Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER				

COMPUTER SCIENCE **9608/43**

Paper 4 Further Problem-solving and Programming Skills **October/November 2020**

2 hours

You must answer on the question paper.

No additional materials are needed.

INSTRUCTIONS

- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **20** pages. Blank pages are indicated.

[Turn over

1 A company is implementing a new software system for their latest product.

Developing the software will include the following activities:

Activity	Description	Time to complete (weeks)	Predecessor
A	Identify requirements	1	–
B	Produce design	3	A
C	Write code	6	B
D	Test modules	4	B
E	Integration testing	2	D
F	Final system black-box testing	2	E
G	Install software	1	F
H	Acceptance testing	2	G
I	Create user documentation	2	G
J	Create training documents	3	G
K	Pilot implementation	4	H

Complete the GANTT chart to correspond with the given table.

A																				
B																				
C																				
D																				
E																				
F																				
G																				
H																				
I																				
J																				
K																				
Week number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

[5]

2 There are several different types of testing.

(a) Tick (✓) **one or more** boxes in each row to indicate whether each statement applies to **Integration**, **Acceptance**, **Alpha** or **Beta** testing.

Statement	Integration	Acceptance	Alpha	Beta
Software is tested in-house by dedicated testers				
Software is tested by the client before it is signed off				
Software is tested by combining modules that have previously been tested to check they work as expected				
Software is tested using normal, abnormal and boundary data				
Software is tested by releasing it to selected customers, who test it in normal circumstances				

[4]

(b) Identify **one** other method of testing.

..... [1]

3 A programmer is creating a program for a puzzle competition.

The programmer has designed the `PuzzlePlayer` class to store details for each player, including the player's score.

The following diagram shows the design for the `PuzzlePlayer` class.

PuzzlePlayer

`PlayerID : STRING // initialised in constructor to "PL12a3"`
`Name : STRING // initialised in constructor to ""`
`Score : INTEGER // initialised in constructor to 0`

`Constructor() // method used to create an instance of the`
`// PuzzlePlayer class and initialise its attributes`
`GetPlayerID() // returns PlayerID`
`GetName() // returns Name`
`GetScore() // returns Score`
`SetPlayerID() // validates the parameter value to make sure it`
`// is 6 characters in length and starts with`
`// "PL", then sets PlayerID to this value`
`SetName() // sets the Name to the parameter value`
`SetScore() // sets the Score to the parameter value`

(a) Write **program code** for the `Constructor()` method.

Use the appropriate constructor method for your chosen programming language.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [2]

(b) Write program code for `GetPlayerID()` method.

Programming language .....

Program code

.....

.....

.....

.....

.....

..... [2]

(c) The method `SetPlayerID()` validates the parameter value. It checks that it is 6 characters in length and that the first two characters are `"PL"`. The method sets `PlayerID` to the parameter value. It returns `TRUE` if the parameter value is valid and `FALSE` if it is not valid.

The function `Length(Variable)` returns the length of the string stored in `Variable` as an integer.

The function `Substring(Variable, StartingCharacter, NumberOfCharacters)` can be used to return one or more characters from a string. The first character position in a string is 0.

For example, when the string `"Computer"` is stored in the variable `Message`:

`Substring(Message,1,1)` would return the character `"o"`.

Write **pseudocode** for the `SetPlayerID()` method.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

