Operating system

A computer operating system (OS) uses paging for memory management.

In paging:

- main memory is divided into equal-size blocks, called page frames
- each process that is executed is divided into blocks of the same size, called pages
- each process has a page table that is used to manage the pages of this process

The following table is the incomplete page table for a process X.

Page	Presence flag	Page frame address	Additional data
1	1	132	
2	1	245	
3	1	232	
4	0	0	
5	1	542	
6	0	0	
⌒	⌒	⌒	⌒
135	0	0	

When a particular page of the process is currently in main memory, the Presence flag entry in the page table is set to 1.

If the page is not currently present in memory, the Presence flag is set to 0.

(a) The page frame address entry for Page 2 is 245.

State what the value 245 could represent.

This address is the start of the page ..... [1]

(b) Process X executes until the next instruction is the first instruction in Page 4. Page 4 is not currently in main memory.

State a hardware device that could be storing this page.

HDD ..... [1]

(c) When an instruction to be accessed is not present in main memory, its page must be loaded into a page frame. If all page frames are currently in use, the contents of a page frame will be overwritten with this new page.

The page that is to be replaced is determined by a page replacement algorithm.

One possible algorithm is to replace the page that has been resident in main memory for the longest time.

(i) Give the additional data that would need to be stored in the page table.

Time of entry

..... [1]

(ii) Complete the table entries below to show what happens when Page 4 is swapped into main memory. Assume that Page 5 is the one to be replaced.

In the final column, give an example of the data you have identified in **part (c)(i)**.

Page	Presence flag	Page frame address	Additional data
4	1	542	0:10:59

[3]

An alternative algorithm is to replace the page that has been used least.

(iii) Give the different additional data that the page table would now need to store.

Count the frequency of the fie being used

..... [1]

(iv) In the following table, complete the missing data to show what happens when Page 3 is swapped into main memory. Assume that Page 1 is the one to be replaced.

In the final column, give an example of the data you have identified in **part (c)(iii)**.

Page	Presence flag	Page frame address	Additional data
3	1	132	0

[3]

(d) Explain why the algorithms given in **part (c)** may not be the best choice for efficient memory management.

Longest resident Maybe this is the page which is being used most by the current application, and by taking it out, it will be needed again.

Least used The moment a new page is loaded, a page will have the frequency of 0, then this algorithm would always remove the new page that has been loaded in.

[4]

A computer operating system (OS) uses paging for memory management.

In paging:

- main memory is divided into equal-size blocks, called page frames
- each process that is executed is divided into blocks of the same size, called pages
- each process has a page table that is used to manage the pages of this process

The following table is the incomplete page table for a process, Y.

Page	Presence flag	Page frame address	Additional data
1	1	221	
2	1	222	
3	0	0	
4	0	0	
5	1	542	
6	0	0	
~	~	~	~
249	0	0	

(a) State **two** facts about Page 5.

1 Present in memory

2 It is stored at memory address 542

[2]

(b) Process Y executes the last instruction in Page 5. This instruction is not a branch instruction.

(i) Explain the problem that now arises in the continued execution of process Y.

To execute the next instruction, the file of page 6 has to be loaded into the main memory. As its current presence value is 0, that means it has not been loaded into the RAM yet.

[2]

(ii) Explain how interrupts help to solve the problem that you explained in **part (b)(i)**.

We couldn't continue execution as the next page had not been loaded in.

With interrupts, as soon as page 5 is executed, it will send an interrupt to the processor. This will cause the next page to be loaded in.

[3]

(c) When the next instruction is not present in main memory, the OS must load its page into a page frame. If all page frames are currently in use, the OS overwrites the contents of a page frame with the required page.

The page that is to be replaced is determined by a page replacement algorithm.

One possible algorithm is to replace the page which has been in memory the shortest amount of time.

(i) Give the additional data that would need to be stored in the page table.

Time of entry

[1]

(ii) Complete the table entry below to show what happens when Page 6 is swapped into main memory. Include the data you have identified in **part (c)(i)** in the final column. Assume that Page 1 is the one to be replaced.

In the final column, give an example of the data you have identified in **part (c)(i)**.

Page	Presence flag	Page frame address	Additional data
6	1	221	10:10:45

[3]

Process Y contains instructions that result in the execution of a loop, a very large number of times. All instructions within the loop are in Page 1.

