Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTER SCIENCE **9608/42**

Paper 4 Further Problem-solving and Programming Skills **May/June 2019**

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **19** printed pages and **1** blank page.

Cambridge Assessment International Education

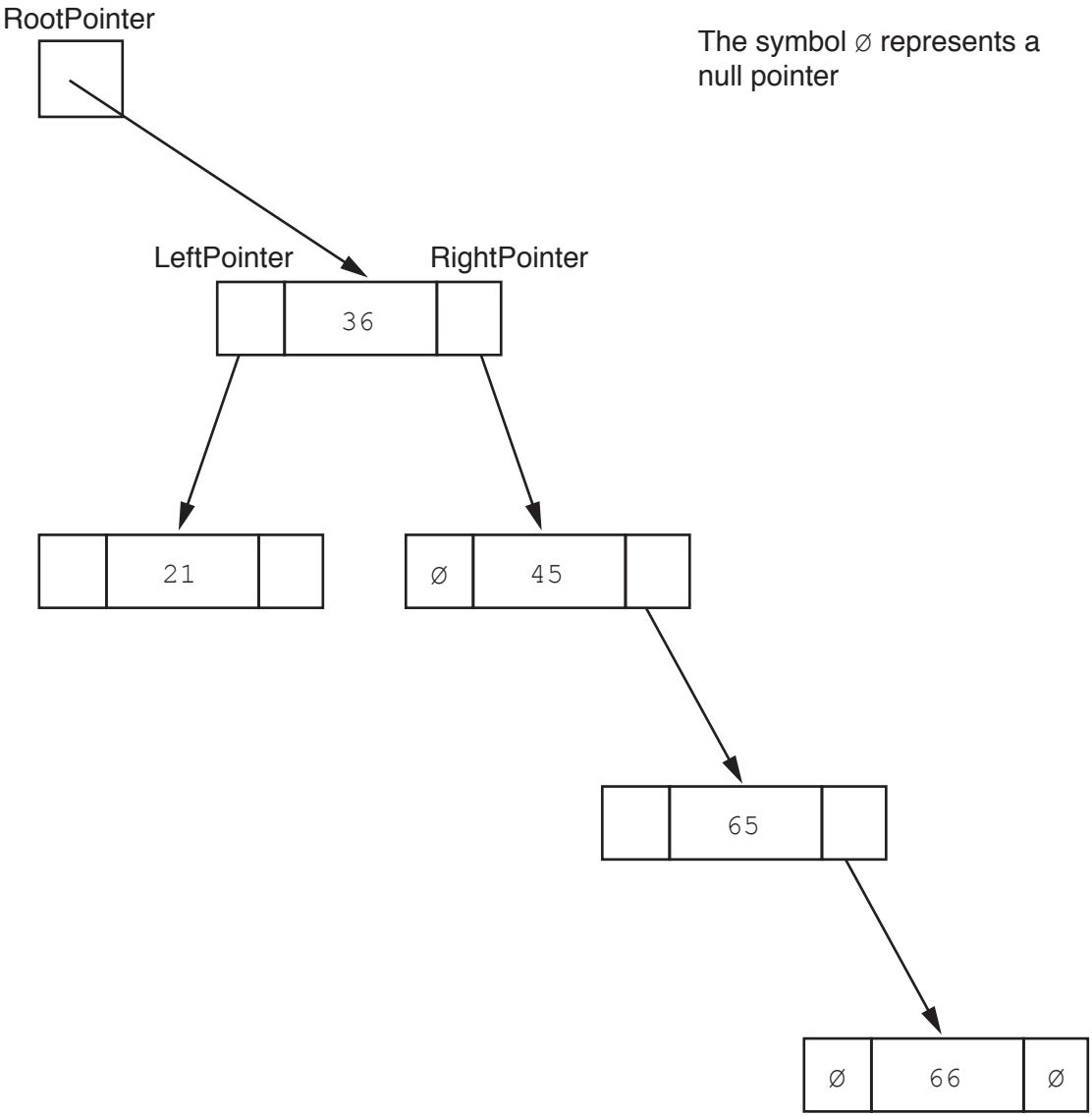
1 A company wants an online marking system for an examination.

(a) The following is a selection of data showing final marks.

```
36, 45, 21, 65, 66, 13, 54, 53, 34
```

A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.

(i) Complete the binary tree to show how the data above will be organised.



[5]

(ii) The following diagram shows a 2D array that stores the nodes of the binary tree's linked list.

Add the correct pointer values to complete the diagram, using your answer from **part (a)(i)**.

