Cambridge International AS & A Level

**Cambridge International Examinations**
Cambridge International Advanced Subsidiary and Advanced Level

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| CANDIDATE NAME | | | | | | | | | |
| CENTRE NUMBER | | | | | | CANDIDATE NUMBER | | | |

*1747453880*

**COMPUTER SCIENCE** **9608/42**

Paper 4 Further Problem-solving and Programming Skills **May/June 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 **22** printed pages and **2** blank pages.

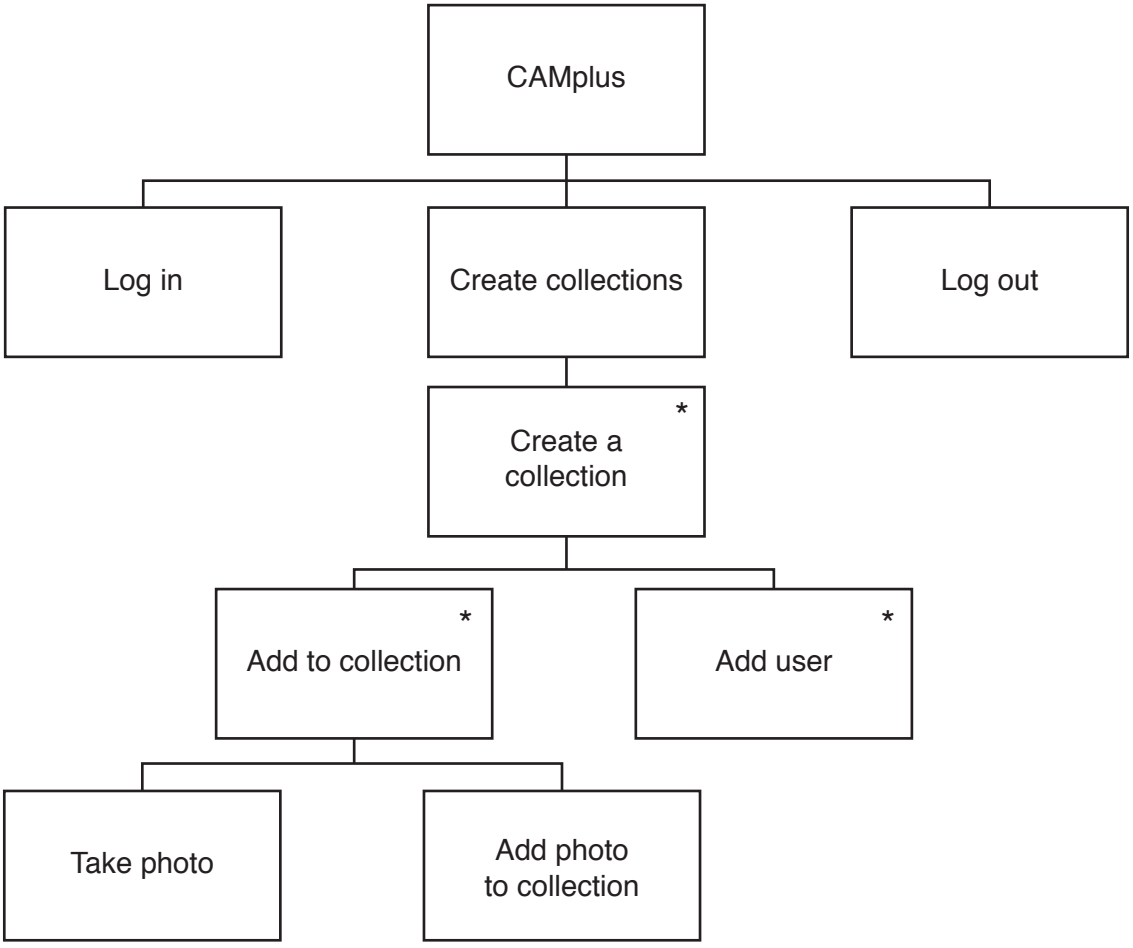



CAMBRIDGE International Examinations

**1** Paul is using an application (app) called CAMplus. The app allows users to:

- log in
- create a new collection of photographs
- use the camera to take new photographs
- automatically add new photographs to the new collection
- share the new collection with other users
- start another collection or log out of the app.