The loop contains a call to a procedure whose instructions are all in Page 3.

All page frames are currently in use. Page 1 is the page that has been in memory for the shortest time.

(iii) Explain what happens to Page 1 and Page 3, each time the loop is executed.

When page 1 is completely executed, page 3 is placed into main memory and page 1 is swapped out. At the end of page 3's code, it calls in page 1 again. This process is repeated. [3]

(iv) Name the condition described in **part (c)(iii)**.

Disk Thrashing [1]

A compiler uses a keyword table and a symbol table. Part of the keyword table is shown.

- Tokens for keywords are shown in hexadecimal.
- All of the keyword tokens are in the range 00 – 5F.

Keyword	Token
←	01
+	02
=	03
<>	04
IF	4A
THEN	4B
ENDIF	4C
ELSE	4D
REPEAT	4E
UNTIL	4F
TO	50
INPUT	51
OUTPUT	52
ENDFOR	53

Entries in the symbol table are allocated tokens. These values start from 60 (hexadecimal).

Study the following piece of pseudocode.

```
Counter ← 0
INPUT Password
REPEAT
   IF Password <> "Cambridge"
      THEN
         INPUT Password
   ENDIF
   Counter ← Counter + 1
UNTIL Password = "Cambridge"
OUTPUT Counter
```

(a) Complete the symbol table to show its contents after the lexical analysis stage.

Symbol	Token	
	Value	Type
Counter	60	Variable
0	61	Constant
PASSWORD	62	Variable
"Cambridge"	63	String
1	64	Constant

[3]

(b) The output from the lexical analysis stage is stored in the following table. Each cell stores one byte of the output.

Complete the output from the lexical analysis using the keyword table **and** your answer to **part (a)**.

60	01	61	51	62	4E	4A	62	04	63	4B	51	62	4C	60	01	60	02	64	4F	62	03	63	52	60

[2]

(c) The following table shows assembly language instructions for a processor which has one general purpose register, the Accumulator (ACC).

Instruction		Explanation
Op code	**Operand**	
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC
ADD	<address>	Add the contents of the given address to the ACC
STO	<address>	Store the contents of ACC at the given address

After the syntax analysis is completed successfully, the compiler generates object code.

The following lines of high level language code are compiled.

```
X = X + Y
Z = Z + X
```

The compilation produces the assembly language code as follows:

```
LDD 236
ADD 237
STO 236
LDD 238
ADD 236
STO 238
```

(i) The final stage in the compilation process that follows this code generation stage is code optimisation.

Rewrite the equivalent code after optimisation.

LDD 236

ADD 237

STO 236

ADD 238

STO 238

[3]

(ii) Explain why code optimisation is necessary.

Less instructions to execute

Occupies less space in main memory

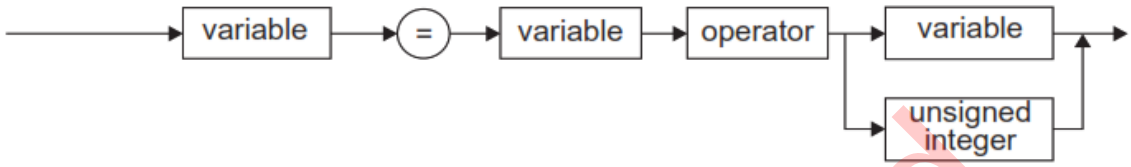
Reduced execution time

[2]

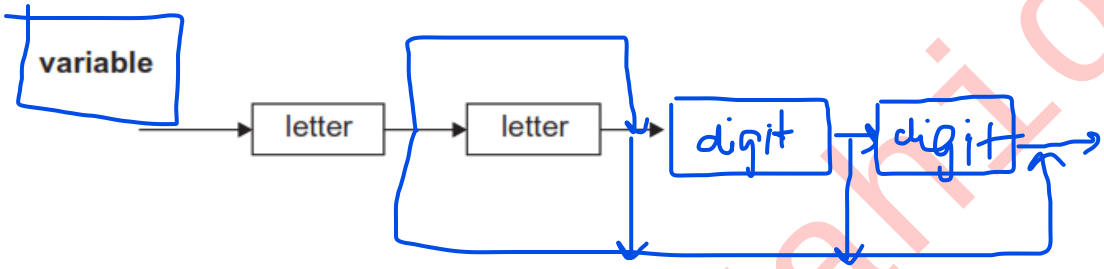
5 The following syntax diagrams for a programming language show the syntax of:

