Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

1378702323

COMPUTER SCIENCE **9608/42**

Paper 4 Further Problem-solving and Programming Skills **October/November 2018**

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 **15** printed pages and **1** blank page.

CAMBRIDGE International Examinations

1 A bank provides bank accounts to customers.

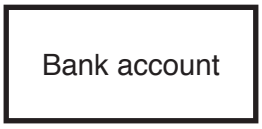
The following pseudocode represents the operation of the bank account.

```
CALL OpenAccount()
CALL AccountLifeTime()
CALL CloseAccount()

PROCEDURE AccountLifeTime()
   REPEAT
      CALL Transactions()
   UNTIL AccountClosed() = TRUE
ENDPROCEDURE

PROCEDURE CloseAccount()
   IF ReopenAccount() = TRUE
      THEN
         CALL FlagToReopen()
      ELSE
         CALL DeletePermanently()
   ENDIF
ENDPROCEDURE
```

(a) Complete the JSP structure diagram for this bank account from the pseudocode given.



[6]

(b) A transaction can be a credit (deposit) or a debit (withdrawal).

There are two types of transaction that are credits (deposits). These are:

- SWIFT payment
- BACS payment

There are three types of transaction that are debits (withdrawals). These are:

- Debit card payment
- Cheque payment
- Online payment

Complete the JSP structure diagram to represent these additional requirements.

[4]

2 A declarative language is used to represent facts and rules about dogs that perform tasks to help people.

```
01 type(pointer, gundog).
02 type(flushing, gundog).
03 type(retriever, gundog).
04
05 is_a(labrador, retriever).
06 is_a(newfoundland, retriever).
07 is_a(cocker_spaniel, flushing).
08 is_a(springer_spaniel, flushing).
09 is_a(king_charles, flushing).
10 is_a(english_setter, pointer).
11 is_a(irish_setter, pointer).
12
13 fav_bird(pointer, grouse).
14 fav_bird(flushing, pheasant).
15 fav_bird(retriever, waterfowl).
16
17 type(X, Y) IF is_a(Z, X) AND type(Z, Y).
```

These clauses have the following meaning:

Clause	Explanation
01	A pointer is a type of gundog.
05	A labrador is an example of a retriever.
13	The favourite bird of a pointer is a grouse.
17	X is a type of Y if Z is an example of X and Z is a type of Y.

(a) More facts are to be included.

A **standard poodle** is an example of a waterdog. A waterdog is a type of gundog.

Write the additional clauses to record these facts.

18 .....

19 .....

[2]

(b) Using the variable `P`, the goal

```
is_a(P, retriever)
```

returns

```
P = labrador, newfoundland
```

Write the result returned by the goal

```
is_a(H, pointer)
```

`H =` ..... [2]

(c) Write a query, using the variable `W`, to find out what an **irish setter** is an example of.

..... [2]

(d) `Y` is the favourite bird of dog `X`.

Complete the following rule:

`fav_bird(X, Y) IF` .....

..... [3]

(e) State the value returned by the goal

```
NOT(is_a(labrador, retriever))
```

..... [1]

3 A bubble sort algorithm is used to sort an integer array, `List`. This algorithm can process arrays of different lengths.

(a) Write **pseudocode** to complete the bubble sort algorithm shown.

```
01 FOR Outer ← ..... TO 0 STEP - 1
02   FOR Inner ← 0 TO (.....)
03     IF .....  >  .....
04       THEN
05         Temp ← .....
06         List[Inner] ← .....
07         List[Inner + 1] ← .....
08     ENDIF
09   ENDFOR
10 ENDFOR
```

[7]

(b) (i) State the order of the sorted array.

..... [1]

(ii) State which line of the algorithm you would change to sort the array into the opposite order.

State the change you would make.

Line .....

Change .....

.....