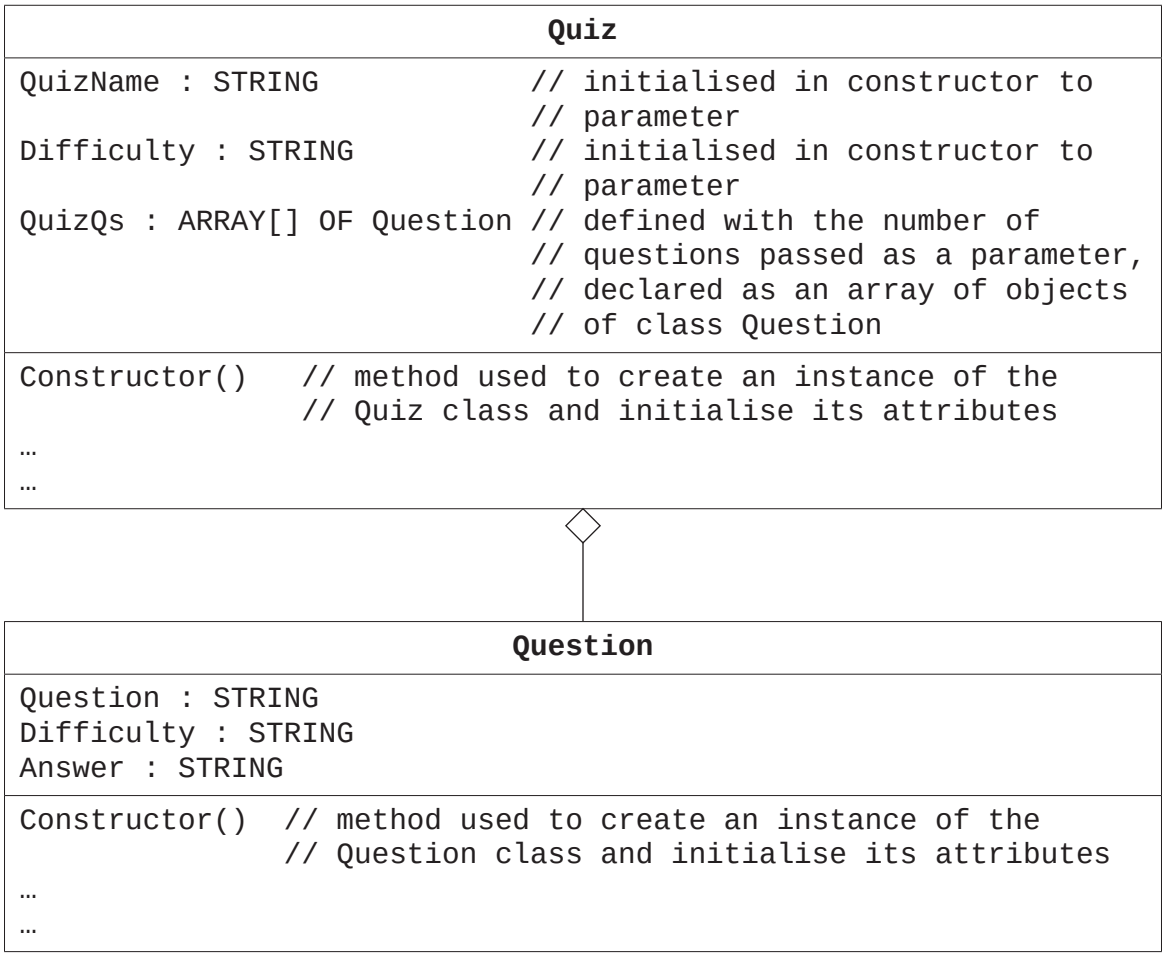
.....

..... [5]

(d) The program uses object-oriented programming to store the puzzles.

One type of puzzle is a quiz that consists of several questions. The two classes, `Quiz` and `Question`, are defined.

The class diagram shows parts of these classes.



(i) Explain what is meant by **containment**, using an example from the class diagram.

.....

.....

.....

..... [2]

(ii) Classes, objects and containment are all features of object-oriented programming.

Identify **and** describe **one other** feature of object-oriented programming.

.....

.....

.....

..... [2]

(iii) The main program stores the collection of quizzes in the array `QuizBank` that can store 100 objects.

Define the array `QuizBank` using **pseudocode**.

.....

..... [1]

(iv) A new quiz is created with the name 'Famous people'. The difficulty level is 'Low' and it consists of 10 questions.

Write **program code** to declare the quiz object and store it in the first element in `QuizBank`.

Programming language .....

Program code

.....

..... [2]

4 A declarative programming language is used to represent the following knowledge base.

```
01 type(cheddar).
02 type(brie).
03 type(paneer).
04 type(parmesan).
05 country(england).
06 country(france).
07 country(india).
08 country(italy).
09 hard(parmesan).
10 soft(brie).
11 strong_smell(brie).
12 strong_smell(cheddar).
13 origin(brie, france).
```

These clauses have the following meanings:

Clause	Meaning
02	Brie is a type of cheese
06	France is a country
09	Parmesan is a hard cheese
10	Brie is a soft cheese
12	Cheddar has a strong smell
13	Brie is from France

(a) Camembert is a type of soft cheese from France and has a strong smell.

Write additional clauses to represent this information.

14 .....

15 .....

16 .....

17 .....

[4]

(b) Stilton (X) could be from England (Y) and Stilton could be a hard cheese, if Stilton is a type of cheese, England is a country and Stilton is not a soft cheese.

Write this as a rule.

```
Cheese_Question(X, Y)

IF .....

.....

..... [4]
```

5 There are several sorting algorithms. One type of sorting algorithm is an insertion sort.

(a) Explain how an insertion sort puts a set of data into ascending order.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

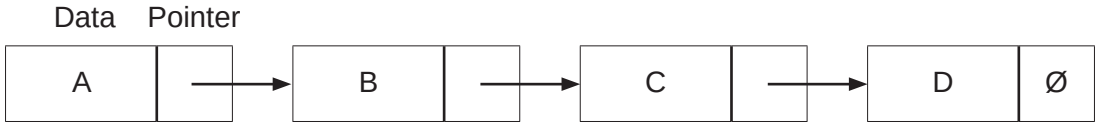
(b) The following algorithm performs a bubble sort. It is currently incomplete.

