Cambridge International A Level

Cambridge International Examinations
Cambridge International Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTER SCIENCE **9608/43**

Paper 4 Further Problem-solving and Programming Skills **May/June 2015**

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

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

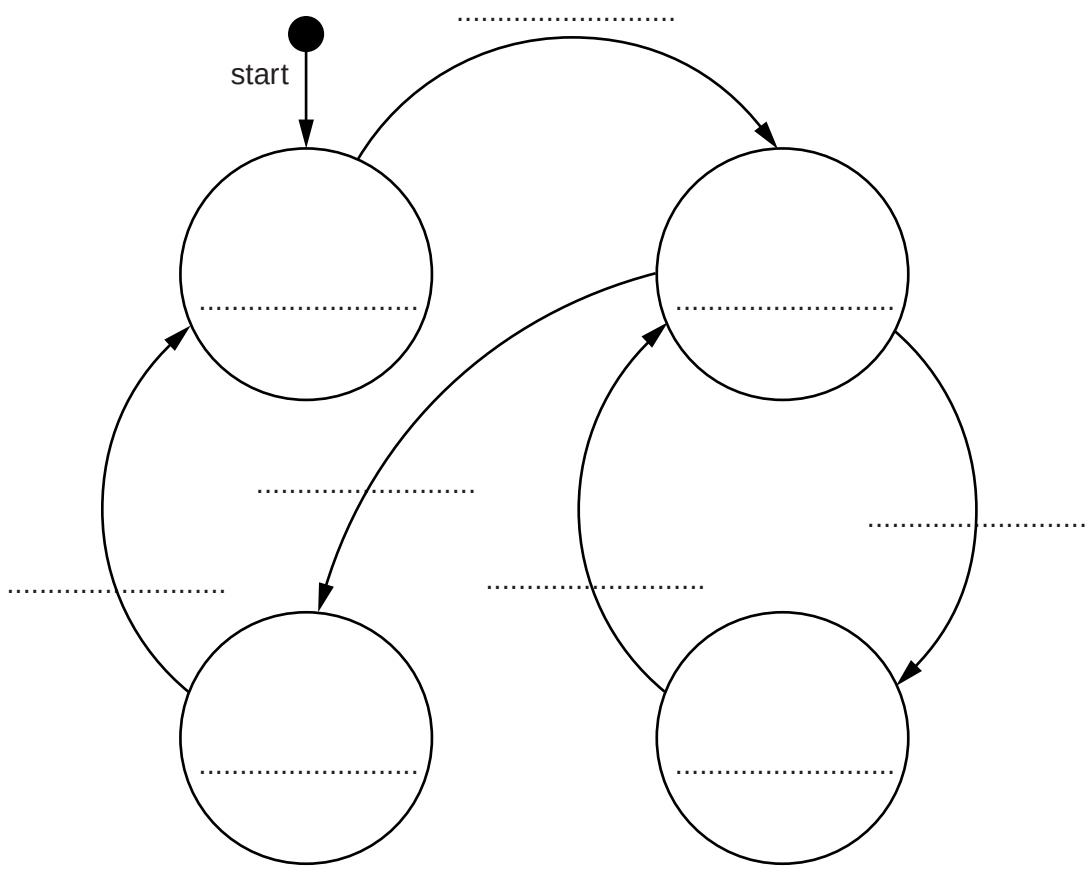
1 A petrol filling station has a single self-service petrol pump.

A customer can use the petrol pump when it is ready to dispense petrol.
The pump is in use when the customer takes the nozzle from a holster on the pump.
The pump dispenses petrol while the customer presses the trigger on the nozzle.
When the customer replaces the nozzle into the holster, the pump is out of use.
The cashier must press a reset button to make the pump ready for the next customer to use.

The petrol pump's four possible states and the transition from one state to another are as shown in the table below.

Current state	**Event**	**Next state**
Pump ready	Take nozzle	Pump in use
Pump in use	Press trigger	Pump dispensing
Pump dispensing	Stop pressing trigger	Pump in use
Pump in use	Replace nozzle	Pump out of use
Pump out of use	Reset pump display	Pump ready

Complete the state transition diagram for the petrol pump:



[9]

2 A declarative programming language is used to represent the knowledge base shown below:

```
01  dairy_product(cheese).
02  meat(beef).
03  meat(chicken).
04  meat(lamb).
05  made_with(burger, beef).
06  made_with(kofta, lamb).
07  made_with(quiche, cheese).
08  made_with(quiche, egg).
09  made_with(quiche, flour).
```

These clauses have the following meaning:

Clause	Explanation
01	Cheese is a dairy product
02	Beef is a meat
05	A burger is made with beef

(a) More facts are to be included.

Laasi is made with the dairy products milk and yogurt.

Write additional clauses to record this.

10 .....

.....

11 .....

.....

12 .....

.....

13 .....