- an assignment statement
- a variable
- an unsigned integer
- a digit
- a letter
- an operator

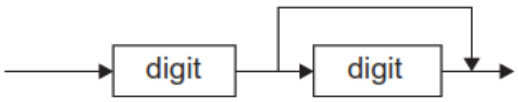
assignment statement



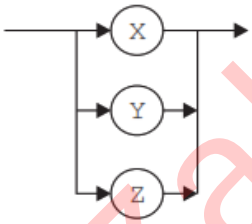
variable



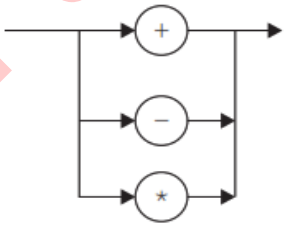
unsigned integer



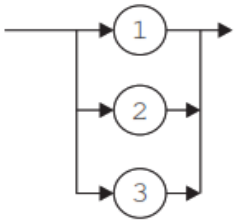
letter



operator



digit



(a) Give reasons why each of these statements is **invalid**.

```
X = XY + 21
```

X is not a variable as it doesn't have 2 letters

```
YZ := YZ * 3
```

Colon punctuation is no defined in the diagram

```
XY = XY - 5
```

The only digits are 1, 2, 3

[3]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagrams shown.

<letter> has been completed for you.

<letter> ::= X|Y|Z

<assignment_statement> ::=

<variable> = <variable><operator><variable>|<variable><operator><unsigned_integer>

<variable> ::=

<letter><letter>

<digit> ::=

1|2|3

<unsigned_integer> ::=

<digit>|<digit><digit>

<operator> ::=

+|-|*

[5]

(c) The syntax of a **variable** is changed to allow one or two letters followed by zero, one or two digits.

(i) Draw an updated syntax diagram for the **variable**.

[3]

(ii) Give the BNF for the revised **variable**.

[3]

6 Duraid writes a short program in a high-level programming language. An interpreter executes the program.

The following is part of Duraid's program.

```
DECLARE P, Q, R, X, Y : INTEGER
CONSTANT M = 10

P = 4
Q = 2
R = 1
X = (P + Q) * (P - Q)
Y = (M / Q) * (P + Q - R)
```

(a) Write the Reverse Polish Notation (RPN) for the following expression from Duraid's program.

```
(P + Q) * (P - Q)
```

PQ+PQ-* ..... [2]

(b) The interpreter is executing Duraid's program. The expressions are in infix form.

The interpreter converts the infix to RPN.

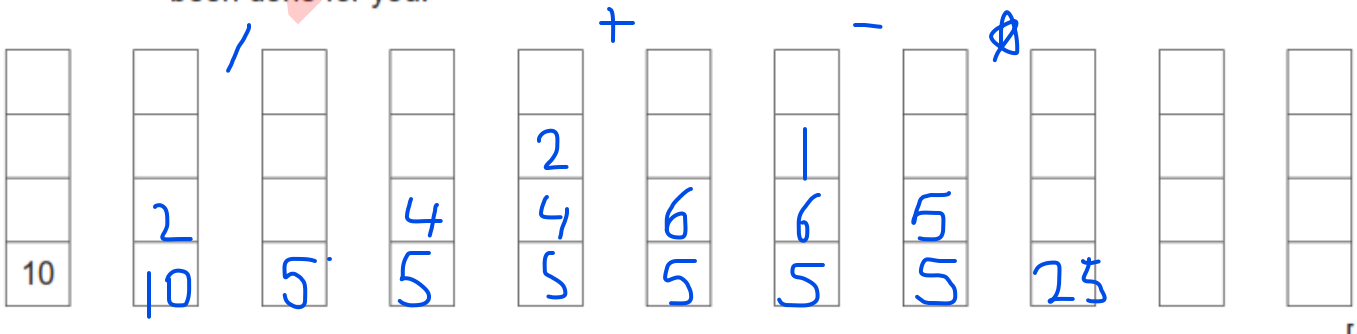
The RPN expression for Y is:

```
M Q / P Q + R - *
```

The interpreter evaluates this RPN expression using a stack.