The following JSP structure diagram represents the operation of CAMplus.



**(a)** An algorithm has been written in pseudocode to represent the **Create collections** operation from the JSP structure diagram. The algorithm is incomplete.

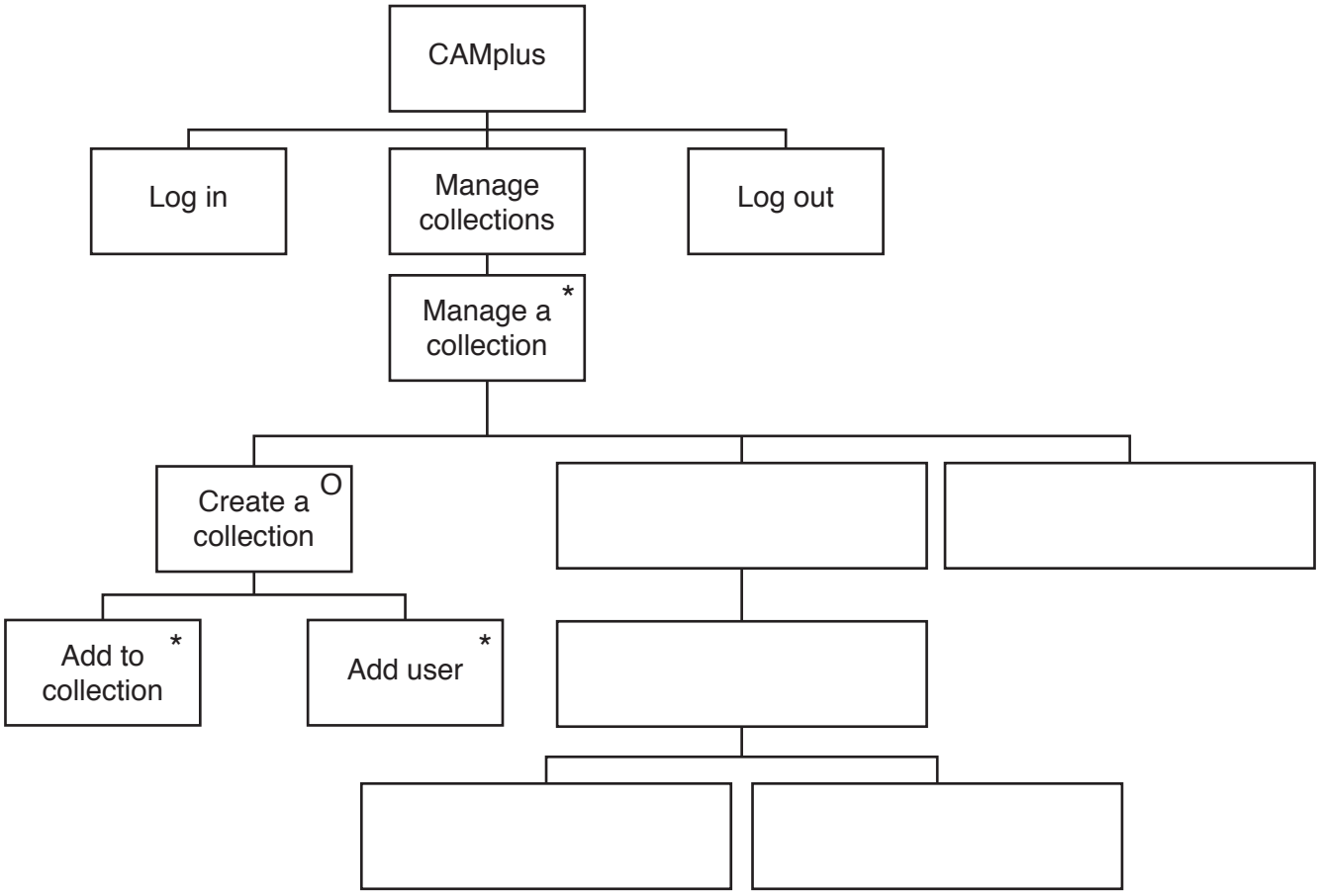
Write **pseudocode** to complete this algorithm.

```
REPEAT

   REPEAT

      CALL TakePhoto

      ..........

      OUTPUT "Do you want to take another photo?"

      INPUT AddPhoto

   UNTIL AddPhoto = "No"

   REPEAT

      ..........

      OUTPUT "Do you want to add another user?"

      INPUT NewUser

   UNTIL ..... = "No"

   OUTPUT "Do you want to create another collection?"

   ..........

UNTIL NewCollection = "No"
```

[4]

**(b)** The app is updated. Paul can now add and delete photos from chosen collections. Paul can also delete collections.