[1]

(c) Use **pseudocode** to write an alternative version of this bubble sort algorithm that will exit the algorithm when the list is fully sorted.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

4 A circus is made up of performers. There are three types of performer: clown, acrobat and aerial.

The following data are stored for each performer.

- First name
- Last name
- Secondary role (that can be edited)
- Stage name (that can be edited)
- Type of performer (PerfType)

The following statements apply to performers.

- An acrobat may or may not use fire in his or her act.
- An aerial performer can be one of two types: either catcher or flyer.
- Each clown has an item, such as a water-spraying flower or a unicycle.
- Each clown also has a musical instrument, such as a guitar or an oboe.

Each of the three types of performer has a method that will display all of the information about that performer in a specific format. For example:

Sally Superstar (real name Sally Smith) is an acrobat. Fire is part of Sally Superstar's act. When not performing, Sally Superstar is a set changer.

(a) Complete the following class diagram to show the **attributes**, **methods** and **inheritance** for the program.

You do not need to write the get and set methods.

Performer
FirstName : STRING LastName : STRING SecondaryRole : STRING StageName : STRING PerfType : STRING
Constructor() EditSecondaryRole() EditStageName()

Acrobat
UseFire : BOOLEAN
Constructor() PerformerInfo()

Clown
....
Constructor() .....

Aerial
.....
Constructor() .....

[4]

(b) Write **program code** for the `Performer` class.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(c) The program will display the acrobat information as follows:

> *Sally Superstar (real name Sally Smith) is an acrobat. Fire is part of Sally Superstar's act. When not performing, Sally Superstar is a set changer.*

Write **program code** for the `Acrobat` class.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(d) Information about a performer is as follows:

Amazing Alex (real name Alex Tan) is an acrobat. Fire is part of Alex's act. When not performing, Amazing Alex is a popcorn seller.

(i) Write **program code** to create an instance of an object with the identifier `Acrobat_1`.

All attributes of the instance should be fully initialised.

.....

.....

.....

..... [3]

(ii) Explain **inheritance** with reference to the circus example.

.....

.....

.....

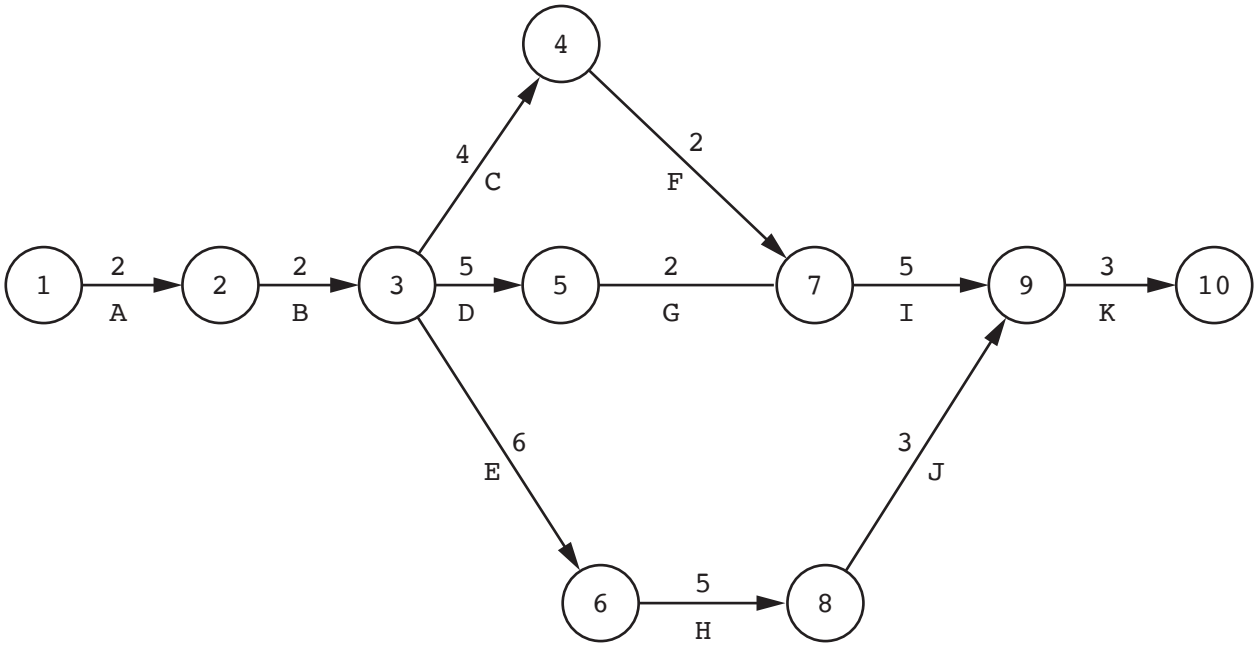
.....

