Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME			
CENTRE NUMBER	□□□□□	CANDIDATE NUMBER	□□□□

0251529316

COMPUTER SCIENCE **9608/42**

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.

1 There are several different searching and sorting algorithms.

(a) Identify **two** sorting algorithms.

1 ....

2 ....
[2]

(b) Consider the following pseudocode algorithm.

```
LowerBound ← 0
UpperBound ← LengthOfList - 1
ValueFound ← FALSE
OUTPUT "Value to find: "
INPUT ValueToFind
WHILE ValueFound = FALSE AND UpperBound <> LowerBound
    MidPoint ← (LowerBound + UpperBound)DIV 2
    IF List[MidPoint] = ValueToFind
        THEN
            ValueFound ← TRUE
        ELSE
            IF List[MidPoint] < ValueToFind
                THEN
                    LowerBound ← MidPoint + 1
                ELSE
                    UpperBound ← MidPoint - 1
            ENDIF
    ENDIF
ENDWHILE
IF ValueFound = FALSE
    THEN
        MidPoint ← (LowerBound + UpperBound) DIV 2
        IF List[MidPoint] = ValueToFind
            THEN
               OUTPUT "Item in position " & MidPoint & " in list"
            ELSE
               OUTPUT "Not in list"
        ENDIF
    ELSE
        OUTPUT "Item in position " & MidPoint & " in list"
ENDIF
```

Note: `DIV` is an operator that performs integer division.

The array `List` contains the following values:

2, 5, 21, 25, 36, 48, 51, 59, 65, 70

(i) Complete the trace table to show a dry run of the algorithm, when the value 21 is input.

`LowerBound`	`UpperBound`	`ValueFound`	`ValueToFind`	`MidPoint`

[3]

(ii) Identify this type of searching algorithm.

.... [1]

(iii) The value 59 is input.

State the number of times the while loop condition is executed.

.... [1]

(iv) State the minimum number of times the while loop condition will be executed to search for a value.

.... [1]

(v) `MidPoint` is calculated and checked again after the while loop is terminated.

Explain why this additional calculation and check is necessary.

....

....

....

.... [2]

(vi) A new data set is used as follows:

5, 9, 10, 12, 15, 13, 17, 19, 20, 2

Explain why the algorithm will not find the value 2 in this data set.

....

....

....

.... [2]

BLANK PAGE

2 A company is developing a new puzzle game application for a mobile phone.

The development includes the following activities:

Activity	**Description**	**Time taken in weeks**	**Predecessor**
A	Identify requirements	2	–
B	Produce design	3	A
C	Focus group feedback on design	2	B
D	Program 5 levels	4	C
E	Graphics development	6	C
F	Focus group testing on 5 levels	2	D
G	Program remaining levels	5	D
H	Combine modules	2	G
I	White-box testing	2	H
J	Black-box testing	2	H
K	User testing	2	H
L	Beta release	2	K

(a) Complete the GANTT chart for the given activities.

A																						
B																						
C																						
D																						
E																						
F																						
G																						
H																						
I																						
J																						
K																						
L																						
Week number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

[5]

(b) **(i)** State the activities that form the critical path.

.... [1]

(ii) Give **three** activities that can run in parallel.

.... [1]

(c) Identify another project planning technique that could be used when developing the puzzle game.

.... [1]

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

```
01 person(jordan).
02 person(gita).
03 person(paolo).
04 person(cassie).
05 animal(cat).
06 animal(hamster).
07 animal(gecko).
08 animal(fish).
09 has_pet(cassie, gecko).
10 has_pet(paolo, fish).
11 has_pet(cassie, cat).
12 has_pet(jordan, cat).
```

These clauses have the following meanings:

Clause	Meaning
`01`	Jordan is a person
`05`	A cat is an animal
`09`	Cassie has a pet gecko

(a) Clive is a person who has a pet guinea pig and a pet gecko.

Write additional clauses to represent this information.

13 ....

14 ....

15 ....

16 ....

[4]

(b) Using the variable `PetOwner`, the goal:

```
has_pet(PetOwner, cat)
```

returns:

```
PetOwner = cassie, jordan
```

Write the result that is returned by the goal:

```
has_pet(cassie, PetAnimals)
```

PetAnimals = ...

