QUESTION 1.

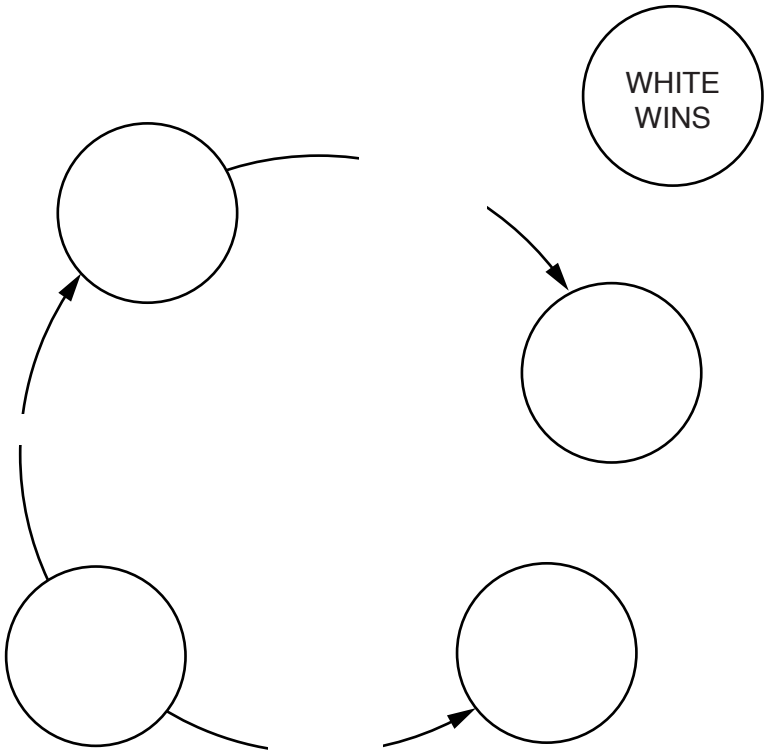
6 In a board game, one player has white pieces and the other player has black piec
alternate turns to move one of their pieces. White always makes the first move.

The game ends if:

- a player is unable to make a move when it is their turn. In this case, there is no winner. This
called 'stalemate'.
- a player wins the game as a result of their last move and is called a 'winner'.

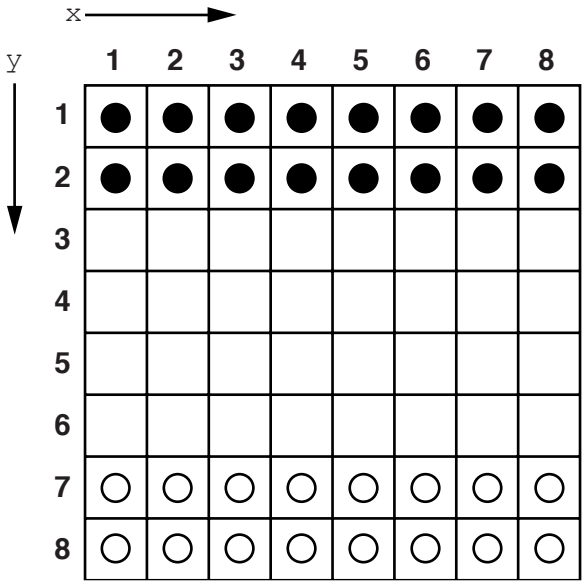
(a) A state-transition diagram is drawn to clarify how the game is played.

Complete the following state-transition diagram.



[4]

(b) The layout of the board at the start of the game is shown below:



The programmer decides to use a 2D array to represent the board. The ind
be used is as shown.

Each square on the board is either occupied by one piece only, or is empty.

The data stored in the array indicate whether or not that square is occupied, and if so, wit
black piece or a white piece.

