE)&&(s2==DEAD))||
 ((s1==DEAD)&&(s2==ALIVE))) return ALIVE;
 else return DEAD;

- Q7.** Hãy đọc mô tả sau về chương trình COBOL và đọc chương trình đó, rồi trả lời câu hỏi sau.

[Mô tả chương trình]

Chương trình này lấy ra các tổng số bản ghi lỗi phân theo mã lỗi, từ một tệp tình trạng sửa chữa ghi lại các tình trạng sửa chữa máy PC. Ngoài ra, nó in ra các tổng số bản ghi theo mã lỗi, cũng như tỷ lệ phần trăm tương ứng.

(1) Định dạng bản ghi trong tệp tình trạng sửa chữa như sau.

Số thứ tự 6 chữ số	Xảy ra ngày 8 chữ số	Mã lỗi (fault code) 3 chữ số	Thông tin khác 26 chữ số
-----------------------	-------------------------	---------------------------------	-----------------------------

(2) Định dạng in như sau.

FAULT CODE	COUNT	PERCENTAGE
XXX	z,zz9	zz9.9
XXX	z,zz9	zz9.9
XXX	z,zz9	zz9.9
:	:	:
XXX	z,zz9	zz9.9

- ① Các tổng số bản ghi theo mã lỗi có giá trị bằng hoặc nhỏ hơn 9,999.
- ② Các giá trị phần trăm được tính bằng cách chia tổng số bản ghi lỗi của từng mã lỗi cho tổng số bản ghi có trong tệp tình trạng sửa chữa. Các giá trị này chỉ lấy đến một chữ số ở phần thập phân.
- ③ Đầu đề được in một lần đầu tiên.

[Chương trình]

```

DATA DIVISION.
FILE SECTION.
FD REPAIR-FILE.
01 REPAIR-REC.
  02 ORDER-REP          PIC X(06).
  02 DATE-REP           PIC X(08).
  02 FAULT-REP          PIC X(03).
  02 EST-REP            PIC X(26).
FD PRINT-FILE.
01 PRINT-REC           PIC X(29).

```

```

SD  SORT-FILE.
01  SORT-REC          PIC  X(03).
WORKING-STORAGE SECTION.
01  END-ST            PIC  9(01).
01  COUNT-T           PIC  9(07).
01  FAULT-WRK         PIC  X(03).
01  COUNT-WRK         PIC  9(04).
01  TOP-HEADER        PIC  X(29) VALUE
                      "FAULT CODE  COUNT  PERCENTAGE".
01  DATA-PRN.
    02  FAULT-PRN       PIC  X(03).
    02                   PIC  X(09) VALUE SPACE.
    02  COUNT-PRN        PIC  Z,ZZ9.
    02                   PIC  X(07) VALUE SPACE.
    02  PER-PRN          PIC  ZZ9.9.

PROCEDURE DIVISION.
SORT-RTN.
    SORT SORT-FILE
    ON ASCENDING KEY SORT-REC
        INPUT PROCEDURE IS SELECT-RTN
        OUTPUT PROCEDURE IS TOTAL-RTN.

STOP RUN.

SELECT-RTN.
    OPEN INPUT REPAIR-FILE.
    INITIALIZE END-ST COUNT-T.
    PERFORM UNTIL END-ST = 1
        READ REPAIR-FILE AT END
        MOVE 1 TO END-ST
        NOT AT END
            RELEASE SORT-REC FROM FAULT-REP
            COMPUTE COUNT-T = COUNT-T + 1
        END-READ
    END-PERFORM.
    CLOSE REPAIR-FILE.

TOTAL-RTN.
    OPEN OUTPUT PRINT-FILE.
    WRITE PRINT-REC FROM TOP-HEADER AFTER PAGE.
    INITIALIZE END-ST.
    RETURN SORT-FILE AT END
    MOVE 1 TO END-ST
    NOT AT END
        PERFORM INIT-RTN
    END-RETURN.
    PERFORM UNTIL END-ST = 1
        RETURN SORT-FILE AT END
        PERFORM PRINT-RTN
        MOVE 1 TO END-ST
        NOT AT END
            IF [ ] a
                COMPUTE COUNT-WRK = COUNT-WRK + 1
            ELSE
                PERFORM PRINT-RTN
            [ ] b
        END-IF

```

```

        END-RETURN
        END-PERFORM.
        CLOSE PRINT-FILE.
PRINT-RTN.
        MOVE COUNT-WRK TO COUNT-PRN.
        [ ] .  

        C
MOVE FAULT-WRK TO FAULT-PRN.
        [ ] .
        d
INIT-RTN.
        MOVE 1 TO COUNT-WRK.
        [ ] .
        e

```

Câu hỏi con

Tùy các nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào các ô trống từ **a** đến **e** trong chương trình trên.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Nhóm câu trả cho a:

- | | |
|------------------------------|-----------------------------|
| a) FAULT-WRK = FAULT-REP | b) FAULT-WRK = SORT-REC |
| c) FAULT-WRK > FAULT-REP | d) FAULT-WRK > SORT-REC |
| e) FAULT-WRK NOT = FAULT-REP | f) FAULT-WRK NOT = SORT-REC |

Nhóm câu trả cho b và e:

- | | |
|--------------------------------|--|
| a) MOVE FAULT-REP TO FAULT-WRK | |
| b) MOVE FAULT-WRK TO FAULT-PRN | |
| c) MOVE FAULT-WRK TO FAULT-REP | |
| d) MOVE FAULT-WRK TO SORT-REC | |
| e) MOVE SORT-REC TO FAULT-PRN | |
| f) MOVE SORT-REC TO FAULT-WRK | |
| g) PERFORM INIT-RTN | |
| h) PERFORM PRINT-RTN | |
| i) PERFORM SELECT-RTN | |
| j) PERFORM TOTAL-RTN | |

Nhóm câu trả lời cho c:

- a) COMPUTE PER-PRN = 100 * COUNT-T / COUNT-WRK
- b) COMPUTE PER-PRN = 100 * COUNT-WRK / COUNT-T
- c) COMPUTE PER-PRN = COUNT-T / COUNT-WRK
- d) COMPUTE PER-PRN = COUNT-T / COUNT-WRK / 100
- e) COMPUTE PER-PRN = COUNT-WRK / COUNT-T
- f) COMPUTE PER-PRN = COUNT-WRK / COUNT-T / 100
- g) MOVE COUNT-T TO PER-PRN
- h) MOVE COUNT-WRK TO PER-PRN

Nhóm câu trả lời cho d:

- a) WRITE PRINT-REC
- b) WRITE PRINT-REC AFTER 1
- c) WRITE PRINT-REC FROM DATA-PRN AFTER 1
- d) WRITE PRINT-REC FROM TOP-HEADER AFTER 1



<http://www.vitec.org.vn>

- Q8.** Hãy đọc mô tả sau về chương trình Java và đọc chương trình đó, sau đó trả lời câu hỏi con.

[Mô tả chương trình]

Chương trình này tính diện tích của hình và đưa ra kết quả. Hình (tam giác, chữ nhật hoặc vuông) được xác định trong chương trình như một đối tượng (object) hình với các thuộc tính như sau.

Tam giác: Độ dài ba cạnh

Chữ nhật: Độ dài hai cạnh (dài và rộng)

Vuông: Độ dài cạnh

Chương trình này gồm có 5 lớp sau.

AreaTest

Lớp này có phương thức main, và thực hiện các quy trình sau:

- (1) Nó xây dựng các đối tượng tam giác, chữ nhật, và vuông, và đặt chúng vào trong mảng figures.
- (2) Nó tính diện tích của mỗi hình và đưa ra kết quả. Trong trường hợp này, giả thiết rằng các giá trị số được cung cấp cho phép dựng được hình đúng.

Figure

Lớp này là một lớp trừu tượng của các hình. Nó khai báo phương thức trừu tượng getArea, tính diện tích và đưa ra kết quả.

Triangle

Lớp này là lớp tam giác. Nó định nghĩa phương thức toString, phương thức này trả về một thuộc tính dưới dạng một chuỗi ký tự; và phương thức getArea, phương thức này tính diện tích hình tam giác theo công thức Heron và đưa ra kết quả.

Rectangle

Lớp này là lớp hình chữ nhật. Nó định nghĩa phương thức toString, phương thức này trả về một thuộc tính dưới dạng một chuỗi ký tự; và phương thức getArea, phương thức này tính diện tích hình chữ nhật và đưa ra kết quả.

Square

Lớp này là lớp hình vuông. Nó định nghĩa phương thức toString, phương thức này trả về một thuộc tính dưới dạng một chuỗi ký tự;

Việc thực hiện Program 1 cho kết quả sau đây.

```
Triangle : sides = 2.0, 3.0, 3.0 : area = 2.8284271247461903  
Rectangle : height = 5.0, width = 8.0 : area = 40.0  
Square : width = 5.0 : area = 25.0
```

Hình: Kết quả thực hiện

[Program 1]

```
public class AreaTest {  
    public static void main(String args[]) {  
        Figure[] figures = {  
            new Triangle(2, 3, 3),  
            new Rectangle(5, 8),  
            new Square(5)};  
        for (int i = 0; i < figures.length; i++) {  
            System.out.println(figures[i] +  
                "area = " + figures[i].getArea());  
        }  
    }  
}
```

đào tạo

[Program 2]

```
public abstract class Figure {  
    public abstract double getArea();  
}
```

[Program 3]

```
public class Triangle extends [REDACTED] a {  
    double la;  
    double lb;  
    double lc;  
    public Triangle(double la, double lb, double lc) {  
        this.la = la;  
        this.lb = lb;  
        this.lc = lc;  
    }  
    public String toString() {  
        return "Triangle : sides = " + la + ", " + lb + ", " +  
            lc + " : ";  
    }  
}
```

```

public double getArea() {
    double s = (la + lb + lc) / 2.0;
    return Math.sqrt(s * (s - la) * (s - lb) * (s - lc));
}
}

```

[Program 4]

```

public class Rectangle extends [ ] b {
    double height;
    double width;
    public Rectangle(double height, double width) {
        this.height = height;
        this.width = width;
    }
    public String toString() {
        return "Rectangle : height = " + height +
               ", width = " + width + " : ";
    }
    public double getArea() {
        return [ ] c ;
    }
}

```

tào tạo

[Program 5]

```

public class Square extends [ ] d {
    public Square(double width) {
        [ ] e ;
    }
    public String toString() {
        return "Square : width = " + width + " : ";
    }
}

```

Câu hỏi con

Tùy các nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào ô trống từ **a** đến **e** trong các chương trình trên. Có thể chọn cùng một câu trả lời nhiều lần.