..... [4]

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

```
meat(TypeOfMeat)
```

returns

```
TypeOfMeat =  beef, chicken, lamb
```

Write the result returned by the goal:

```
made_with(quiche, Ingredient)
```

`Ingredient` = ...

..... [2]

(c) Complete the rule to list the dishes made with meat.

```
contains_meat(Dish)
```

`IF` .....

.....

.....

.....

..... [4]

3 An insurance company calculates the cost of car insurance from a basic price.

The driver may:
- get a discount on the basic price of the insurance
- have to pay an extra charge

The decision is arrived at as follows:
- for a driver aged 25 or over:
  - 5% discount if no previous accident
  - no discount if a previous accident
- for a driver under the age of 25:
  - 5% discount if no previous accident and licence held for 3 or more years
  - no discount if a previous accident but licence held for 3 or more years
  - no discount if no previous accident but licence held for less than 3 years
  - 10% extra charge if a previous accident and licence held for less than 3 years

(a) Complete the decision table.

Conditions	Age under 25	Y	Y	Y	Y	N	N	N	N
	Previous accident	Y	Y	N	N	Y	Y	N	N
	Licence held for 3 or more years	Y	N	Y	N	Y	N	Y	N
Actions	10% extra charge								
	No discount								
	5% discount								

[6]

(b) Simplify your solution by removing redundancies.

Conditions	Age under 25								
	Previous accident								
	Licence held for 3 or more years								
Actions	10% extra charge								
	No discount								
	5% discount								

[3]

(c) The simplified table produced in **part (b)** is used as a design for program code.

The following identifier table shows the parameters to be passed to the function `CostPercentageChange`. This function returns the percentage change from the basic price as an integer. A discount should be shown as a negative integer. An extra charge should be shown as a positive integer.

Identifier	Data type	Comment
`DriverAge`	`INTEGER`	Age of driver in years
`HadAccident`	`BOOLEAN`	Whether driver has had a previous accident
`YearsLicenceHeld`	`INTEGER`	Number of years the driver has held licence

Write **program code** for this function.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

4 A sports club stores data about its members. A program is to be written using an object-oriented programming language.

A `Member` class is designed. Two subclasses have been identified:
- `FullMember`
- `JuniorMember`

(a) Draw an inheritance diagram for these classes.

[3]

(b) The design for the `Member` class consists of

- properties
  - `MemberName`
  - `MemberID`
  - `SubscriptionPaid`

- methods
  - `SetMemberName`
  - `SetMemberID`
  - `SetSubscriptionPaid`

Write **program code** for the class definition of the superclass `Member`.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

(c) Additionally a `DateOfBirth` property is required for the `JuniorMember` class.

(i) Write **program code** for the class definition for the subclass `JuniorMember`.

.....

.....

.....

.....

.....

..... [3]

(ii) Write **program code** to create a new instance of `JuniorMember`. Use identifier `NewMember` with the following data:
name Ahmed with member ID 12347, born on 12/11/2001, who has paid his subscription.

.....

.....

.....

.....

.....

..... [3]

5 A stack Abstract Data Type (ADT) has these associated operations:

- create stack
- add item to stack (push)
- remove item from stack (pop)

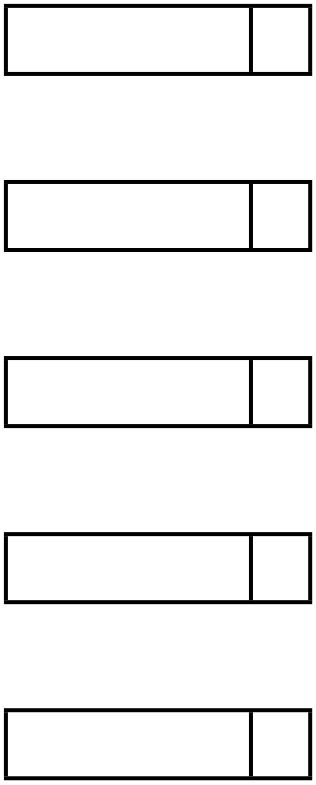
The stack ADT is to be implemented as a linked list of nodes.

Each node consists of data and a pointer to the next node.

(a) There is one pointer: the top of stack pointer, which points to the last item added to the stack. Draw a diagram to show the final state of the stack after the following operations are carried out.

```
CreateStack
Push("Ali")
Push("Jack")
Pop
Push("Ben")
Push("Ahmed")
Pop
Push("Jatinder")
```

Add appropriate labels to the diagram to show the final state of the stack. Use the space on the left as a workspace. Show your final answer in the node shapes on the right:



[3]

(b) Using pseudocode, a record type, `Node`, is declared as follows:

```
TYPE Node
  DECLARE Name : STRING
  DECLARE Pointer : INTEGER
ENDTYPE
```