Complete the JSP structure diagram to show the changes.




[5]

**Question 2 begins on the next page.**

**2** A declarative language is used to represent the following facts and rules about iguanas and lizards.

```
01   has(reptile, cold_blood).
02   has(reptile, air_breathing).
03   has(reptile, scales).
04
05   is_a(squamata, reptile).
06   is_a(iguana, squamata).
07   is_a(lizard, squamata).
08   is_a(green_iguana, iguana).
09   is_a(cayman, iguana).
10   is_a(smooth_iguana, iguana).
11
12   maxsize(green_iguana, 152).
13   maxsize(cayman, 90).
14   maxsize(smooth_iguana, 70).
```

These clauses have the following meaning:

| Clause | Explanation |
|---|---|
| 01 | A reptile has cold blood. |
| 09 | A cayman is a type of iguana. |
| 12 | The maximum size of a green iguana is 152 cm. |

**(a)** More facts are to be included.

A gecko is a type of lizard. It has a maximum size of 182 cm.

Write the additional clauses to record these facts.

15 .....

16 .....

[2]

**(b)** Using the variable `R`, the goal

```
is_a(R, squamata).
```

returns

```
R = iguana, lizard
```

Write the result returned by the goal

```
is_a(T, iguana).
```

`T =` .....[2]

**(c)** Write the goal, using the variable `X`, to find what a squamata is.

.....[2]

**(d)** All iguanas and lizards are squamata. All squamata are reptiles.

Write a recursive rule to make all lizards and iguanas inherit the properties of reptiles.

```
has(X, Y)

IF
```

.....

.....[3]

**(e)** State what the following goal returns.

```
NOT(maxsize(cayman, 70)).
```

.....[1]

**3** The arrays `PollData[1:10]` and `CardData[1:10]` store data.

| PollData | 12 | 85 | 52 | 57 | 25 | 11 | 33 | 59 | 56 | 91 |
|---|---|---|---|---|---|---|---|---|---|---|

| CardData | 11 | 12 | 25 | 33 | 52 | 56 | 57 | 59 | 91 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|

An **insertion sort** sorts these data.

**(a)** State why it will take less time to complete an insertion sort on `CardData` than on `PollData`.

.....

.....[1]

**(b)** The following pseudocode algorithm performs an insertion sort on the `CardData` array.

Complete the following **pseudocode** algorithm.

```
01  ArraySize ← 10
02  FOR Pointer ← 2 TO .....
03    ValueToInsert ← CardData[Pointer]
04    HolePosition ← .....
05    WHILE (HolePosition > 1 AND (..... > ..... ))
06      CardData[HolePosition] ← CardData[.....]
07      HolePosition ← .....
08    ENDWHILE
09    CardData[HolePosition] ← .....
10  ENDFOR
```

[7]

**(c) (i)** A binary search algorithm is used to find a specific value in an array.

Explain why an array needs to be sorted before a binary search algorithm can be used.

.....

.....

.....

.....

.....

.....

.....

.....[2]

**(ii)** The current contents of `CardData` are shown.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 12 | 25 | 33 | 52 | 56 | 57 | 59 | 85 | 91 |

Explain how a binary search will find the value `25` in `CardData`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

**(d)** Complete this procedure to carry out a binary search on the array shown in **part (c)(ii)**.

```
PROCEDURE BinarySearch(CardData, SearchValue)
   DECLARE Midpoint : INTEGER
   First ← 1
   Last ← ARRAYLENGTH(.....)
   Found ← FALSE
   WHILE (First <= Last) AND NOT(Found)
      Midpoint ← .....
      IF CardData[Midpoint] = SearchValue
         THEN
            Found ← TRUE
         ELSE
            IF SearchValue < CardData[Midpoint]
               THEN
                  Last ← .....
               ELSE
                  First ← .....
            ENDIF
         ENDIF
   ENDWHILE
ENDPROCEDURE
```

[4]

**Question 4 begins on the next page.**

**4** X-Games is an international extreme sports competition.

A program will store and process data about the teams in the competition.