.... [1]

(c) Z wants a pet Y, if Z is a person and Y is an animal and Z does not have a pet Y.

Write this as a rule.

wants_pet(.. , ...)

IF ....

....

.... [5]

4 A programmer wants to create a mobile application to record the number of calories a person eats in a day.

The programmer has designed the class, `FoodItem`, to store details for each item of food.

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

```
                              FoodItem
FoodID   : STRING  // initialised in constructor to parameter value
Name     : STRING  // initialised in constructor to an empty string
Calories : INTEGER // initialised in constructor to 0

Constructor()      // method used to create an instance of the
                   // FoodItem class and initialise its attributes
GetFoodID()        // returns FoodID
GetName()          // returns Name
GetCalories()      // returns Calories
SetFoodID()        // sets the FoodID to the parameter value
SetName()          // sets the Name to the parameter value
SetCalories()      // validates the parameter value to make sure
                   // it is a positive integer less than 2000, and
                   // then sets Calories to this value
```

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

Use the appropriate constructor method for your chosen programming language.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [3]

(b) Write **program code** for the `GetCalories()` method.

Programming language ....

Program code

....

....

....

....

....

.... [2]

(c) The method `SetCalories()` validates the integer parameter value that is passed to it. It checks that the value is positive and is less than 2000.

The method sets `Calories` to the parameter value and returns `TRUE` if the parameter value is valid. It returns `FALSE` if the parameter value is not valid.

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

....

....

....

....

....

....

....

....

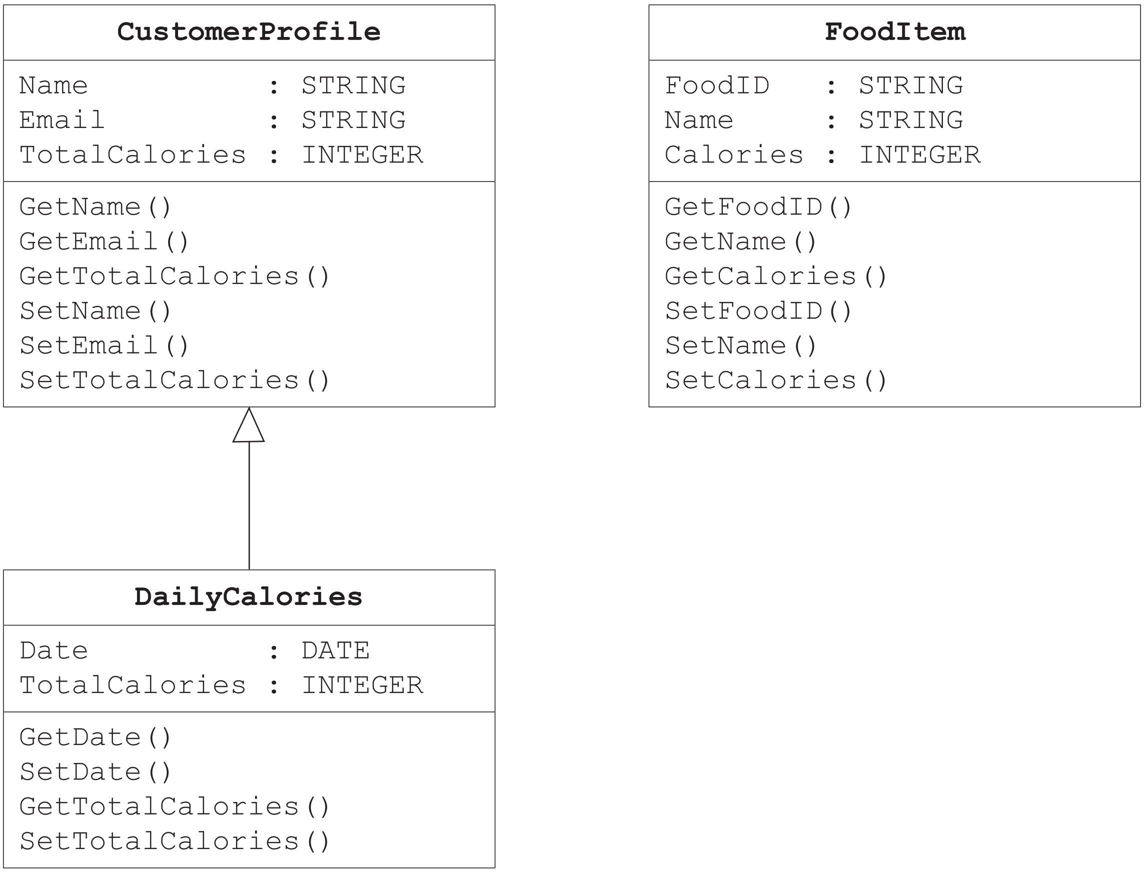
....