(i) Write **program code** to initialise the contents of the array to represent the board at the start of the game. Use characters as follows for each square:

- ●
- `'W'` represents a white piece ○
- `'E'` represents an empty square

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

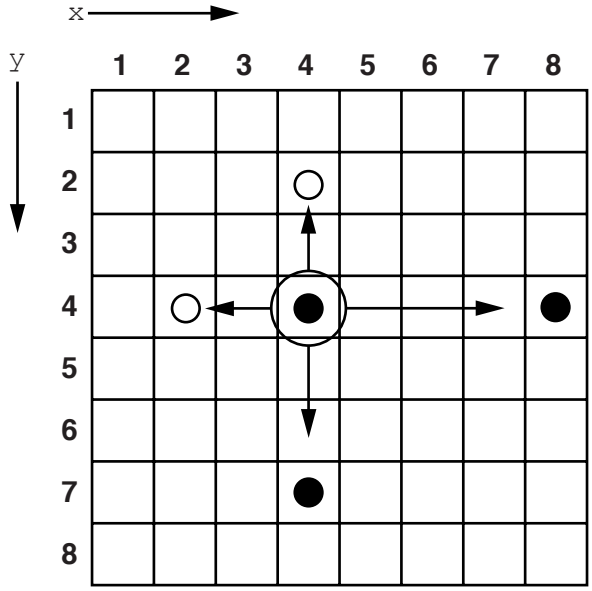
(ii) When a piece is to be moved, a procedure will calculate and outp
destination squares for the moving piece.

A piece can move one or more squares, in the `x` or `y` direction, from its current p

This will be a move:

- either to an empty square, with no occupied squares on the way
- or to a square containing a piece belonging to another player, with no occupied squares on the way. The other player's piece is then removed.

For example, for the circled black piece there are nine possible destination squares. Each of the two destination squares contains a white piece which would be removed.



The program requires a procedure `ValidMoves`.

It needs three parameters:

- `PieceColour` – colour of the moving piece
- `xCurrent` – current `x` position
- `yCurrent` – current `y` position

The procedure will calculate all possible destination squares **in the `x` direction only**.

Example output for the circled black piece is:

```
Possible moves are:
Moving LEFT
3 4
2 4 REMOVE piece
Moving RIGHT
5 4
6 4
7 4
```

Write **program code** for procedure `ValidMoves` with the following procedure header:

```
PROCEDURE ValidMoves(PieceColour : CHAR, xCurrent : INTEGER,
                                         yCurrent : INTEGER).
```

Visual Basic and Pascal: You should include the declaration statements
Python: You should show a comment statement for each variable used with

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

(c) The problem is well suited to an object-oriented design followed by programming.

(i) Describe how classes and objects could be used in this problem.

.....

.....

.....

.....

.....[2]

(ii) For a class you identified in **part(c)(i)**, state **two** properties and **two** methods.

Class .....

Properties

1 .....

2 .....

Methods

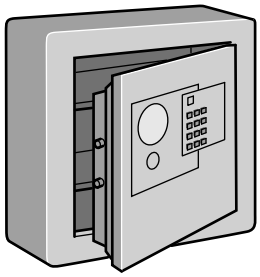
1 .....

2 .....

[2]

QUESTION 2.

1 A user can lock a safety deposit box by inputting a 4-digit code. The user can un
the same 4-digit code.



There is a keypad on the door of the safety deposit box. The following diagram shows the keys on the keypad.

1	2	3
4	5	6
7	8	9
R	0	Enter

Initially, the safety deposit box door is open and the user has not set a code.