Nhóm câu trả lời cho a, b, và d:

- | | | |
|--------------|-----------|------------|
| a) abstract | b) Figure | c) getArea |
| d) Rectangle | e) Square | f) super |

Nhóm câu trả lời cho c:

- a) height
- c) height * width
- e) width * width

- b) height * height
- d) width

Nhóm câu trả lời cho e:

- a) super(height)
- c) super(width)
- e) super(width, width)
- g) this.height = width
- i) this.width = width

- b) super(height, height)
- d) super(width, height)
- f) this.height = height
- h) this.width = height

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

- Q9.** Hãy đọc mô tả chương trình hợp ngữ sau và đọc chính chương trình, sau đó trả lời các câu hỏi con 1 và 2.

[Mô tả chương trình]

Mười sáu từ nối nhau được xem như là một ma trận gồm bốn dòng và bốn cột. Chương trình con ROTATE là chương trình quay nội dung của ma trận đi 90° theo chiều kim đồng hồ và lưu nó trong ma trận Y.

Từ 0	A	B	C	D
Từ 4	E	F	G	H
Từ 8	I	J	K	L
Từ 12	M	N	O	P

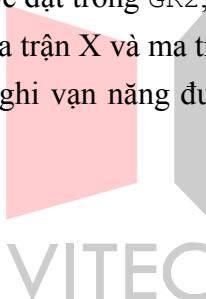
Ma trận X

M	I	E	A
N	J	F	B
O	K	G	C
P	L	H	D

Ma trận Y

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

- (1) Địa chỉ đầu của ma trận X được đặt trong GR1 và được chuyển từ chương trình chính.
- (2) Địa chỉ đầu của ma trận Y được đặt trong GR2, và được chuyển từ chương trình chính.
- (3) Giả thiết rằng các vùng lưu ma trận X và ma trận Y không bị chồng lên nhau.
- (4) Các nội dung gốc của thanh ghi vạn năng được khôi phục lại khi trở về từ chương trình con.



<http://www.vitec.org.vn>

[Chương trình]

(Số dòng)

```
1 ROTATE START ;  
2 RPUSH ;  
3 LAD GR3,16 ;  
4 LOOP0 LAD GR4,4 ;  
5 LOOP1 LD GR5,0,GR1 ;  
6 ST GR5,3,GR2 ;  
7 SUBA GR3,=1 ;  
8 JZE FIN ;  
9 LAD GR1,1,GR1 ; Cập nhật con trỏ của ma trận X  
10 SUBA GR4,=1 ;  
11 JZE NXTROW ; Kết thúc việc xử lý một dòng?  
12 [ ] a ;  
13 JUMP LOOP1 ;  
14 NXTROW [ ] b ; Sang dòng sau  
15 JUMP LOOP0 ;  
16 FIN RPOP ;  
17 RET ;  
18 END ;
```

Trí

tin và Hỗ trợ đào tạo

Câu hỏi con 1

Tùy nhóm câu trả lời sau đây, hãy chọn các câu trả lời đúng để chèn vào các ô trống
[] a và [] b trong chương trình sau.

Nhóm câu trả lời:

- | | |
|--------------------|--------------------|
| a) LAD GR2,-16,GR2 | b) LAD GR2,-15,GR2 |
| c) LAD GR2,-14,GR2 | d) LAD GR2,-13,GR2 |
| e) LAD GR2,1,GR2 | f) LAD GR2,2,GR2 |
| g) LAD GR2,3,GR2 | h) LAD GR2,4,GR2 |
| i) LAD GR2,6,GR2 | j) LAD GR2,8,GR2 |

VITEC

<http://www.vitec.org.vn>

Câu hỏi con 2

Từ nhóm câu trả lời sau đây, hãy chọn các câu trả lời đúng để chèn vào các ô trống **c** và **d** trong đoạn văn bản sau.

Để thay đổi nội dung ma trận X sao cho nó được quay 90° ngược chiều kim đồng hồ và được lưu trong ma trận Y, dòng số 5 phải được thay thành **c** và dòng số 9 phải được thay thành **d**.

Tù	0	A	B	C	D
Tù	4	E	F	G	H
Tù	8	I	J	K	L
Tù	12	M	N	O	P

→

D	H	L	P
C	G	K	O
B	F	J	N
A	E	I	M

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Nhóm câu trả lời cho c:

- | | | | | | |
|----------|----|--------------|----------|----|--------------|
| a) LOOP1 | LD | GR5, 3, GR1 | b) LOOP1 | LD | GR5, 4, GR1 |
| c) LOOP1 | LD | GR5, 7, GR1 | d) LOOP1 | LD | GR5, 8, GR1 |
| e) LOOP1 | LD | GR5, 15, GR1 | f) LOOP1 | LD | GR5, 16, GR1 |

Nhóm câu trả lời cho d:

- | | | | |
|--------|--------------|--------|--------------|
| a) LAD | GR1, -4, GR1 | b) LAD | GR1, -3, GR1 |
| c) LAD | GR1, -2, GR1 | d) LAD | GR1, -1, GR1 |
| e) LAD | GR1, 2, GR1 | f) LAD | GR1, 3, GR1 |
| g) LAD | GR1, 4, GR1 | h) LAD | GR1, 5, GR1 |

Chọn một trong bốn câu hỏi sau (Q10, Q11, Q12, hoặc Q13) để trả lời. Chú ý phải tô đen vào  trong cột Selection Column trên phiếu trả lời đối với câu hỏi mà bạn chọn trả lời. Nếu bạn chọn trả lời nhiều hơn một câu hỏi, thì chỉ có câu trả lời đầu tiên được chấm điểm.

Q10. Đọc mô tả sau về chương trình C và đọc chính chương trình, sau đó trả lời câu hỏi con.

[Mô tả chương trình]

Một chương trình sẽ được tạo ra để xác định hệ số điều chỉnh nhiệt độ dùng để hiệu chỉnh nhiệt độ của một bộ cảm biến áp điện (tức thiết bị tạo ra điện áp phù hợp với cường độ áp suất sử dụng). Đặc tính đầu ra của bộ cảm biến áp điện này biến đổi theo nhiệt độ xung quanh, nên giá trị đầu ra phải được hiệu chỉnh. Các kết quả cho thấy trong bảng dưới đây thu được thông qua việc nghiên cứu thực nghiệm về thay đổi trong giá trị đầu ra của bộ cảm biến áp điện liên quan đến nhiệt độ xung quanh. Bảng này cho thấy các tỷ lệ, với giá trị đầu ra ở 0°C bằng 1.00 (ở đây dấu chấm được hiểu là dấu phẩy thập phân).

Bảng Nhiệt độ xung quanh và tỷ lệ giá trị đầu ra của bộ cảm biến

Nhiệt độ	Tỷ lệ giá trị đầu ra của bộ cảm biến
-40°C	0.20
-20°C	0.60
-10°C	0.80
0°C	1.00
10°C	1.17
30°C	1.50
50°C	1.80

Để hiệu chỉnh giá trị đầu ra của bộ cảm biến áp điện thành giá trị không phụ thuộc vào nhiệt độ xung quanh trong quá trình đo, hệ số hiệu chỉnh nhiệt độ K được xác định dựa trên các dữ liệu trong bảng. Nhiệt độ được hiệu chỉnh bằng cách nhân K với giá trị đầu ra của bộ cảm biến.

- (1) Bảng hiệu chỉnh nhiệt độ được tạo ra từ các dữ liệu trong bảng trên. Bảng điều chỉnh nhiệt độ được thiết lập với cấu trúc mảng.

```
typedef struct {
    int     Temp;   /* Nhiệt độ */  

    double  Ratio; /* Tỷ lệ giá trị đầu ra của bộ cảm biến(giá trị đo thực tế) */  

    double  Step;  /* Tăng từng 1°C */  

} CURVE;
```

- (2) Hàm số `SetupCurve`, cho giá trị ban đầu vào bảng điều chỉnh nhiệt độ, và hàm `GetK`, xác định hệ số điều chỉnh nhiệt độ K, được tạo ra.
- (3) Để xác định hệ số điều chỉnh nhiệt độ K cho nhiệt độ `Degree`, `GetK` tìm bảng điều chỉnh nhiệt độ bằng cách sử dụng phương pháp tìm nhị phân. Nhiệt độ xung quanh được giả thiết là không nhỏ hơn -40°C và không lớn hơn 50°C . Nếu không có giá trị nhiệt độ trong ứng trong bảng điều chỉnh nhiệt độ, thì tỷ lệ giá trị đầu ra ứng với nhiệt độ `Degree` được xác định bằng nội suy tuyến tính, và đảo ngược của giá trị này được trả về là K.
- (4) Chương trình chính là một chương trình kiểm thử, chương trình này xác định và hiển thị hệ số điều chỉnh nhiệt độ K từ -40°C đến 50°C được thêm mỗi lần 1°C , để kiểm chứng hoạt động của hai hàm số này. Dưới đây là ví dụ của việc hiển thị này.