Complete the algorithm.

```
Counter ← NumberOfItems – 2
REPEAT
    DataSwapped ← FALSE
    FOR CurrentValue ← 0 TO ……
        IF DataList[CurrentValue] > DataList[CurrentValue + 1]
            THEN
                ValueTemp ← DataList[……]
                DataList[CurrentValue] ← DataList[CurrentValue + 1]
                DataList[CurrentValue + 1] ← ……
                DataSwapped ← ……
        ENDIF
    ENDFOR
UNTIL DataSwapped = ……
```

[5]

6 Consider the following diagram that represents a linked list:



The symbol Ø represents a null pointer.

(a) A new node with the data value E is added between the nodes that have the data values B and C.

Show the state of the linked list after the node with the data value E is added.

[2]

(b) State why the node with the data value D has a null pointer.

.....

..... [1]

(c) A 1D array, `LinkedList`, is used to implement the linked list. The array is declared as a record data type with two fields, `Data` and `Pointer`.

The global variable `StartPointer` stores the index of the first node in the list.

(i) The following pseudocode algorithm finds a value in a linked list. The algorithm is incomplete.

The function `FindValue()`, takes as a parameter, the value to be searched for in the linked list. The function follows the pointers in the linked list. It either returns `-1` if the value is not found, or it returns the pointer to the value if it is found.

Complete the algorithm.

```
FUNCTION FindValue(Value : INTEGER) ……

   DECLARE CurrentPointer : INTEGER

   CurrentPointer ← StartPointer

   WHILE …… <> NULL

        AND LinkedList[CurrentPointer]. …… <> ……

        CurrentPointer ← LinkedList[……].Pointer

   ENDWHILE

   IF LinkedList[CurrentPointer].Data = Value

      THEN

         RETURN ……

      ELSE

         RETURN -1

   ENDIF

ENDFUNCTION
```

[6]

(ii) The function `DeleteNode()` takes the data to be removed from `LinkedList` as a parameter.

The function starts at the first node and follows the pointers to check the data in each node.

If the data is found, it removes the node containing that data, updates the pointer and returns TRUE. Otherwise, it returns FALSE.

Write **pseudocode** for `DeleteNode()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

BLANK PAGE

7 The following table shows part of the instruction set for a processor that has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the Index Register. Copy the contents of ACC to this calculated address.
ADD	<address>	Add the contents of the given address to ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

The assembly language program in the table on the opposite page allows a username to be input as a string of up to 8 characters in length.

For strings of less than 8 characters, the user enters the exclamation mark (!) character to indicate that no more characters will be entered. The exclamation mark is not saved as part of the username.

The program then outputs each character of the username in the order input.

The program will use consecutive memory locations, starting at the address labelled USERNAME, storing one character in each location.

17

The program in the table is incomplete. The comment column contains descriptions for some program instructions.

Complete the program using the given instruction set.

Label	Instruction		Comment
	Op code	Operand	
	LDR	#0	
	LDM	#0	// initialise LENGTH to 0
	STO	LENGTH	
LOOP:	IN		
			// is character = EXCLAMATION(!)?
			// if TRUE, jump to OUTPUT
			// store character in USERNAME + contents of IX
	INC	IX	// increment Index Register
			// increment LENGTH
			// is LENGTH = MAX ?
			// if FALSE, jump to LOOP
OUTPUT:	LDR	#0	
			// initialise COUNT to 0
	LDX	USERNAME	
	OUT		
	INC	IX	
	LDD	COUNT	
	INC	ACC	// increment COUNT
	STO	COUNT	
			// is COUNT = LENGTH ?
	JPN	OUTPUT	
	END		// end program
LENGTH:			
EXCLAMATION:	B0010001		
MAX:	8		
COUNT:			
USERNAME:			

[8]

[Turn over

8 Recursion can be used when writing computer programs.

Consider the following pseudocode algorithm.

```
01 FUNCTION NumberPattern(Value1, Value2, EndValue : INTEGER) RETURNS INTEGER
02    OUTPUT Value1
03    IF Value1 <= EndValue
04       THEN
05          Temp ← Value2
06          Value2 ← Value1
07          Value1 ← Value1 + Temp
08          RETURN NumberPattern(Value1, Value2, EndValue) + 1
09       ELSE
10          RETURN 0
11    ENDIF
12 ENDFUNCTION
```

(a) State the line number in the pseudocode algorithm that shows function `NumberPattern()` is recursive. Justify your choice.

Line number .....

Justification .....

.....

.....

[2]

(b) The function is called as follows:

```
NumberPattern(1,1,12)
```

Dry run the algorithm and complete the following trace table. State the final value returned.

Show your working.

Value1	Value2	Temp	EndValue	OUTPUT	RETURN value

Final value returned .....

Working

.....

.....

.....

.....

.....

.....

..... [5]

(c) State the purpose of the algorithm.

.....

..... [1]

BLANK PAGE