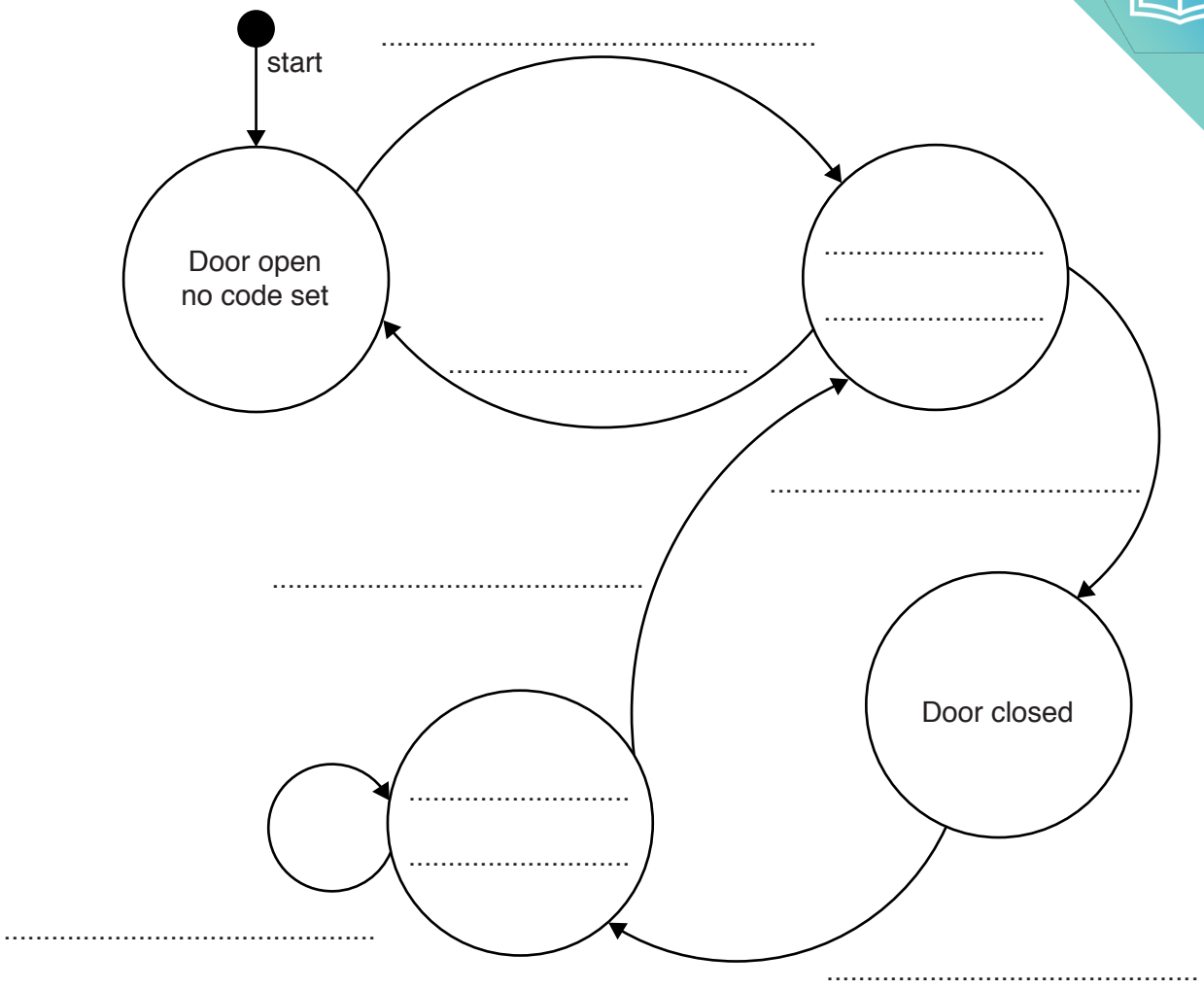
The operation of the safety deposit box is as follows:

A) To set a new code the door must be open. The user chooses a 4-digit code and sets it by pressing the numerical keys on the keypad, followed by the Enter key. Until the user clears this code, it remains the same. (See point E below)
B) The user can only close the door if the user has set a code.
C) To lock the door, the user closes the door, enters the set code and presses the Enter key.
D) To unlock the door, the user enters the set code. The door then opens automatically.
E) The user clears the code by opening the door and pressing the R key, followed by the Enter key. The user can then set a new code. (See point A above)

The following state transition table shows the transition from one state to another of the safety deposit box:

Current state	Event	Next state
Door open, no code set	4-digit code entered	Door open, code set
Door open, code set	R entered	Door open, no code set
Door open, code set	Close door	Door closed
Door closed	Set code entered	Door locked
Door locked	Set code entered	Door open, code set
Door locked	R entered	Door locked

(a) Complete the state-transition diagram.