RootPointer

0

FreePointer

Index	LeftPointer	Data	RightPointer
0		36	
1		45	
2		21	
3		65	
4		66	
5		13	
6		54	
7		53	
8		34	
9			

[6]

[Turn over

(b) The company wants to implement a program for the marking system. It will do this with object-oriented programming (OOP).

Many candidates take the examination. Each examination paper is given a `PaperID` that is made up of the centre (school) number followed by the candidate number.

Each examination paper is awarded a grade.

The following diagram shows the design for the `ExaminationPaper` class. This includes the attributes and methods.

ExaminationPaper

`FinalMark : INTEGER // maximum 2 digits, initialised to 0`
`Grade : STRING // "Pass", "Merit", "Distinction"`
`// or "Fail", initialised to "Fail"`
`PaperID : STRING // centre number followed by the`
`// candidate number, for example`
`// "ZZ00991001"`

`Create() // creates and initialises a new instance`
`// of the ExaminationPaper class using`
`// language-appropriate constructor`
`SetFinalMark() // checks that the mark parameter has a`
`// valid value, if so, assigns it to`
`// FinalMark`
`SetGrade() // sets Grade based on FinalMark`
`GetFinalMark() // returns FinalMark`
`GetGrade() // returns Grade`
`GetPaperID() // returns PaperID`

(i) The constructor receives the centre number and candidate number as parameter values to create `PaperID`. Other properties are initialised as instructed in the class diagram.

Write **program code** for the `Create()` constructor method.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(ii) Get and set methods are used to support the security and integrity of data in object-oriented programming.

Explain how get and set methods are used to support security and integrity.

.....

.....

.....

.....

.....

..... [3]

(iii) Write **program code** for the following three get methods.

Programming language .....

`GetFinalMark()`

Program code

.....

.....

.....

.....

`GetGrade()`

Program code

.....

.....

.....

.....

`GetPaperID()`

Program code

.....

.....

.....

.....

[4]

(iv) The method `SetFinalMark()` checks that its `INTEGER` parameter `Mark` is valid. It is then set as the final mark if it is valid. A valid mark is greater than or equal to 0 and less than or equal to 90.

If the mark is valid, the method sets the final mark and returns `TRUE`.
If the mark is not valid, the method does not set the final mark and returns `FALSE`.

Write **program code** for `SetFinalMark(Mark : INTEGER)`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(v) Write **program code** for the method:

```
SetGrade(DistMark, MeritMark, PassMark : INTEGER)
```

Use the properties in the original class definition.

Grades are awarded as follows:

Grade	Criteria
Distinction	`>= DistMark`
Merit	`>= MeritMark`
Pass	`>= PassMark`
Fail	`< PassMark`

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(vi) Emily is a candidate who has taken the examination paper. The grades are awarded as follows:

Grade	Criteria
Distinction	`>= 80`
Merit	`>= 70`
Pass	`>= 55`

The procedure `Main()` performs the following tasks.

- allows the centre number, candidate number and mark to be input, with suitable prompts
- assigns an instance of `ExaminationPaper` to the variable `ThisPaper`
- sets the mark for the object
- sets the grade for the object
- outputs the grade for the object

Write **program code** for the `Main()` procedure.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(c) The examination paper will be taken by many candidates in centres around the world.

The program stores the objects of the `ExaminationPaper` class in a file. The company has decided to use a hash table, rather than a linked list to store the objects.

Explain why a hash table is more suitable than a linked list to store the objects.

.....

.....

.....

.....

.....

.....

.....

..... [4]

Question 2 begins on the next page.

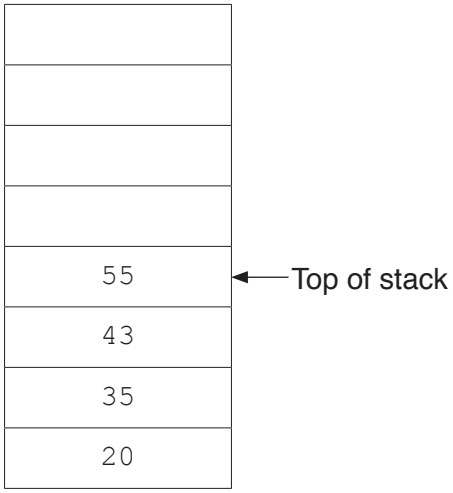
2 A stack is an Abstract Data Type (ADT).

(a) Tick (✓) **one** box to show the statement that describes a stack data structure.

Statement	Tick (✓)
Last in first out	
First in first out	
Last in last out	

[1]

(b) A stack contains the values `20`, `35`, `43`, `55`.