Temperature	Temperature correction coefficient
-40	5.00
-39	4.55
-38	4.17
-37	3.85
-36	3.57
-35	3.33
-34	3.13
-33	2.94
-32	2.78
-31	2.63
-30	2.50
-29	2.38
-28	2.27
-27	2.17
-26	2.08
-25	2.00
:	

[Chương trình]

```
#include <stdio.h>

typedef struct {
    int      Temp      ; /* Nhiệt độ */ 
    double   Ratio     ; /* Tỷ số giá trị đầu ra của bộ cảm biến (giá trị đo thực tế) */
    double   Step      ; /* Mức tăng trên 1°C */ 
} CURVE ;

void    SetupCurve( CURVE * , int );
double  GetK ( int , CURVE * , int );
#define   ITEMS   7

main()
{
    int      Degree ;
    double   k ;
    CURVE  Curve[ ITEMS ] = {
        { -40 , 0.20 , 0.0 },{ -20 , 0.60 , 0.0 },
        { -10 , 0.80 , 0.0 },{ 0 , 1.00 , 0.0 },
        { 10 , 1.17 , 0.0 },{ 30 , 1.50 , 0.0 },
        { 50 , 1.80 , 0.0 } };

    SetupCurve( Curve , ITEMS );
    printf(" Temperature Temperature correction coefficient \n" );
    for( Degree = -40 ; Degree <= 50 ; Degree++ ) {
        k = GetK( Degree , Curve , ITEMS );
        printf( " %3d      %4.2f \n" , Degree , k );
    }
}

void SetupCurve( CURVE *p , int Points )
{
    /* Khởi tạo bảng hiệu chỉnh nhiệt độ */
    int i ;
    for( i = 0 ; i < Points - 1 ; i++ , [ a ] ){
        p->Step = ( [ b ] ) / ( (p+1)->Temp - p->Temp );
    }
}

double GetK( int Temp , CURVE *p , int Points )
{
    /* Trả về hệ số hiệu chỉnh nhiệt độ K là giá trị của hàm */
    int i,j,n ;
    i = 0 ; j = Points - 1 ;
    if ( ( Temp < p->Temp ) || ( Temp > (p+j)->Temp ) )
        return 0.0; /* Trả về giá trị 0.0 nếu ngoài khoảng nhiệt độ đo */
```

Trí

lào tạo

```

/* Nếu trùng với nhiệt độ của phần tử cuối cùng trong bảng hiệu chỉnh */
if ( Temp == ( p+j )->Temp )
    return 1.0 / ( p+j )->Ratio ;

/* Tìm bảng hiệu chỉnh nhiệt độ bằng phương pháp tìm nhị phân */

while( 1 ) {
    n = [c] ;
    if ( ( Temp >= ( p+n )->Temp ) &&
        ( Temp < ( p+n+1 )->Temp ) ) break ;

    if ( Temp < ( p+n )->Temp ) j = n ;
    else [d] ;
}

p += n ;
return 1.0 / ( p->Ratio + p->Step * ( [e] ) );
}

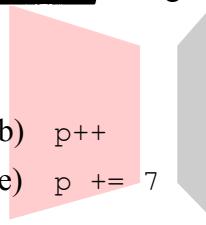
```

Câu hỏi con

Từ các nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào các ô trống từ **a** đến **e** trong chương trình sau.

Nhóm câu trả lời cho a:

- | | | |
|-----------|-----------|--------------|
| a) p-- | b) p++ | c) p->Temp++ |
| d) p += 3 | e) p += 7 | |



Nhóm câu trả lời cho b:

- | | |
|----------------------------|--------------------------------|
| a) (p-1)->Ratio - p->Ratio | b) (p-1)->Ratio - (p+1)->Ratio |
| c) p->Ratio - (p-1)->Ratio | d) p->Ratio - (p->Ratio - 1) |
| e) (p+1)->Ratio - p->Ratio | |

Nhóm câu trả lời cho c:

- | | |
|------------------|------------------|
| a) (i + j) / 2 | b) (i * j) / 2 |
| c) (i % j) / 2 | d) (i + j) * 2 |
| e) (i % j) * 2 | |

Nhóm câu trả lời cho d:

- | | | |
|------------|----------------|----------------|
| a) $i = 0$ | b) $i = n - 1$ | c) $i = n + 1$ |
| d) $j = n$ | e) $j = n + 1$ | |

Nhóm câu trả lời cho e:

- | | | |
|-------------------|-------------------|-------------------|
| a) Temp - p->Step | b) Temp - p->Temp | c) Temp + p->Temp |
| d) p->Temp - n | e) p->Temp - Temp | |

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>

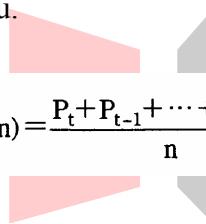
Q11. Hãy đọc mô tả sau về chương trình COBOL và đọc chính chương trình, sau đó trả lời câu hỏi con.

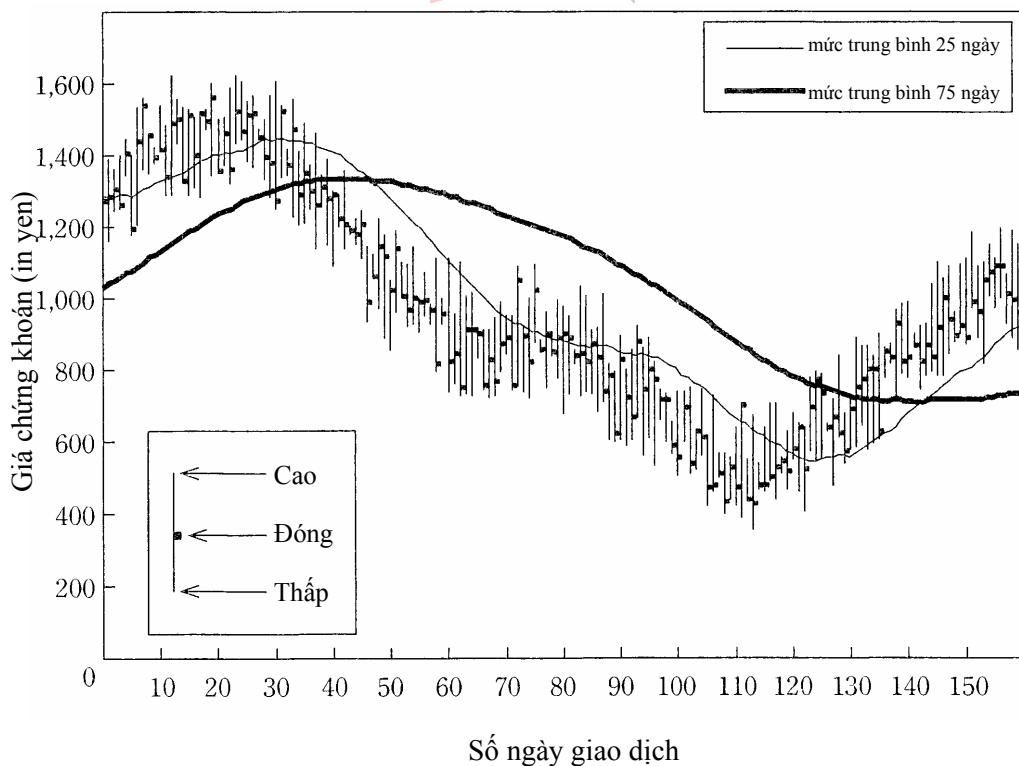
Chương trình này mô phỏng việc giao dịch chứng khoán (mua và bán) theo các nguyên tắc mô tả dưới đây, có sử dụng thông tin về giá chứng khoán trong thời gian đã qua. Nó đọc tệp chứa các giá hàng ngày đối với một đợt phát hành chứng khoán nào đó, ghi các bản ghi giao dịch dựa trên các nguyên tắc này vào một tệp đầu ra, và cuối cùng hiển thị lãi/lỗ - tức kết quả của những đợt mua bán.

[Mô tả chương trình]

(1) Các nguyên tắc buôn bán chứng khoán như sau:

- ① Sử dụng giá trung bình động (moving average) trong 25 ngày và giá trung bình động trong 75 ngày. Trong trường hợp này, giá trung bình động trong n ngày là một đồ thị kiểu đường xác định giá trị trung bình cho giá chứng khoán trên n ngày đã qua, chuyển động theo trực thời gian. Nếu giá trong ngày giao dịch t là P_t , thì giá trung bình động trong n ngày $MA_t(n)$ tại ngày t được xác định theo công thức sau.

$$MA_t(n) = \frac{P_t + P_{t-1} + \dots + P_{t-n+1}}{n}$$




- ② Giá đóng cửa của chứng khoán (tức giá cuối cùng mà chứng khoán được mua bán trong ngày đã cho) được sử dụng để tính giá trung bình động.
- ③ Khi đường giá trung bình động trong 25 ngày cắt đường giá trung bình động trong 75 ngày theo chiều từ dưới lên phía trên, thì chứng khoán được mua vào ngày giao dịch tiếp theo. Cụ thể, nếu $s < t$ và $MA_s(25) < MA_s(75)$ được thiết lập, một khi bắt đầu thực hiện này bị thay đổi lần đầu tiên thành $MA_t(25) < MA_t(75)$, thì chứng khoán được mua vào ngày giao dịch $t + 1$.
- ④ Khi đường giá trung bình động trong 25 ngày cắt đường giá trung bình động trong 75 ngày theo chiều từ trên xuống phía dưới, thì chứng khoán được bán vào ngày giao dịch tiếp theo. Cụ thể, nếu $s < t$ và $MA_s(25) > MA_s(75)$ được thiết lập, một khi bắt đầu thực hiện này bị thay đổi lần đầu tiên thành $MA_t(25) > MA_t(75)$, thì chứng khoán được bán vào ngày giao dịch $t + 1$. Tuy nhiên, chứng khoán không thể bán được nếu nó còn chưa được mua.
- ⑤ Giả thiết rằng chứng khoán có thể được mua và được bán ở giá trị trung bình của cao (giá buôn bán cao nhất) và thấp (giá buôn bán thấp nhất) vào ngày giao dịch t .
- ⑥ Giả thiết rằng không có chi phí nào khác liên quan đến việc mua bán.
- ⑦ Giả thiết rằng mỗi lần có một cổ phiếu chứng khoán được mua bán.

- (2) Tệp đầu vào (INFILE) là một tệp tuần tự, trong đó dữ liệu về giá chứng khoán trong ít nhất 75 ngày giao dịch của một đợt phát hành chứng khoán đã cho được lưu theo các thời gian. Bản ghi có định dạng như sau.

Ngày 8 chữ số	Mở 9 chữ số	Đóng 9 chữ số	Cao 9 chữ số	Thấp 9 chữ số
------------------	----------------	------------------	-----------------	------------------

- (3) Tệp đầu ra (OUTFILE) là một tệp tuần tự chứa các bản ghi mua bán xảy ra khi các quy tắc trong (1) được áp dụng với dữ liệu giá chứng khoán trong tệp dữ liệu đầu vào. Định dạng của bản ghi như sau.

Ngày 8 chữ số	Cờ (flag) 1 chữ số	Giá mua bán 10 chữ số
------------------	-----------------------	--------------------------

- ① Bản ghi đưa ra tệp đầu ra (OUTFILE) chứa ngày ứng với ngày (ngày giao dịch t) khi có quyết định mua bán, cũng như cờ và giá mua bán.
- ② Cờ chứa “S” nếu bán, và “B” nếu mua.
- ③ Giá mua bán chứa một giá trị với một chữ số đằng sau dấu phẩy thập phân.

- (4) Lãi/lỗ của các đợt mua bán được hiển thị là giá trị số với dấu cộng hoặc dấu trừ, và chỉ với một chữ số ở đằng sau dấu phẩy thập phân.