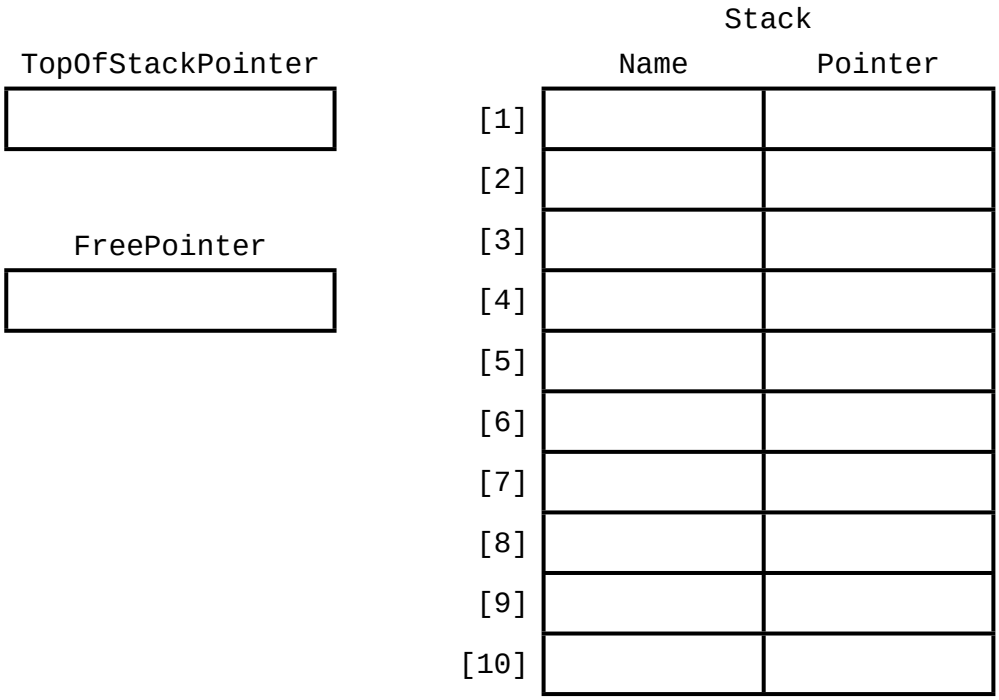
The statement

```
DECLARE Stack : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array `Stack`.

(i) The `CreateStack` operation links all nodes and initialises the `TopOfStackPointer` and `FreePointer`.

Complete the diagram to show the value of all pointers after `CreateStack` has been executed.



[4]

(ii) The algorithm for adding a name to the stack is written, using pseudocode, as a procedure with the header

```
PROCEDURE Push (NewName)
```

Where `NewName` is the new name to be added to the stack. The procedure uses the variables as shown in the identifier table.

Identifier	Data type	Description
`Stack`	`Array[1:10] OF Node`	
`NewName`	`STRING`	Name to be added
`FreePointer`	`INTEGER`	Pointer to next free node in array
`TopOfStackPointer`	`INTEGER`	Pointer to first node in stack
`TempPointer`	`INTEGER`	Temporary store for copy of `FreePointer`

```
PROCEDURE Push(BYVALUE NewName : STRING)
  // Report error if no free nodes remaining
  IF FreePointer = 0
     THEN
        Report Error
     ELSE
         // new name placed in node at head of free list
         Stack[FreePointer].Name ← NewName
         // take a temporary copy and
         // then adjust free pointer
         TempPointer ← FreePointer
         FreePointer ← Stack[FreePointer].Pointer
         // link current node to previous top of stack
         Stack[TempPointer].Pointer ← TopOfStackPointer
         // adjust TopOfStackPointer to current node
         TopOfStackPointer ← TempPointer
  ENDIF
ENDPROCEDURE
```

Complete the **pseudocode** for the procedure Pop. Use the variables listed in the identifier table.

```
PROCEDURE Pop()
  // Report error if Stack is empty
.....
.....
.....
.....
   OUTPUT Stack [.....].Name
   // take a copy of the current top of stack pointer
.....
   // update the top of stack pointer
.....
   // link released node to free list
.....
.....
.....
ENDPROCEDURE
```

[5]

6 A recursively defined procedure X is defined below:

```
PROCEDURE X(BYVALUE n : INTEGER)
   IF (n = 0) OR (n = 1)
      THEN
         OUTPUT n
      ELSE
         CALL X(n DIV 2)
         OUTPUT (n MOD 2)
   ENDIF
ENDPROCEDURE
```

(a) Explain what is meant by recursively defined.

.....

..... [1]

(b) Explain how a stack is used during the execution of a recursive procedure.

.....

.....

.....

..... [2]

(c) Dry run the procedure X by completing the trace table for the procedure call:

```
CALL X(40)
```

Call number	n	(n = 0) OR (n = 1)	n DIV 2	n MOD 2
1	40	FALSE	20	
2				
3				
4				
5				
6				

OUTPUT ..... [6]

(d) State the process that is carried out by procedure X.

.....

..... [1]

(e) Write **program code** for procedure X.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

..... [5]

BLANK PAGE