- Each team is made up of members.
- Members can be added and removed from each team.
- Each member has a first name, last name, date of birth and gender.
- Each member can be an official or a competitor.
- Each official has a job title and may be first-aid trained.
- Each competitor takes part in one sport.

The program is written using object-oriented programming.

The program can output the full name and date of birth of any member. For example, “Nadia Abad 16/05/1995”

An introduction about a team member can be output using their name. For example, “Hello, I’m Nadia Abad”.

The program outputs a different version of the introduction for a competitor. This version includes the competitor’s sport. For example, “Hello, I’m Sally Jones and my sport is Skateboard Park.”

**(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.

| **Member** |
| --- |
| FirstName : STRING<br>LastName : STRING<br>DateOfBirth : DATE<br>Gender : STRING |
| Constructor()<br>Introduction()<br>DisplayFullnameAndDateOfBirth() |

| **Team** |
| --- |
| TeamName : STRING<br>TeamList : ARRAY OF Member |
| Constructor()<br>.....<br>..... |

| **Competitor** |
| --- |
| Sport : STRING |
| Constructor()<br>Introduction() |

| **Official** |
| --- |
| .....<br>..... |
| Constructor()<br>DisplayJobTitle() |

[3]

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

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

**(c)** Write **program code** for the `Competitor` class.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

**(d)** Omar Ellaboudy is an official at X-Games. He is first-aid trained and his job title is Judge. He is male and was born on 17/03/1993.

Write **program code** to create an instance of an object with the identifier `BMXJudge`. All attributes of the instance must be fully initialised.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....[3]

**Question 5 begins on the next page.**

**5** A company is developing an application program. The project manager has been asked to create a work breakdown schedule for the project as follows:

| Activity | | Days to complete | Predecessor activity |
|---|---|---|---|
| **A** | Gather User Requirements | 6 | |
| **B** | Design work | 4 | A |
| **C** | Develop server code | 4 | B |
| **D** | Develop application code | 5 | B |
| **E** | User Interface Development | 6 | B |
| **F** | Test server code | 2 | C |
| **G** | Test application | 2 | D, E |
| **H** | Test application/server integration | 6 | F, G |
| **I** | Roll out mobile application | 6 | H |

**(a)** A GANTT chart is created from the work breakdown schedule. Activities **A** and **B** have already been added to the chart.

Complete the GANTT chart.

| Activity | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **B** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **C** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **D** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **E** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **F** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **G** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **H** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **I** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Day 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** | **26** | **27** | **28** | **29** | **30** |

[5]

**(b)** State which activities can run in parallel on the following days.

**(i)** Day 14

.....[1]

**(ii)** Day 16

.....[1]

**(c)** Explain how the project manager will use the GANTT chart to make sure the project is completed on time.

.....

.....

.....

.....[2]

**6** An Abstract Data Type (ADT) is used to create an unordered binary tree. The binary tree is created as an array of nodes. Each node consists of a data value and two pointers.

A record type, `Node`, is declared using pseudocode.

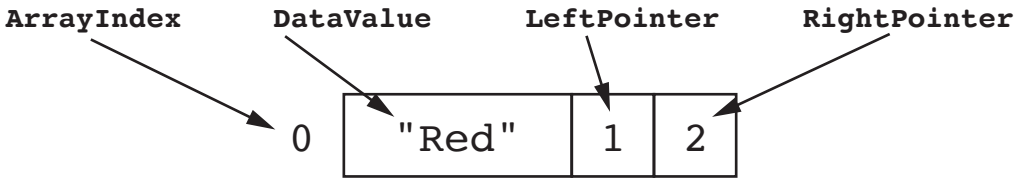
```
TYPE Node
   DECLARE DataValue : STRING
   DECLARE LeftPointer : INTEGER
   DECLARE RightPointer : INTEGER
ENDTYPE
```

The following statement declares an array `BinaryTree`.

```
   DECLARE BinaryTree : ARRAY[0:14] OF Node
```

A variable, `NextNode`, points to the next free node.

The following diagram shows a possible node.