Example :

Total Revenue:	-114.5
----------------	--------

[Program]

```

DATA DIVISION.
FILE SECTION.
FD INFILE.
01 I-MARKET-PRICE-REC.
  05 I-YYYYMMDD-MARKET      PIC X(8).
  05 I-OPENING-PRICE        PIC 9(9).
  05 I-CLOSING-PRICE        PIC 9(9).
  05 I-HIGHEST-PRICE        PIC 9(9).
  05 I-LOWEST-PRICE         PIC 9(9).

FD OUTFILE.
01 O-EXCHANGE-REC.
  05 O-YYYYMMDD-EXCHANGE    PIC X(8).
  05 O-ACTION               PIC X.
  05 O-PRICE                PIC 9(9)V9.

WORKING-STORAGE SECTION.
01 SERIES-SIZE            PIC 9(4) VALUE 100.
01 RANGE-SHORT             PIC 9(4) VALUE 25.
01 RANGE-LONG              PIC 9(4) VALUE 75.
01 W-MARKET-PRICE-REC.
  05 W-YYYYMMDD-MARKET      PIC X(8).
  05 W-OPENING-PRICE        PIC 9(9).
  05 W-CLOSING-PRICE        PIC 9(9).
  05 W-HIGHEST-PRICE        PIC 9(9).
  05 W-LOWEST-PRICE         PIC 9(9).

01 MARKET-PRICE-SERIES.
  05 TBL-MARKET-PRICE-REC OCCURS 100.
    10 TBL-YYYYMMDD-MARKET    PIC X(8).
    10 TBL-OPENING-PRICE      PIC 9(9).
    10 TBL-CLOSING-PRICE      PIC 9(9).
    10 TBL-HIGHEST-PRICE      PIC 9(9).
    10 TBL-LOWEST-PRICE       PIC 9(9).

  01 SERIES-TOP              PIC 9(4).
  01 TAIL-LONG               PIC 9(4).
  01 TAIL-SHORT              PIC 9(4).
  01 SUM-SHORT               PIC 9(12).
  01 SUM-LONG                PIC 9(12).
  01 MA-SHORT                PIC 9(9)V9.
  01 MA-LONG                 PIC 9(9)V9.
  01 MA-COMPARE-CURRENT     PIC X.
    88 LONG-LT-SHORT-CUR     VALUE "S".
    88 SHORT-LT-LONG-CUR     VALUE "L".
    88 SHORT-EQ-LONG-CUR     VALUE "E".

  01 MA-COMPARE-PREVIOUS    PIC X.
    88 LONG-LT-SHORT-PRE     VALUE "S".

```

Trun

Hỗ trợ đào tạo

```

88  SHORT-LT-LONG-PRE      VALUE "L".
88  SHORT-EQ-LONG-PRE      VALUE "E".
01  INFILE-EOF-FLG         PIC X.
  88  INFILE-EOF            VALUE "Y".
  88  INFILE-NOT-EOF        VALUE "N".
01  TRADING-PRICE          PIC 9(9)V9.
01  REVENUE                PIC S9(9)V9.
01  REVENUE-ED              PIC +++++++9.9.
01  INDX                   PIC 9(9).
01  TRADING-ACTION         PIC S9.
  88  FLG-NO-TRADE         VALUE 0.
  88  FLG-SELL              VALUE 1.
  88  FLG-BUY                VALUE -1.
01  STOCK-HOLDING          PIC X.
  88  HAS-STOCK             VALUE "Y".
  88  HAS-NO-STOCK          VALUE "N".
PROCEDURE DIVISION.
MAIN-PARAGRAPH.
  OPEN INPUT INFILE  OUTPUT OUTFILE.
  PERFORM INIT-MOVING-AVARAGE.
  IF MA-SHORT = MA-LONG THEN
    SET SHORT-EQ-LONG-CUR TO TRUE
  END-IF.
  PERFORM SET-CURRENT-MA-COMPARISON.
  MOVE ZERO TO REVENUE.
  SET HAS-NO-STOCK TO TRUE.
  SET FLG-NO-TRADE TO TRUE.
  SET INFILE-NOT-EOF TO TRUE.
  PERFORM UNTIL INFILE-EOF
    IF FLG-BUY OR FLG-SELL THEN
      PERFORM TRADE-STOCK-AND-MAKE-RECORD
    END-IF
    READ INFILE INTO W-MARKET-PRICE-REC
    AT END
      SET INFILE-EOF TO TRUE
    NOT AT END
      PERFORM CALC-NEXT-MOVING-AVERAGE
      PERFORM DECIDE-TRADING
    END-READ
  END-PERFORM.
  MOVE REVENUE TO REVENUE-ED.
  DISPLAY "Total Revenue: " REVENUE-ED.
  CLOSE INFILE OUTFILE.
  STOP RUN.

INIT-MOVING-AVARAGE.
  PERFORM VARYING SERIES-TOP FROM 1 BY 1
    UNTIL RANGE-LONG < SERIES-TOP
    READ INFILE INTO TBL-MARKET-PRICE-REC(SERIES-TOP)
  END-PERFORM.
  COMPUTE TAIL-SHORT = SERIES-TOP - RANGE-SHORT.
  COMPUTE TAIL-LONG = SERIES-TOP - RANGE-LONG.
  SUBTRACT 1 FROM SERIES-TOP.
  MOVE ZERO TO SUM-SHORT.
  PERFORM VARYING INDX   a

```

```

        ADD TBL-CLOSING-PRICE(INDX) TO SUM-SHORT
END-PERFORM.
MOVE SUM-SHORT TO SUM-LONG.
PERFORM VARYING INDX b
        ADD TBL-CLOSING-PRICE(INDX) TO SUM-LONG
END-PERFORM.
COMPUTE MA-SHORT ROUNDED = SUM-SHORT / RANGE-SHORT.
COMPUTE MA-LONG ROUNDED = SUM-LONG / RANGE-LONG.
SET-CURRENT-MA-COMPARISON.
EVALUATE TRUE
    WHEN MA-SHORT = MA-LONG
        CONTINUE
    WHEN MA-SHORT < MA-LONG
        SET SHORT-LT-LONG-CUR TO TRUE
    WHEN MA-LONG < MA-SHORT
        SET LONG-LT-SHORT-CUR TO TRUE
    END-EVALUATE.
TRADE-STOCK-AND-MAKE-RECORD.
EVALUATE TRUE
    WHEN FLG-BUY
        MOVE "B" TO O-ACTION
        SET HAS-STOCK TO TRUE
    WHEN FLG-SELL AND c
        MOVE "S" TO O-ACTION
        SET HAS-NO-STOCK TO TRUE
    END-EVALUATE.
IF O-ACTION = "B" OR O-ACTION = "S"
THEN
    COMPUTE TRADING-PRICE ROUNDED
        = ( TBL-HIGHEST-PRICE(SERIES-TOP)
            + TBL-LOWEST-PRICE(SERIES-TOP) ) / 2
    COMPUTE REVENUE = REVENUE + TRADING-PRICE * d
    MOVE TBL-YYYYMMDD-MARKET(SERIES-TOP) TO
        O-YYYYMMDD-EXCHANGE
    MOVE TRADING-PRICE TO O-PRICE
    WRITE O-EXCHANGE-REC
END-IF.
CALC-NEXT-MOVING-AVERAGE.
ADD W-CLOSING-PRICE TO SUM-SHORT SUM-LONG.
SUBTRACT TBL-CLOSING-PRICE(TAIL-SHORT) FROM SUM-SHORT.
SUBTRACT TBL-CLOSING-PRICE(TAIL-LONG) FROM SUM-LONG.
COMPUTE MA-SHORT ROUNDED = SUM-SHORT / RANGE-SHORT.
COMPUTE MA-LONG ROUNDED = SUM-LONG / RANGE-LONG.
ADD 1 TO SERIES-TOP TAIL-SHORT TAIL-LONG.
EVALUATE TRUE
    WHEN SERIES-SIZE < SERIES-TOP
        MOVE 1 TO SERIES-TOP
    WHEN SERIES-SIZE < TAIL-SHORT
        MOVE 1 TO TAIL-SHORT
    WHEN SERIES-SIZE < TAIL-LONG
        MOVE 1 TO TAIL-LONG
END-EVALUATE.
MOVE W-MARKET-PRICE-REC TO TBL-MARKET-PRICE-REC(SERIES-TOP).

```

```

DECIDE-TRADING.
MOVE MA-COMPARE-CURRENT TO MA-COMPARE-PREVIOUS.
PERFORM SET-CURRENT-MA-COMPARISON.
SET FLG-NO-TRADE TO TRUE.
EVALUATE TRUE
WHEN [ ] e AND LONG-LT-SHORT-CUR
      SET FLG-BUY TO TRUE
WHEN LONG-LT-SHORT-PRE AND SHORT-LT-LONG-CUR
      SET FLG-SELL TO TRUE
END-EVALUATE.

```

Câu hỏi con

Tùy các nhóm câu trả lời sau, hãy chọn các câu trả lời đúng để điền vào ô trống từ **a** đến **e** trong chương trình trên.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo

Nhóm câu trả lời cho a và b:

- a) FROM 1 BY 1 UNTIL SERIES-SIZE < INDX
- b) FROM SERIES-TOP - 1 BY -1 UNTIL INDX < TAIL-SHORT
- c) FROM SERIES-TOP BY -1 UNTIL INDX <= TAIL-SHORT
- d) FROM TAIL-LONG BY 1 UNTIL TAIL-SHORT <= INDX
- e) FROM TAIL-SHORT BY 1 UNTIL SERIES-TOP < INDX

Nhóm câu trả lời cho c tới e:

- | | |
|-----------------------|----------------------|
| a) FLG-BUY | b) FLG-SELL |
| c) HAS-STOCK | d) LONG-LT-SHORT-CUR |
| e) MA-COMPARE-CURRENT | f) SHORT-LT-LONG-PRE |
| g) TRADING-ACTION | |

Q12. Hãy đọc mô tả chương trình Java và đọc chính chương trình đó, sau đó trả lời các Câu hỏi con 1 và 2.

[Mô tả chương trình]

Thư viện của một trường học có 30 chỗ tự học, các chỗ đó được quản lý theo quy tắc sau.