(i) Show the changing contents of the stack, as the interpreter evaluates the expression for Y.

Use the values of the variables and constant given in the program. The first entry has been done for you.



[4]

(ii) Convert the following RPN expression back to its infix form.

P Q + M * R P - -

(P+Q) * M - (R-P)

..... [2]

(c) Explain how RPN is used by an interpreter to evaluate expressions.

Expressions are solved left to right.

Each operator uses the last 2 values

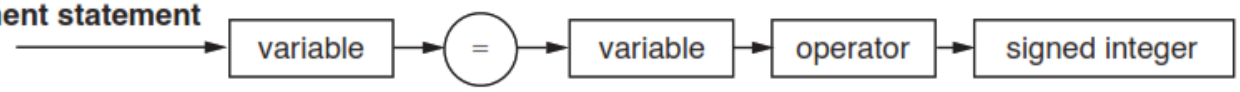
Each operator is applied to the top 2 values

..... [2]

5 The following syntax diagrams show the syntax of:

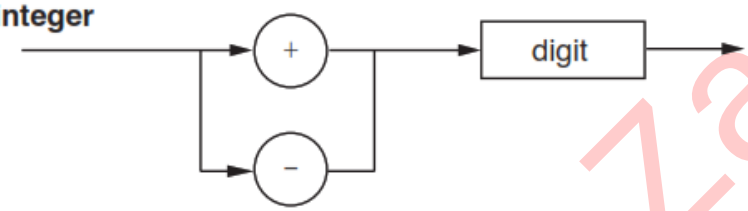
- an assignment statement
- a variable
- a signed integer
- a letter
- a digit
- an operator

assignment statement

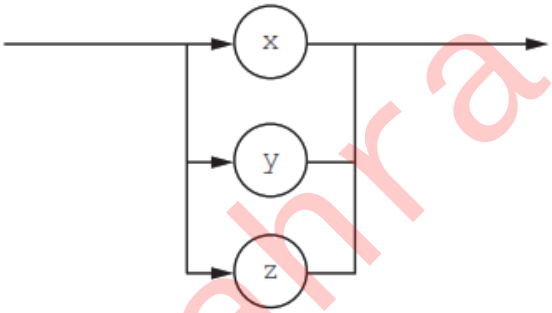


variable

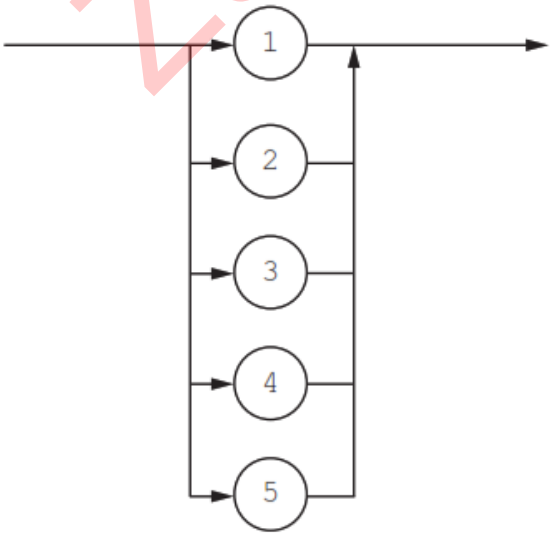
signed integer



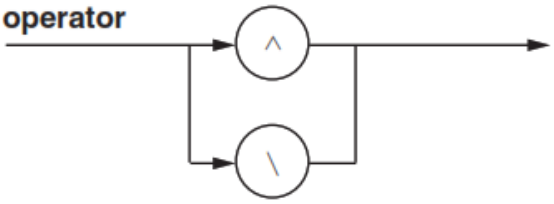
letter



digit



operator



(a) The following assignment statements are invalid.

Give the reason in each case.

(i) `xy = xy ^ c4`

Reason .....

.....[1]

(ii) `zy = zy \ 10`

Reason .....

.....[1]

(iii) `yy := xz ^ - 6`

Reason .....

.....[1]

(b) Complete the Backus-Naur Form (BNF) for the syntax diagrams on the opposite page.

<assignment statement> ::=

.....

<variable> ::=

.....

<signed integer> ::=

.....

<operator> ::=

.....

[4]

(c) Rewrite the BNF rule for a variable so that it can be any number of letters.

<variable> ::=

.....[2]