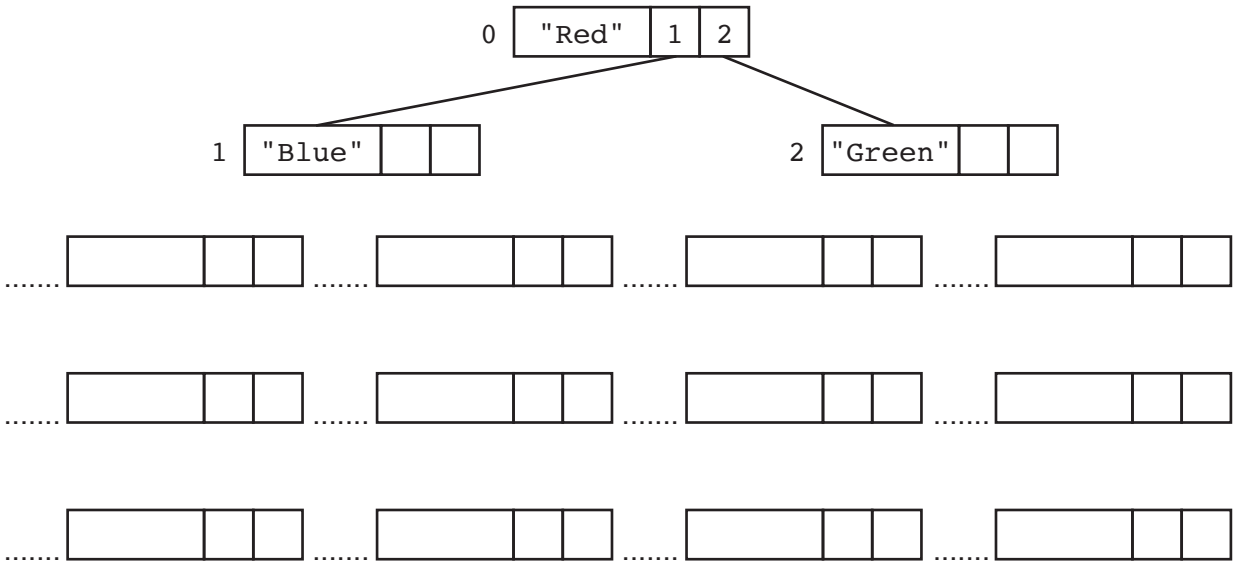

The commands in the following table create and add nodes to the binary tree.

| Command | Comment |
|---|---|
| `CreateTree(NodeData)` | Sets `NextNode` to `0`.<br>Writes `NodeData` into `DataValue` at the position `NextNode`<br>Updates `NextNode` using `NextNode = NextNode + 1` |
| `AttachLeft(NodeData, ParentNode)` | Writes `NodeData` into `DataValue` of `NextNode`<br>Sets the `LeftPointer` of node `ParentNode` to `NextNode`<br>Updates `NextNode` using `NextNode = NextNode + 1` |
| `AttachRight(NodeData, ParentNode)` | Writes `NodeData` into `DataValue` of `NextNode`<br>Sets the `RightPointer` of node `ParentNode` to `NextNode`<br>Updates `NextNode` using `NextNode = NextNode + 1` |

**(a)** The following commands are executed.

```
CreateTree("Red")
AttachLeft("Blue", 0)
AttachRight("Green", 0)
```

The following diagram shows the current state of the binary tree.




Write on the diagram to show the state of the binary tree after the following commands have been executed.

```
AttachRight("Black", 2)
AttachLeft("Brown", 2)
AttachLeft("Peach", 3)
AttachLeft("Yellow", 1)
AttachRight("Purple", 1)
AttachLeft("White", 6)
AttachLeft("Pink", 7)
AttachLeft("Grey", 9)
AttachRight("Orange", 9)
```

[5]

**(b)** A new command has been added to initialise the pointers of the binary tree to `-1` to indicate they are not in use.

A leaf is a node of the binary tree which has no children. In the case of this binary tree, a node with a `LeftPointer` of `-1` and a `RightPointer` of `-1` is a leaf.

Write a **recursive** function, in **program code**, to traverse the binary tree and output the value of `DataValue` for each leaf node.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

**BLANK PAGE**

**BLANK PAGE**