(1) Các quy tắc quản lý

- ① Học sinh muốn dùng chỗ phải đưa yêu cầu tại thường trực và dùng chỗ đã được phân cho. Khi học sinh không sử dụng chỗ nữa thì phải báo cho thường trực.
- ② Nếu có học sinh muốn dùng chỗ nhưng không có chỗ, thì học sinh đang sử dụng chỗ lâu nhất (quá 1 giờ) phải giải phóng chỗ cho người học sinh mới muốn dùng chỗ. Nếu không có chỗ trống và không có học sinh nào đang sử dụng chỗ quá 1 giờ, thì người học sinh muốn dùng chỗ không thể dùng chỗ.
- ③ Một học sinh không thể dùng nhiều chỗ cùng một lúc.

(2) Chương trình được thiết kế như sau để hỗ trợ trong việc quản lý các chỗ tự học trên cơ sở các quy tắc quản lý như trên.

- ① Lớp `ListElement` được định nghĩa để thực hiện một danh sách hai chiều như chỉ ra trong sơ đồ dưới đây. Danh sách hai chiều có dạng vòng tròn, và được thiết kế sao cho đầu cuối không nhận xử lý đặc biệt trong quy trình chèn và xoá phần tử. `ListElement` biểu diễn cả đầu (bắt đầu) danh sách và các phần tử. Các phương thức sau được thực hiện trong `ListElement`.

`nextElement`

Trả về phần tử tiếp theo của thẻ nghiệm (instance) `ListElement` này.

`previousElement`

Trả về phần tử trước của thẻ nghiệm `ListElement` này.

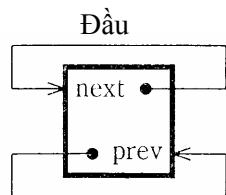
`insertBefore`

Chèn thẻ nghiệm `ListElement` này trước phần tử được xác định trong đối số.

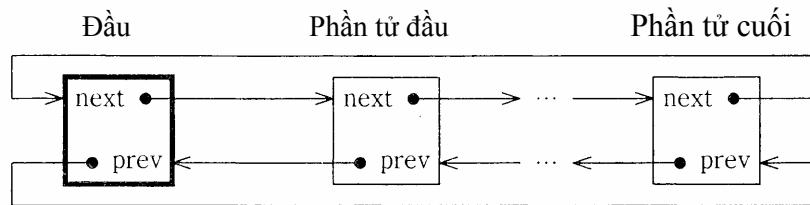
`remove`

Loại thẻ nghiệm `ListElement` này ra khỏi danh sách.

Danh sách rỗng



Danh sách chứa các phần tử



Hình: Các trạng thái của danh sách hai chiều đối với danh sách rỗng và danh sách có chứa các phần tử

- ② Các chỗ được chỉ ra bởi lớp Seat, lớp này mở rộng ListElement. Mỗi chỗ được gán một số hiệu.
- ③ Lớp SeatManager được dùng để thi hành các phương thức thực hiện các quy tắc quản lý.
- ④ Danh sách các chỗ trống và danh sách các chỗ bị chiếm tương ứng là freeSeats và occupiedSeats. Các thẻ nghiệm Seat trở thành các phần tử của mỗi danh sách.
- ⑤ Các chỗ bị chiếm được đăng ký trong occupiedSeats. Danh sách này được quản lý sao cho các chỗ được sắp xếp từ chỗ có thời gian sử dụng ngắn nhất tới chỗ có thời gian sử dụng lâu nhất.
- ⑥ SeatManager thực hiện các phương thức public sau.

checkin

Khẳng định rằng học sinh muốn có chỗ - học sinh này được xác định bằng một đối số - chưa sử dụng chỗ. Nếu có chỗ trống trong freeSeats (danh sách các chỗ trống), thì chỗ đó được gán cho người học sinh muốn có chỗ. Nếu không thì chỗ đã bị sử dụng hơn một giờ được gán. Chỗ được gán được đăng ký như là phần tử đầu tiên của trong occupiedSeats (danh sách các chỗ bị chiếm), và trả về giá trị là thẻ nghiệm Seat. Nếu không có chỗ nào dùng được, trả về giá trị null.

checkout

Xoá chỗ mà người dùng (được xác định bằng đối số) đã dùng ra khỏi occupiedSeats, chuyển chỗ đó lại vào freeSeats, và trả về giá trị true.

Nếu người dùng không được tìm thấy trong occupiedSeats, trả về giá trị false.

Đầu tiên, lớp ListElement và lớp Seat được thi hành và tiến hành phép kiểm thử đơn vị để khẳng định các thao tác là đúng. Tiếp theo, phép kiểm thử với lớp SeatManager được thi hành. Trong phép kiểm thử này, phát hiện ra một vấn đề là một học sinh đã có thể sử dụng nhiều chỗ cùng một lúc. Về các khía cạnh khác, hoạt động diễn ra bình thường.

[Program 1]

Trung | đào tạo

```
public class ListElement {
    private ListElement prev, next;
    public ListElement() {
        prev = next = this;
    }
    public ListElement nextElement() { return next; }
    public ListElement previousElement() { return prev; }
    public void insertBefore(ListElement element) {
        next = element;
        prev = element.prev;
        next.prev = prev.next = [a];
    }
    public void remove() {
        [b] = next;
        [c] = prev;
        prev = next = this;
    }
}
```

[Program 2]

```
public class Seat extends ListElement {
    private String userID;      // Mã người dùng
    private long checkinTime;   // Thời điểm nhận chỗ
    private int seatNumber;     // Số hiệu chỗ

    public Seat(int seatNumber) {
        this.seatNumber = seatNumber;
    }
    public int getSeatNumber() {
        return seatNumber;
    }
}
```

```

public String getUserId() {
    return userID;
}
public void setUserId(String userID) {
    this.userID = userID;
}
public boolean isUsedBy(String userID) {
    return this.userID.equals(userID);
}
public long getCheckinTime() {
    return checkinTime;
}
public void setCheckinTime(long time) {
    checkinTime = time;
}
}

```

[Program 3]

Trung **Đào tạo**

```

public class SeatManager {
    private static final int NSEATS = 30; // Tổng số chỗ
    // Thời gian sử dụng tối đa (ms)
    private static final int MAXTIME = 60 * 60 * 1000;
    // Danh sách chỗ còn trống
    private ListElement freeSeats = new ListElement();
    // Danh sách chỗ đang bận
    private ListElement occupiedSeats = new ListElement();

    public SeatManager() {
        for (int i = 1; i <= NSEATS; i++) {
            Seat seat = new Seat(i);
            seat.insertBefore(freeSeats);
        }
    }

    // Nếu có 1 chỗ trống trong d. sách chỗ trống, thề nghiệm Seat bị xoá đi trong d.sách chỗ trống
    // và trả về thề nghiệm. Nếu không còn chỗ trống, trả về null.
    private Seat getFreeSeat() {
        ListElement le = freeSeats.nextElement();
        if (le != freeSeats) {
            le.remove();
            return (Seat) le;
        }
        return null;
    }

    // Kiểm tra danh sách chỗ bận. Nếu có người chiếm chỗ lâu hơn thời gian sử dụng tối đa
    // đưa ra một thông báo và gọi phương thức checkout.
    private void vacateExpiredSeat(long time) {
        ListElement le = [REDACTED] d;
    }
}

```

```

        if (le != occupiedSeats) {
            Seat seat = (Seat) le;
            if ((seat.getCheckinTime() + MAXTIME) < time) {
                System.out.println("Seat#" +
                    seat.getSeatNumber() + " " +
                    seat.getUserID() +
                    " must check out.");
                checkout(seat.getUserID());
            }
        }
    }

// Tìm chỗ đang bị chiếm bởi người dùng chỉ ra trong danh sách chỗ bận. Nếu tìm thấy người dùng
// trả về chỗ tương ứng. Nếu không, trả về null.
private Seat findUser(String userID) {
    ListElement le = [REDACTED] e [REDACTED];
    while (le != occupiedSeats) {
        Seat seat = (Seat) le;
        if (seat.isUsedBy(userID)) {
            return seat;
        }
        le = le.nextElement();
    }
    return null;
}

public Seat checkin(String userID) {
    long now = System.currentTimeMillis();
    Seat seat = getFreeSeat();
    if (seat == null) {
        vacateExpiredSeat(now);
        seat = getFreeSeat();
    }
    if (seat != null) {
        seat.setCheckinTime(now);
        seat.setUserID(userID);
        seat.insertBefore(occupiedSeats.nextElement());
    }
    return seat;
}

public boolean checkout(String userID) {
    Seat seat = findUser(userID);
    if (seat != null) {
        seat.remove();
        seat.setUserID(null);
        seat.insertBefore(freeSeats);
        return true;
    }
    return false;
}
}

```

Trí

Io tạo

Câu hỏi con 1

Từ các nhóm câu trả lời sau, hãy chọn các câu trả lời đúng để điền vào các ô trống từ a đến e trong chương trình trên.

Nhóm câu trả lời cho a tới c:

- a) element
- b) element.next
- c) element.prev
- d) new ListElement()
- e) next
- f) next.prev
- g) null
- h) prev
- i) prev.next
- j) this

Nhóm câu trả lời cho d và e:

- a) freeSeats
- b) freeSeats.nextElement()
- c) freeSeats.previousElement()
- d) occupiedSeats
- e) occupiedSeats.nextElement()
- f) occupiedSeats.previousElement()

Câu hỏi con 2

Chương trình 3 cần được sửa chữa sao cho một học sinh không thể sử dụng nhiều chỗ ngồi cùng một lúc. Từ nhóm câu trả lời sau, hãy chọn phương thức phù hợp để thực hiện việc điều chỉnh này. Lưu ý rằng câu hỏi này giả thiết rằng các câu trả lời đúng đã được điền vào tất cả các ô trống từ a đến e trong các chương trình trên.

<http://www.vitec.org.vn>

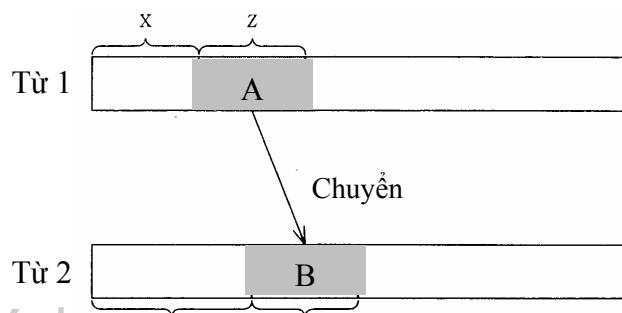
Nhóm câu trả lời:

- a) Trong phương thức checkin, occupiedSeats được kiểm tra trước khi bố trí một chỗ. Nếu học sinh muốn dùng chỗ mà lại đang sử dụng một chỗ khác, lỗi sẽ xuất hiện và quy trình bố trí chỗ không được thực hiện.
- b) Trong phương thức checkout, các chỗ không còn bận nữa không được loại khỏi occupiedSeats. Vì thế, hãy xóa cho đúng những chỗ không còn bận nữa ra khỏi occupiedSeats.
- c) Trong phương thức vacateExpiredSeat, phương thức checkout được gọi để bắt người dùng đã quá hạn sử dụng tối đa (MAXTIME) phải trả chỗ. Lập tức sau đó, một quy trình được thêm vào để xóa chỗ mà người dùng đó đã sử dụng ra khỏi occupiedSeats.