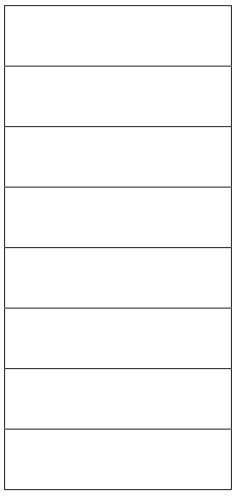


(i) Show the contents of the stack in **part (b)** after the following operations.

```
POP()

POP()

PUSH(10)
```



[1]

(ii) Show the contents of the stack from **part (b)(i)** after these further operations:

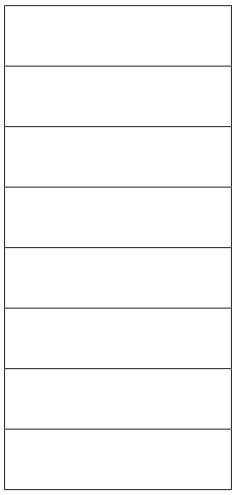
```
POP()

PUSH(50)

PUSH(55)

POP()

PUSH(65)
```



[1]

(iii) The stack is implemented as a 1D array, with eight elements, and given the identifier `ArrayStack`.

The global variable `Top` contains the index of the last element in the stack, or –1 if the stack is empty.

The function `Push()`:

- takes as a parameter an `INTEGER` value to place on the stack
- adds the value to the top of the stack and returns `TRUE` to show that the operation was successful
- returns `FALSE` if the stack is full.

Write an algorithm in **pseudocode** for the function `Push()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

3 **(a)** Identify **and** describe **two** features of an editor that can help a programmer to write program code.

Feature 1 .....

Description .....

.....

.....

Feature 2 .....

Description .....

.....

.....

[4]

(b) A programmer can use three types of test data when testing a program.

Identify the **three** different types of test data.

1 .....

2 .....

3 .....

[3]

4 **(a)** A program has sorted some data in the array, `List`, in ascending order.

The following binary search algorithm is used to search for a value in the array.

```
01   ValueFound ← FALSE
02   UpperBound ← LengthOfList - 1
03   LowerBound ← 0
04   NotInList ← FALSE
05
06   WHILE ValueFound = FALSE AND NotInList = FALSE
07      MidPoint ← ROUND((LowerBound + UpperBound) / 2)
08
09      IF List[LowerBound] = SearchValue
10         THEN
11            ValueFound ← TRUE
12         ELSE
13            IF List[MidPoint] < SearchValue
14               THEN
15                  UpperBound ← MidPoint + 1
16               ELSE
17                  UpperBound ← MidPoint - 1
18            ENDIF
19            IF LowerBound > MidPoint
20               THEN
21                  NotInList ← TRUE
22            ENDIF
23      ENDIF
24   ENDWHILE
25
26   IF ValueFound = FALSE
27      THEN
28         OUTPUT "The value is in the list"
29      ELSE
30         OUTPUT "The value is not found in the list"
31   ENDIF
```

Note:

The pseudocode function

`ROUND(Reall : REAL) RETURNS INTEGER`

rounds a number to the nearest integer value.

For example: `ROUND(4.5)` returns `5` and `ROUND(4.4)` returns `4`

(i) There are four errors in the algorithm.

Write the line of code where an error is present **and** write the correction in **pseudocode**.

Error 1 .....

Correction .....

Error 2 .....

Correction .....

Error 3 .....

Correction .....

Error 4 .....

Correction .....
[4]

(ii) A binary search is one algorithm that can be used to search an array.

Identify another searching algorithm.

..... [1]

(b) The following is an example of a sorting algorithm. It sorts the data in the array `ArrayData`.

```
01  TempValue ← ""
02  REPEAT
03     Sorted ← TRUE
04     FOR Count ← 0 TO 4
05        IF ArrayData[Count] > ArrayData[Count + 1]
06           THEN
07              TempValue ← ArrayData[Count + 1]
08              ArrayData[Count + 1] ← ArrayData[Count]
09              ArrayData[Count] ← TempValue
10              Sorted ← FALSE
11        ENDIF
12     ENDFOR
13  UNTIL Sorted = TRUE
```

(i) Complete the trace table for the algorithm given in **part (b)**, for the `ArrayData` values given in the table.

Count	TempValue	Sorted	ArrayData					
			0	1	2	3	4	5
			5	20	12	25	32	29

[4]

(ii) Rewrite lines 4 to 12 of the algorithm in **part (b)** using a `WHILE` loop instead of a `FOR` loop.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

(iii) Identify the algorithm shown in **part (b)**.

..... [1]

(iv) Identify another sorting algorithm.

..... [1]

BLANK PAGE