[7]

(b) A company wants to simulate the use of a safety deposit box. It will do this wit
programming (OOP).

The following diagram shows the design for the class `SafetyDepositBox`. This incl
properties and methods.

SafetyDepositBox

`Code : STRING // 4 digits`
`State : STRING // "Open-NoCode", "Open-CodeSet", "Closed"`
`// or "Locked"`

`Create()` `// method to create and initialise an object`
`// if using Python use __init__`
`Reset()` `// clears Code`
`SetState()` `// set state to parameter value`
`// and output new state`
`SetNewCode()` `// sets Code to parameter value`
`// output message and new code`
`StateChange()` `// reads keypad and takes appropriate action`

Write **program code** for the following methods.

Programming language .....

(i) `Create()`

.....

.....

.....

.....

..... [3]

(ii) `Reset()`

.....

.....

..... [2]

(iii) `SetState()`

.....

.....

.....

..... [2]

(iv) `SetNewCode()`

.....

.....

.....

..... [2]

(v) The user must enter a 4-digit code.

Write **program code** for a function `Valid(s : STRING)` that returns:

- `TRUE` if the input string `s` consists of exactly 4 digits
- `FALSE` otherwise

Programming language .....

.....

.....

.....

.....

.....

.....

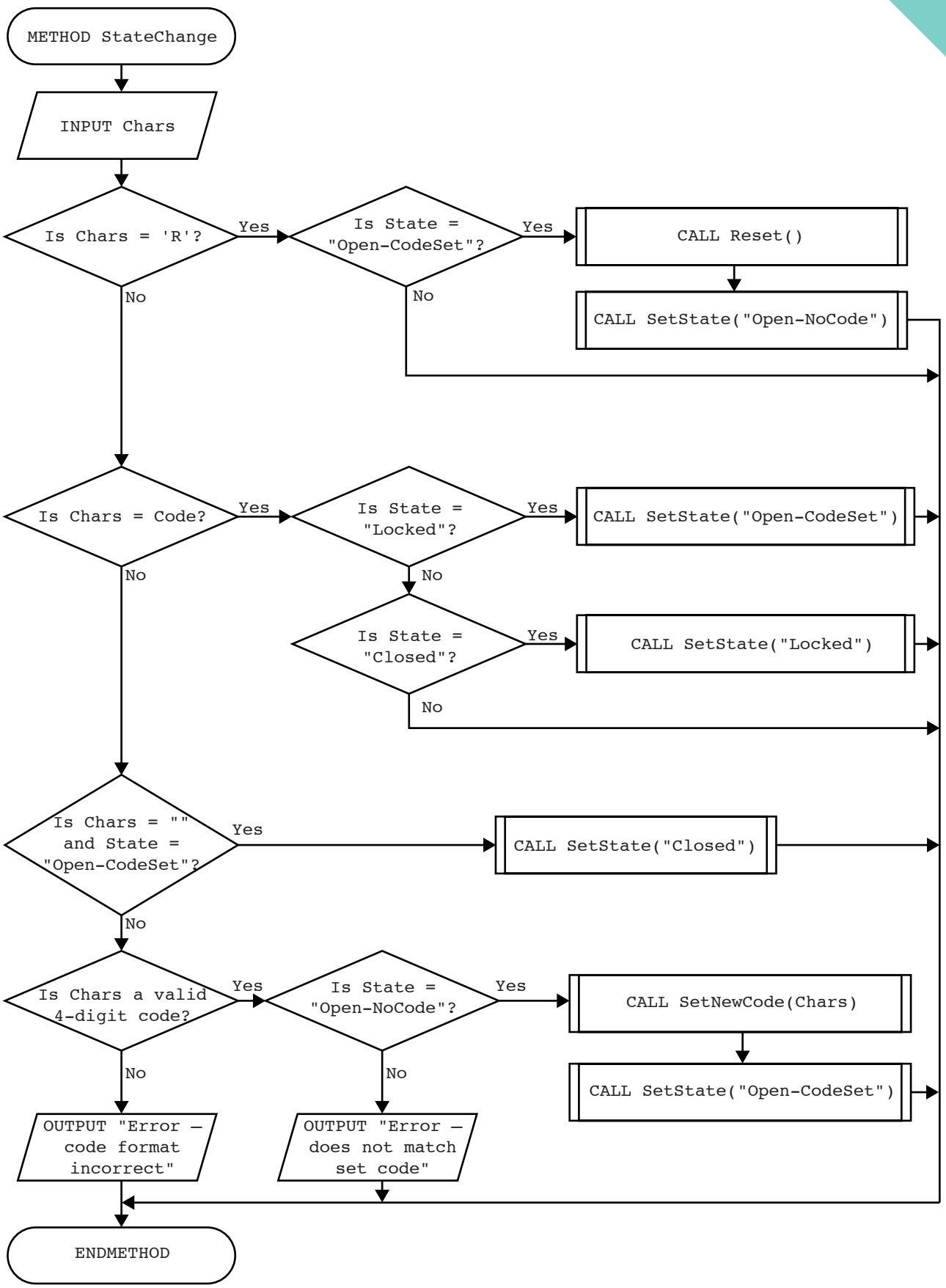
.....