Q13. Hãy đọc mô tả chương trình hợp ngữ sau và đọc chính chương trình, sau đó trả lời các câu hỏi con từ 1 đến 3.

[Mô tả chương trình]

Chương trình con `BMOVE` chuyển chuỗi bit A trong Từ 1 sang vị trí B trong từ 2 như dưới đây. x, y, và z đều chỉ các số lượng bits.



Trung tâm Sát h... Hỗ trợ đào tạo

- (1) Thông tin dưới đây được thiết lập trong các biến từ GR1 đến GR5 và được gọi từ chương trình chính.

GR1: Địa chỉ của Từ 1

GR2: Địa chỉ của Từ 2

GR3: x

GR4: y

GR5: z



- (2) Giả thiết rằng: $x \geq 0, y \geq 0, z \geq 1$.
- (3) Giả thiết rằng: $x + z \leq 16, y + z \leq 16$.
- (4) Từ 1 và Từ 2 là các từ khác nhau.
- (5) Nội dung gốc của thanh ghi vạn năng được khôi phục lại khi được trở về từ chương trình con.

[Chương trình]

(Số dòng)

```
1 BMOVE START ;  
2 PUSH 0,GR6 ;}  
3 PUSH 0,GR7 ;}  
4 LD GR6,=#8000 ;}  
5 [ ] a ;}  
6 LD GR7,GR6 ;}  
7 SRL GR7,0,GR4 ;}  
8 XOR GR7,=#FFFF ;}  
9 AND GR7,0,GR2 ;}  
10 [ ] b ;}  
11 AND GR6,0,GR1 ;}  
12 SLL GR6,0,GR3 ;}  
13 [ ] c ;}  
14 OR GR6,GR7 ;}  
15 ST GR6,0,GR2 ;}  
16 POP GR7 ;}  
17 POP GR6 ;}  
18 RET ;}  
19 END ;}
```

Trung

Câu hỏi con 1

Tù nhom câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào các ô trống từ **a** đến **c** trong chương trình trên.

Nhóm câu trả lời:

- | | |
|-------------------|-------------------|
| a) SLA GR6,0,GR3 | b) SLA GR6,0,GR4 |
| c) SLL GR6,0,GR3 | d) SLL GR6,0,GR4 |
| e) SRA GR6,-1,GR3 | f) SRA GR6,-1,GR5 |
| g) SRL GR6,-1,GR5 | h) SRL GR6,0,GR3 |
| i) SRL GR6,0,GR4 | |

Câu hỏi con 2

Từ nhóm câu trả lời dưới đây, hãy chọn câu trả lời chỉ ra đúng nội dung của GR6 ngay sau khi thực hiện lệnh (SLL) trong dòng 12, khi nội dung của Từ 1, Từ 2, x, y, và z được chuyển từ chương trình chính là như sau.

Từ 1	0010	0110	1101	0111
Từ 2	1100	1101	0111	1010

$$x = 9$$

$$y = 3$$

$$z = 5$$

Nhóm câu trả lời:

Trur. 3

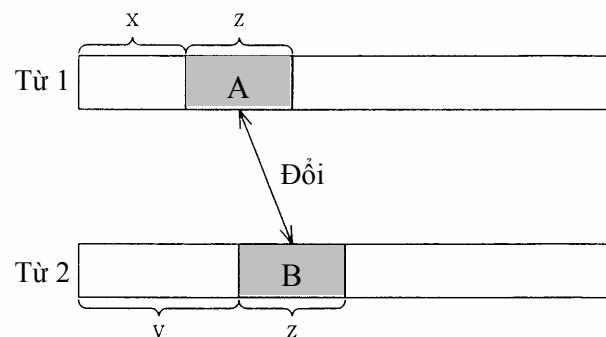
- | | | | | |
|----|------|------|------|------|
| a) | 0000 | 0000 | 0001 | 0101 |
|----|------|------|------|------|
- | | | | | |
|----|------|------|------|------|
| b) | 0000 | 0000 | 0101 | 0100 |
|----|------|------|------|------|
- | | | | | |
|----|------|------|------|------|
| c) | 0001 | 0101 | 0000 | 0000 |
|----|------|------|------|------|
- | | | | | |
|----|------|------|------|------|
| d) | 1010 | 1000 | 0000 | 0000 |
|----|------|------|------|------|



<http://www.vitec.org.vn>

Câu hỏi con 3

Một chương trình có tên là BSWAP, dùng để đổi chỗ chuỗi bit A trong Tù 1 với chuỗi bit B trong Tù 2 bằng các lệnh chương trình con BMOVE, được tạo ra. Từ nhóm câu trả lời dưới đây, hãy chọn các câu trả lời đúng để điền vào các ô trống **d** và **e** như sau. Trong trường hợp này, các tham biến riêng biệt được đặt trong GR1 đến GR5 và được gọi giống như trong chương trình BMOVE.



Trung t

lõi trợ đào tạo

[Số dòng]

```

1 BSWAP  START          ;
2      RPUSH           ; Lưu thanh ghi
3      LD    GR6, 0 ,GR2   ;
4      ST    GR6,WORD     ; Lưu từ 2 vào WORD
5      CALL  BMOVE        ;
6      LD    GR2,GR1      ; Đặt địa chỉ Từ 1 như địa chỉ Từ 2
7      [ ] d              ; Đặt địa chỉ WORD như địa chỉ Từ 1
8      PUSH  0 ,GR4       ;
9      [ ] e              ; } Đổi x và y
10     POP   GR3          ;
11     CALL  BMOVE        ;
12     RPOP             ; Khôi phục thanh ghi
13     RET               ;
14 WORD  DS   1           ;
15 END               ;

```

Nhóm câu trả lời:

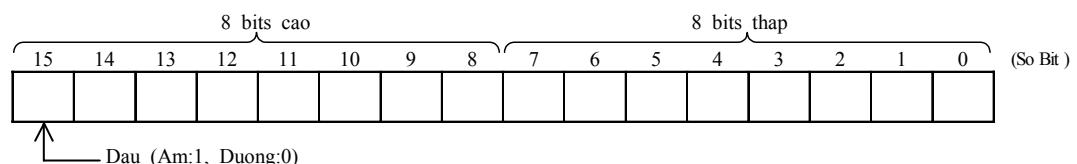
- | | | | |
|--------|----------|---------|----------|
| a) LAD | GR1,WORD | b) LAD | GR4,WORD |
| c) LD | GR1,WORD | d) LD | GR3,GR4 |
| e) LD | GR4,GR3 | f) LD | GR4,WORD |
| g) POP | GR4 | h) PUSH | 0,GR3 |

Đặc tả hợp ngữ

1. Đặc tả phần cứng của máy COMET II

1.1 Đặc tả phần cứng

- (1) Một từ 16 bits, và định dạng bit là như sau:



- (2) Dung lượng bộ nhớ chính là 65,536 từ với số địa chỉ từ 0 tới 65,535.
(3) Các giá trị số được biểu diễn như các số nhị phân 16-bit. Số âm được biểu diễn theo phần bù của hai.
(4) Điều khiển là tuần tự. COMET II dùng từ lệnh một từ và hai từ.

(5) Máy COMET II có bốn kiểu thanh ghi: GR (16 bits), SP (16 bits), PR (16 bits) and FR (3 bits).

Có tám thanh ghi GR (General Register - thanh ghi chung), GR0 tới GR7. Tám thanh ghi này được dùng cho các phép toán số học, logic, so sánh và dịch chuyển. Trong số những thanh ghi này, GR1 tới GR7 cũng còn được dùng làm thanh ghi chỉ số để sửa đổi địa chỉ.

Con trỏ chồng lưu giữ địa chỉ hiện thời tại định chồng.

Thanh ghi PR (Program Register - thanh ghi chương trình) lưu giữ địa chỉ đầu tiên của lệnh tiếp theo.

Thanh ghi FR (Flag Register - thanh ghi cờ) bao gồm ba bits: OF (Overflow Flag - cờ tràn), SF (Sign Flag - cờ dấu) và ZF (Zero Flag - cờ không). Các giá trị sau đây được đặt, tuỳ theo kết quả được sinh ra bởi phép toán và các lệnh phép toán nào đó. Các giá trị này được tham chiếu qua các lệnh nhảy có điều kiện.

OF	Khi kết quả của lệnh phép toán số học vượt ra ngoài miền $-32,768$ tới $32,767$, giá trị là 1, và trong các trường hợp khác, giá trị là 0. Khi kết quả của lệnh phép toán logic là ở ngoài miền từ 0 tới 65,535, thì giá trị là 1, và trong các trường hợp khác, giá trị là 0.
SF	Khi dấu của kết quả phép toán là âm (bit số 15 = 1), thì giá trị là 1, còn trong các trường hợp khác, giá trị là 0.
ZF	Khi kết quả phép toán là 0 (tất cả các bit đều là 0), thì giá trị là 1, còn trong các trường hợp khác, giá trị là 0.

- (6) Phép cộng hay phép trừ logic: Xử lí các dữ liệu được cộng hay trừ như dữ liệu không dấu, và thực hiện phép cộng hay trừ.

1.2 Các lệnh

Định dạng và chức năng của các lệnh được mô tả trong sơ đồ sau. Khi một mã lệnh có hai kiểu toán hạng, thì toán hạng trên chỉ ra lệnh giữa các thanh ghi còn toán hạng dưới chỉ ra lệnh giữa các thanh ghi và bộ nhớ chính.