....

....

.... [4]

(d) The following is a class diagram for the application.



(i) The attributes of the class, `CustomerProfile`, are declared as private.

Explain why it is good practice to declare class attributes as private.

....

....

....

.... [2]

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

....

....

....

.... [2]

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

....

....

....

.... [2]

(e) Object-oriented programming is an example of a programming paradigm. Another example is imperative programming.

Explain what is meant by the **imperative programming paradigm**.

....

....

....

.... [2]

(f) Testing is regularly performed during the development of software.

(i) Independent modules are combined to create the final program. Testing is performed to make sure they interact correctly.

Identify this type of testing.

.... [1]

(ii) Testing is performed to prove to the customer that the system works correctly and meets the requirements specified in the design.

Identify this type of testing.

.... [1]

(iii) Test plans are used when testing data. One item that would be included in a test plan is example test data.

Identify **two other** items that would appear in a test plan.

1 ....

2 ....
[2]

5 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.

Consider the following pseudocode algorithm:

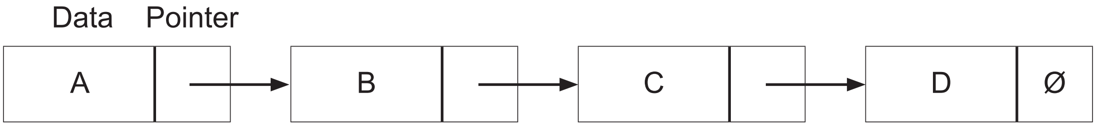
```
Length ← 0
INPUT Character
WHILE Character <> "."
    Message ← Message & Character
    Length ← Length + 1
    INPUT Character
ENDWHILE
```

Complete the table by writing assembly language code for the algorithm, using the given instruction set.

Label	Instruction		Comment
	Op code	Operand	
			// initialise IX to zero
			// initialise LENGTH
LOOP:			// input character
			// is character a FULLSTOP (.) ?
			// jump to ENDP if TRUE
			// store character in MESSAGE + contents of IX
			// increment IX
			// increment LENGTH
			// jump to LOOP
ENDP:	END		// end program
LENGTH:			
FULLSTOP:	B01100000		// ASCII code for a full stop (.)
MESSAGE:			

[8]

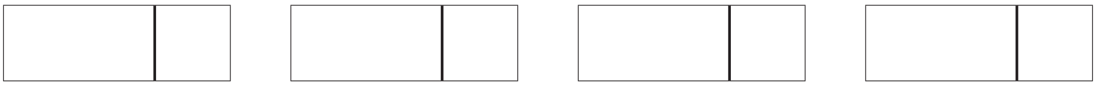
6 The following diagram represents a linked list.



The symbol Ø represents a null pointer.

(a) The node with the data value C is removed from the list.

Show the new state of the linked list.



[2]

(b) State what happens to the node with the data value C when it is removed from the list.

....

.... [1]

(c) State what is meant by a **null pointer**.

.... [1]

(d) The linked list is implemented as a 1D array, `LinkedList`. The array is declared as a record data type, with two fields, `Data` and `Pointer`.

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 returns `-1` if the value is not found, or it returns the pointer to the value if it is found.

The global variable `StartPointer` points to the first element in the list.

Write **pseudocode** for the function `FindValue()`.

[8]

(e) A linked list and a record are both examples of abstract data types.

Identify **and** describe **one other** abstract data type.

Abstract data type ....

Description ....

....

....

....

....

....

.... [4]

BLANK PAGE

BLANK PAGE