.....

.....

..... [2]

5 A Program Evaluation Review Technique (PERT) chart has been constructed for a project that is at the planning stage.



(a) Complete the following GANTT chart using the information in the PERT chart.

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**	**21**	**22**	**23**	**24**	**25**

[5]

(b) There are three teams working on the project. Each team is able to work on any of the activities.

Explain, with reference to the PERT chart, how work can be allocated to the three teams.

.....

.....

.....

.....

.....

..... [2]

(c) The PERT chart is used to calculate the critical path for the project.

(i) List the activities that form the critical path using the given PERT chart on page 12.

..... [1]

(ii) Explain the importance of the critical path for project delivery.

.....

.....

.....

.....

..... [2]

6 A linked list abstract data type (ADT) is created. This is implemented as an array of records. The records are of type `ListElement`.

An example of a record of `ListElement` is shown in the following table.

Data Item	Value
Country	"Scotland"
Pointer	1

(a) (i) Use **pseudocode** to write a definition for the record type, `ListElement`.

.....

.....

.....

.....

..... [3]

(ii) Use **pseudocode** to write an array declaration to reserve space for only 15 nodes of type `ListElement` in an array, `CountryList`. The lower bound element is `1`.

..... [2]

(b) The program stores the position of the last node in the linked list in `LastNode`. The last node always has a `Pointer` value of `-1`. The position of the node at the head of the list is stored in `ListHead`.

After some processing, the array and variables are in the following state.

ListHead
1
LastNode
3

CountryList

	Country	Pointer
1	"Wales"	2
2	"Scotland"	4
3		-1
4	"England"	5
5	"Brazil"	6
6	"Canada"	7
7	"Mexico"	8
8	"Peru"	9
9	"China"	10
10		11
11		12
12		13
13		14
14		15
15		3

A **recursive** algorithm searches the list for a value, deletes that value, and updates the required pointers. When a node value is deleted, it is set to empty "" and the node is added to the end of the list.

A node value is deleted using the pseudocode statement

```
CALL DeleteNode("England", 1, 0)
```

Complete the following **pseudocode** to implement the `DeleteNode` procedure.

```
PROCEDURE DeleteNode(NodeValue: STRING, ThisPointer : INTEGER,
                                                    PreviousPointer : INTEGER)

 IF CountryList[ThisPointer].Value = NodeValue

   THEN

     CountryList[ThisPointer].Value ← ""

     IF ListHead = ……

       THEN

         ListHead ← ……

       ELSE

         CountryList[PreviousPointer].Pointer ← CountryList[ThisPointer].Pointer

     ENDIF

     CountryList[LastNode].Pointer ← ……

     LastNode ← ThisPointer

     ……

   ELSE

     IF CountryList[ThisPointer].Pointer <> -1

       THEN

          CALL DeleteNode(NodeValue, ……,
                                                              ThisPointer)

       ELSE

          OUTPUT "DOES NOT EXIST"

     ENDIF

 ENDIF

ENDPROCEDURE
```

[5]

BLANK PAGE