Lệnh	Định dạng		Mô tả lệnh	Đặt FR
	Mã phép toán	Toán hạng		

(1) Lệnh nạp, lưu giữ, nạp địa chỉ

Load	LD	R1,r2 R,adr [,x]	r1 ← (r2) r ← (Địa chỉ hiệu dụng)	O*1
STore	ST	R,adr [,x]	Địa chỉ hiệu dụng ← (r)	
Load Address	LAD	R,adr [,x]	r ← Địa chỉ hiệu dụng	-

(2) Lệnh phép toán số học và logic

ADD Arithmetic (cộng số học)	ADDA	R1,r2	r1 ← (r1) + (r2)	O
		R,adr [,x]	r ← (r) + (địa chỉ hiệu dụng)	
ADD Logical (cộng logic)	ADDL	R1,r2	r1 ← (r1) + _L (r2)	
		R,adr [,x]	r ← (r) + _L (địa chỉ hiệu dụng)	
SUBtract Arithmetic (trừ số học)	SUBA	R1,r2	r1 ← (r1) - (r2)	
		R,adr [,x]	r ← (r) - (địa chỉ hiệu dụng)	
SUBtract Logical (trừ logic)	SUBL	R1,r2	r1 ← (r1) - _L (r2)	
		R,adr [,x]	r ← (r) - _L (địa chỉ hiệu dụng)	
AND (và)	AND	R1,r2	r1 ← (r1) AND (r2)	O*1
		R,adr [,x]	r ← (r) AND (địa chỉ hiệu dụng)	
OR (hoặc)	OR	R1,r2	r1 ← (r1) OR (r2)	
		R,adr [,x]	r ← (r) OR (địa chỉ hiệu dụng)	
eXclusive OR (hoặc loại trừ)	XOR	R1,r2	r1 ← (r1) XOR (r2)	
		R,adr [,x]	r ← (r) XOR (địa chỉ hiệu dụng)	

(3) Lệnh phép toán so sánh

So sánh số học	CPA	r1,r2 r,adr [,x]	Thực hiện phép toán so sánh số học hay so sánh logic trên (r1) và (r2) hay (r) và (địa chỉ hiệu dụng), và đặt FR như sau, tương ứng với kết quả của phép toán so sánh.	O*1
			Kết quả so sánh	
So sánh logic	CPL	r1,r2 r,adr [,x]	Gia trị	
			SF	
			(r1) > (r2)	
			0	
			(r) > (địa chỉ hiệu dụng)	
			0	
			(r1) = (r2)	
			(r) = (địa chỉ hiệu dụng)	
			(r1) < (r2)	
			1	
			(r) < (địa chỉ hiệu dụng)	

(4) Lệnh phép toán dịch chuyển

Dịch chuyển số học trái	SLA	r,adr [,x]	Dịch chuyển (r) (trừ bit dấu) bên trái hay bên phải theo số bit được xác định bởi địa chỉ hiệu dụng.	○*2
Dịch chuyển số học phải	SRA	r,adr [,x]	Khi việc dịch chuyển trái được thực hiện, các bit bị bỏ trống do việc dịch chuyển này sẽ được điền bằng số không. Khi phép dịch chuyển phải được thực hiện, các bit bị bỏ trống do việc dịch chuyển này được điền bằng cùng giá trị như bit dấu.	
Dịch chuyển logic trái	SLL	r,adr [,x]	Dịch chuyển (r) (kể cả bit dấu) bên trái hay bên phải theo số bit được xác định bởi địa chỉ hiệu dụng.	
Dịch chuyển logic phải	SRL	r,adr [,x]	Những bit bị bỏ trống do việc dịch chuyển này được điền bằng số không.	

(5) Lệnh rẽ nhánh

Jump on PLus (nhảy theo cộng)	JPL adr [,x]	Rẽ nhánh tới địa chỉ hiệu dụng, tùy theo giá trị của FR. Nếu điều khiển không rẽ nhánh sang địa chỉ mới, thì việc thực hiện tiếp tục với lệnh tiếp.	-																										
Jump on MInus (nhảy theo trừ)	JMI adr [,x]																												
Jump on Non Zero (nhảy theo khác không)	JNZ adr [,x]																												
Jump on ZEro (nhảy theo không)	JZE adr [,x]																												
Jump on OVerflow (nhảy theo tràn)	JOV adr [,x]	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2">Lệnh</th> <th colspan="3">Giá trị của FR để rẽ nhánh</th> </tr> <tr> <th>OF</th> <th>SF</th> <th>ZF</th> </tr> </thead> <tbody> <tr> <td>JPL</td> <td></td> <td>0</td> <td>0</td> </tr> <tr> <td>JMI</td> <td></td> <td>1</td> <td></td> </tr> <tr> <td>JNZ</td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>JZE</td> <td></td> <td></td> <td>1</td> </tr> <tr> <td>JOV</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>		Lệnh	Giá trị của FR để rẽ nhánh			OF	SF	ZF	JPL		0	0	JMI		1		JNZ			0	JZE			1	JOV	1	
Lệnh	Giá trị của FR để rẽ nhánh																												
	OF	SF	ZF																										
JPL		0	0																										
JMI		1																											
JNZ			0																										
JZE			1																										
JOV	1																												
unconditional JUMP (nhảy vô điều kiện)	JUMP adr [,x]	Rẽ nhánh vô điều kiện tới địa chỉ hiệu dụng.																											

(6) Lệnh thao tác chồng

PUSH (đẩy vào)	PUSH adr [,x]	SP \leftarrow (SP) $-_L$ 1, (SP) \leftarrow địa chỉ hiệu dụng	-
POP (lấy ra)	POP r	r \leftarrow ((SP)), SP \leftarrow (SP) $+_L$ 1	

(7) Lệnh gọi và trả về

CALL subroutine (gọi chương trình con)	CALL adr [,x]	SP \leftarrow (SP) $-_L$ 1, (SP) \leftarrow (PR), PR \leftarrow địa chỉ hiệu dụng	-
RETurn from subroutine (trở về từ chương trình con)	RET	PR \leftarrow ((SP)), SP \leftarrow (SP) $+_L$ 1	

(8) Các lệnh khác

SuperVisor Call	SVC adr [,x]	Xác định dựa trên địa chỉ hiệu dụng xem như đối. Sau khi thực hiện, GR và FR không được xác định.	-
No OPeration	NOP	N/A (không áp dụng)	

Lưu ý) r, r1, r2

Tất cả đều biểu diễn cho GR. Giá trị từ GR0 tới GR7 có thể được xác định.

adr

Điều này biểu diễn cho địa chỉ. Một giá trị từ 0 tới 65,535 có thể xác định.
x

Điều này biểu diễn cho GR được dùng như thanh ghi chỉ số. Một giá trị từ

GR1 tới GR7 có thể được xác định.

[

] Dấu ngoặc vuông ([]) chỉ ra rằng đặc tả này được chứa trong dấu
ngoặc vuông có thể bị bỏ qua.

(

) Nội dung của thanh ghi hay địa chỉ được chứa trong dấu ngoặc tròn
().

Địa chỉ hiệu
dụng

Một giá trị được tạo ra bằng việc cộng, qua "cộng logic" adr và nội dung
của x, hay địa chỉ được trả tới bởi giá trị đó.

←

Điều này nghĩa là kết quả phép toán được lưu giữ trong thanh ghi hay địa
chi ở phần bên trái.

+L, -L

Cộng logic và trừ logic.

Địa chỉ hiệu

○ : Việc đặt được thực hiện.

dụng cho

○*1: Việc đặt được thực hiện, nhưng 0 được đặt là OF.

việc đặt FR

○*2: Việc đặt được thực hiện, nhưng giá trị bit được gửi từ thanh ghi được

đặt là OF.

- : Giá trị trước khi thực hiện được lưu giữ.

1.3 Tập kí tự

(1) Tập kí tự JIS X0201 Romaji/katakana dùng cách mã 8-bit được sử dụng.

(2) Một phần của tập kí tự này được vẽ trong bảng bên phải. Tám bit được dùng để biểu diễn cho một kí tự; bốn bit cao chỉ ra cột trong bảng này, còn bốn bit thấp chỉ ra hàng. Chẳng hạn, mã thập lục phân cho dấu cách, "4," "H," và "¥" là 20, 34, 48 và 5C, tương ứng. Các kí tự tương ứng với các mã thập lục phân từ 21 tới 7E (và A1 tới DF bị bỏ qua trong bảng này) được gọi là "kí tự đồ hoạ." Kí tự đồ hoạ có thể được hiển thị (in ra) như kí tự trên thiết bị đưa ra.

(3) Nếu bất kì kí tự nào không được liệt kê trong bảng này và cấu hình bit cho các kí tự đó là được cần tới, thì chúng được cho trong bài toán.

Column Row \ Row	02	03	04	05	06	07
0	Space	0	@	P	'	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	\$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(8	H	X	h	x
9)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	[k	{
12	,	<	L	¥	l	
13	-	=	M]	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

2 Đặc tả về hợp ngữ CASL II

2.1 Đặc tả ngôn ngữ

- (1) CASL II là hợp ngữ cho COMET II.
- (2) Một chương trình bao gồm các dòng lệnh và dòng chú thích.
- (3) Một lệnh được mô tả trong một dòng lệnh, và không thể tiếp tục sang dòng tiếp theo.
- (4) Các dòng lệnh và dòng chú thích được viết từ kí tự thứ nhất của dòng theo định dạng mô tả sau:

Kiểu dòng		Định dạng mô tả
Dòng lệnh	có toán hạng	[nhãn]{trống}{mã lệnh}{trống}{toán hạng}[trống][chú thích]
	không toán hạng	[nhãn]{trống}{mã lệnh}{[trống]}[{;}{chú thích}]
Dòng chú thích		[trống]{;}{chú thích}

Lưu ý) [] Dấu ngoặc vuông ([]) chỉ ra rằng đặc tả này được chứa bên trong ngoặc vuông có thể được bỏ qua.

{ } Dấu ngoặc nhọn ({ }) chỉ ra rằng đặc tả được chứa trong ngoặc nhọn là bắt buộc.

Nhãn Nhãn là tên được dùng để nói tới địa chỉ của (từ thứ nhất của) lệnh từ các lệnh khác và các chương trình. Nhãn phải có chiều dài từ 1 tới 8 kí tự, và kí tự đi đầu phải là chữ hoa. Các kí tự tiếp sau có thể là chữ hoa hay số. Các từ dành riêng, GR0 tới GR7, là không được dùng.

Trống Một hay nhiều dấu cách.

Mã lệnh Định dạng mô tả được xác định bởi lệnh.

Toán hạng Định dạng mô tả được xác định bởi lệnh.

Chú thích Thông tin tùy chọn như phần ghi nhớ có thể được viết theo bất kì kí tự nào bởi hệ thống xử lý.

2.2 Các kiểu lệnh

CASL II bao gồm bốn lệnh hợp ngữ (START, END, DS và DC), hai lệnh macro (IN và OUT) và các lệnh ngôn ngữ máy (các lệnh COMET II). Các đặc tả là như sau:

Kiểu lệnh	Nhãn	Mã lệnh	Toán hạng	Chức năng
Lệnh hợp ngữ	Nhãn	START	[Địa chỉ bắt đầu thực hiện]	Xác định địa chỉ bắt đầu để thực hiện chương trình. Xác định tên lối vào để tham chiếu trong các chương trình khác.
		END		Xác định chỗ kết thúc của chương trình.
	[nhãn]	DS	Chiều dài từ	Cấp phát miền.
	[nhãn]	DC	Hằng[, hằng]...	Định nghĩa hằng.
Lệnh macro	[nhãn]	IN	Vùng đưa vào, vùng chiều dài kí tự vào	Đưa vào dữ liệu kí tự từ thiết bị đưa vào.
	[nhãn]	OUT	Vùng đưa ra, miền chiều dài kí tự đưa ra	Đưa ra dữ liệu kí tự từ thiết bị đưa ra.
Lệnh ngôn ngữ máy	[nhãn]	(Xem "1.2 Các lệnh")		

2.3 Các lệnh hợp ngữ

Các lệnh hợp ngữ được dùng cho điều khiển hợp ngữ, v.v.

(1)	START	[Địa chỉ bắt đầu thực hiện]
-----	--------------	-----------------------------

Lệnh START xác định ra định của chương trình.

Tên nhãn được định nghĩa bên trong chương trình này xác định ra địa chỉ bắt đầu thực hiện. Nếu nhãn này được xác định, thì việc thực hiện bắt đầu từ địa chỉ này, và nếu nhãn không có, thì việc thực hiện bắt đầu từ lệnh tiếp của lệnh START.

Nhãn cho lệnh này có thể được tham chiếu tới từ các chương trình khác như tên lối vào.

(2)	END	
-----	------------	--

Lệnh END xác định là chỗ kết thúc của chương trình.

(3)	DS	Chiều dài từ
-----	-----------	--------------

Lệnh DS phân bổ một vùng với chiều dài xác định.

Chiều dài từ được xác định bởi hằng thập phân (≥ 0). Nếu "0" được xác định làm chiều dài từ của một miền, thì miền này không được phân bổ, nhưng nhãn vẫn hợp lệ.

(4)	DC	Hằng[, hằng] ...
-----	-----------	------------------

Lệnh DC lưu giữ dữ liệu đã được xác định như một hằng tính theo từ (liên tiếp).

Có bốn kiểu hằng: hằng thập phân, hằng thập lục phân, hằng kí tự và hằng địa chỉ.

Kiểu hằng	Định dạng	Mô tả lệnh
Hằng thập phân	n	Lệnh này lưu giữ giá trị thập phân được xác định bởi "n" như một từ của dữ liệu nhị phân. Nếu "n" là ngoài phạm vi $-32,768$ tới $32,767$, thì chỉ 16 bit thấp hơn của n mới được lưu giữ.
Hằng thập lục phân	#h	Giá trị "h" là số thập lục phân bốn chữ số. (Kí pháp thập lục phân dù 0 tới 9 và A tới F.) Lệnh này lưu giữ giá trị thập lục phân được xác định bởi "h" như một từ dữ liệu nhị phân. ($0000 \leq h \leq FFFF$)
Hằng kí tự	'xâu kí tự'	Lệnh này phân bổ một vùng liên tục gồm một số các kí tự (> 0) trong xâu này. Kí tự đầu tiên được lưu giữ trong các bit 8 tới 15 của từ thứ nhất, kí tự thứ hai được lưu giữ trong các bit 8 tới 15 của từ thứ hai, và cứ như vậy, sao cho dữ liệu kí tự được lưu giữ tuần tự trong bộ nhớ. Các bits 0 tới 7 của từng từ được rót đầy bằng số không. Dấu cách và nhiều kí tự đồ hoạ có thể được ghi trong một xâu kí tự. Dấu nháy đơn (') phải được viết hai lần liên tiếp.
Hằng địa chỉ	Nhãn	Lệnh này lưu giữ một địa chỉ tương ứng với tên nhãn như một từ dữ liệu nhị phân.

2.4 Các lệnh macro

Các lệnh macro dùng một nhóm các lệnh và dữ liệu đã định sẵn để sinh ra một nhóm các lệnh thực hiện một chức năng mong muốn (chiều dài từ là không xác định).

(1)	IN	Miền vào, miền chiều dài kí tự vào
-----	-----------	------------------------------------

Lệnh IN đọc một bản ghi dữ liệu kí tự từ một thiết bị đưa vào đã được phân bổ trước đó.

Toán hạng miền vào nên là nhãn cho vùng làm việc 256-từ, và dữ liệu vào trong miền này bắt đầu tại địa chỉ bắt đầu, mỗi kí tự một từ. Không lưu giữ mã định biên bản ghi (như mã xuống dòng, khi dùng bàn phím). Định dạng lưu trữ là giống như hằng kí tự với lệnh DC. Nếu dữ liệu đưa vào bé hơn 256 kí tự, thì dữ liệu trước đó sẽ còn lại trong phần còn lại của miền đưa vào. Nếu dữ liệu đưa vào vượt quá 256 kí tự, thì các kí tự vượt quá bị bỏ đi.

Miền chiều dài kí tự đưa vào nên là nhãn của miền làm việc một từ, và chiều dài kí tự là cái vào (≥ 0) được lưu giữ như dữ liệu nhị phân. Nếu chỉ báo cuối tệp được gấp phải thì -1 được lưu giữ.

Khi lệnh IN được thực hiện, thì nội dung của thanh ghi GR được cắt giữ nhưng nội dung của FR là không xác định.

(2)	OUT	Miền ra, miền chiều dài kí tự ra
-----	------------	----------------------------------

Lệnh OUT ghi dữ liệu kí tự như một bản ghi dữ liệu lên thiết bị đưa ra đã được phân bổ trước đó.

Toán hạng miền vào nên là nhãn cho vùng mà dữ liệu cần đưa ra được lưu giữ, một kí tự trên một từ.

Định dạng lưu trữ là giống như hằng kí tự với lệnh DC. Bits 0 tới 7 không phải là số không bởi vì OS bỏ qua chúng.

Miền độ dài kí tự ra nên là nhãn của miền làm việc một từ, và chiều dài kí tự là cái ra (≥ 0) được lưu giữ như dữ liệu nhị phân.

Khi lệnh OUT được thực hiện, thì nội dung của các thanh ghi GR được cất giữ nhưng nội dung của FR là không xác định.

- (3)

RPUSH	
-------	--

 Lệnh RPUSH lưu nội dung của GR trong stack theo thứ tự GR1, GR2, ... và GR7.
- (4)

RPOP	
------	--

 Lệnh RPOP lấy nội dung của một cách tuần tự, và lưu vào GR theo thứ tự GR7, GR6, ... và GR1.

2.5 Các lệnh ngôn ngữ máy

Các toán hạng của các lệnh ngôn ngữ máy được mô tả theo những định dạng sau:

r, r1, r2 GR được xác định bằng việc dùng một kí hiệu từ GR0 tới GR7.

X GR được dùng như thanh ghi chỉ số có thể được xác định bằng một kí hiệu từ GR1 tới GR7.

Adr Địa chỉ này được xác định bằng hằng thập phân, hằng thập lục phân, hằng địa chỉ hay một hằng chữ.

Hằng chữ có thể được mô tả bằng việc gắn dấu bằng (=) trước một hằng thập phân, hằng thập lục phân, hay hằng kí tự. CASL II sinh ra một lệnh DC bằng việc xác định hằng sau dấu bằng như toán hạng, và đặt địa chỉ này cho giá trị adr.

2.6 Các lệnh khác

(1) Các vị trí tương đối của từ lệnh và miền được trình hợp dịch sinh ra đều tuân theo thứ tự của các mô tả trong chương trình hợp ngữ. Tất cả các lệnh DC được sinh ra từ các hằng chữ đều được định vị ngay trước lệnh END.

(2) Các từ lệnh và miền được sinh ra sẽ chiếm một vùng liên tục trong bộ nhớ chính.

3. Hướng dẫn thực hiện chương trình

3.1 Hệ điều hành OS

Những sắp xếp sau đây cần có khi liên quan tới việc thực hiện chương trình.

(1) Trình hợp dịch diễn giải các nhãn không định nghĩa (các nhãn được viết trong cột toán hạng, là bất kì cái gì không được xác định bên trong chương trình) như một tên lối vào (nhãn lệnh START) cho chương trình khác. Trong trường hợp này, trình hợp dịch kèm việc xác định địa chỉ lại và giao phó nhiệm vụ đó cho OS. Trước khi thực hiện chương trình này, OS thực hiện việc xử lí móc nối với tên lối vào chương trình khác và xác định các địa chỉ (móc nối chương trình).

(2) Chương trình được OS cho bắt đầu. Mặc dù miền trong bộ nhớ chính nơi chương trình được nạp vào là chưa xác định, giá trị địa chỉ tương ứng với nhãn trong chương trình được OS sửa lại thành địa chỉ thực tại.

(3) Trong khi chương trình bắt đầu chạy, OS phân bổ đủ các miền chồng cho chương trình, rồi bổ sung thêm một địa chỉ cuối cùng và đặt giá trị đó trong SP.

(4) OS truyền điều khiển cho chương trình bằng lệnh CALL. Khi trả lại điều khiển cho OS sau khi thực hiện chương trình, thì lệnh RET được sử dụng.

(5) Việc phân bổ thiết bị đưa vào cho lệnh IN hay thiết bị đưa ra cho lệnh OUT được người dùng thực hiện trước khi cho thực hiện chương trình.

(6) OS giải quyết các khía cạnh có thể này sinh trong các thủ tục đưa vào và đưa ra do các thiết bị vào/ra và phương tiện có tham dự; vào/ra được thực hiện bằng việc dùng định dạng và thủ tục chuẩn (kể cả giải quyết lỗi). Do đó, người dùng của những lệnh IN và OUT này không cần bận tâm tới sự khác biệt giữa các thiết bị vào/ra.

3.2 Các khoản mục không định nghĩa

Phải đảm bảo rằng bất kì khoản mục nào liên quan tới việc thực hiện chương trình mà không được định nghĩa trong những đặc tả này đều được hệ thống xử lí giải quyết.

Trung tâm Sát hạch Công nghệ thông tin và Hỗ trợ đào tạo



<http://www.vitec.org.vn>