.....

.....

..... [4]

(vi) Convert the flowchart to **program code** for the method `StateCha` properties and methods in the original class definition and the `Valid()` **part (v)**.



Programming language ....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[12]

(vii) The company needs to write a program to simulate a safety deposit b
will create an object with identifier `ThisSafe`, which is an instance
`SafetyDepositBox`.

The main program design is:

```
instantiate ThisSafe (create and initialise ThisSafe)
loop forever (continually use ThisSafe)
   call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(c) It is possible to declare properties and methods as either public or private.

The programmer has modified the class design for `SafetyDepositBox` as follow

SafetyDepositBox
PRIVATE Code : STRING State : STRING
PUBLIC Create() StateChange() PRIVATE Reset() SetState() SetNewCode()

(i) Describe the effects of declaring the `SafetyDepositBox` properties as private.

.....

.....

.....

..... [2]

(ii) Describe the effects of declaring two methods of the class as public and the other three as private.

.....

.....

.....

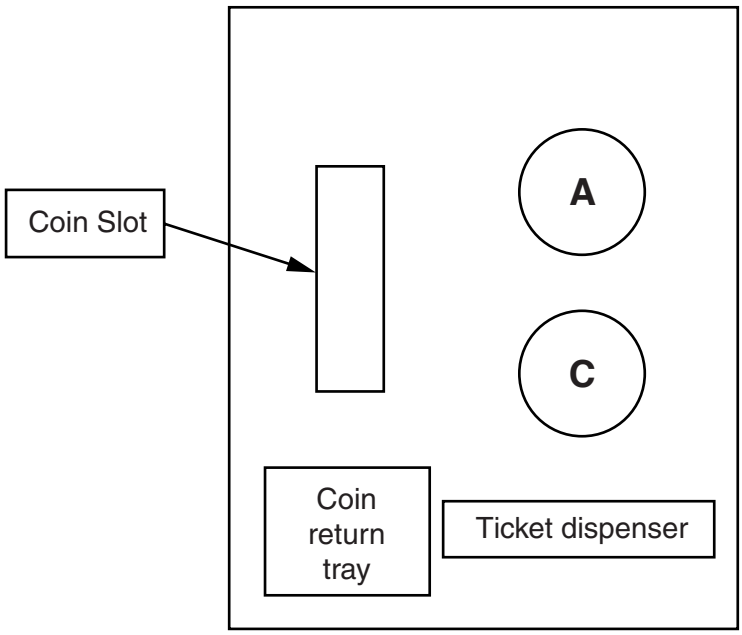
..... [2]

QUESTION 3.

1 The ticket machine in the following diagram accepts the following coins: 10, 20, 5

The ticket machine has:

- a slot to insert coins
- a tray to return coins
- a ticket dispenser
- two buttons:
  - button **A** (Accept)
  - button **C** (Cancel)



When the user has inserted as many coins as required, they press button **A** to print the ticket.

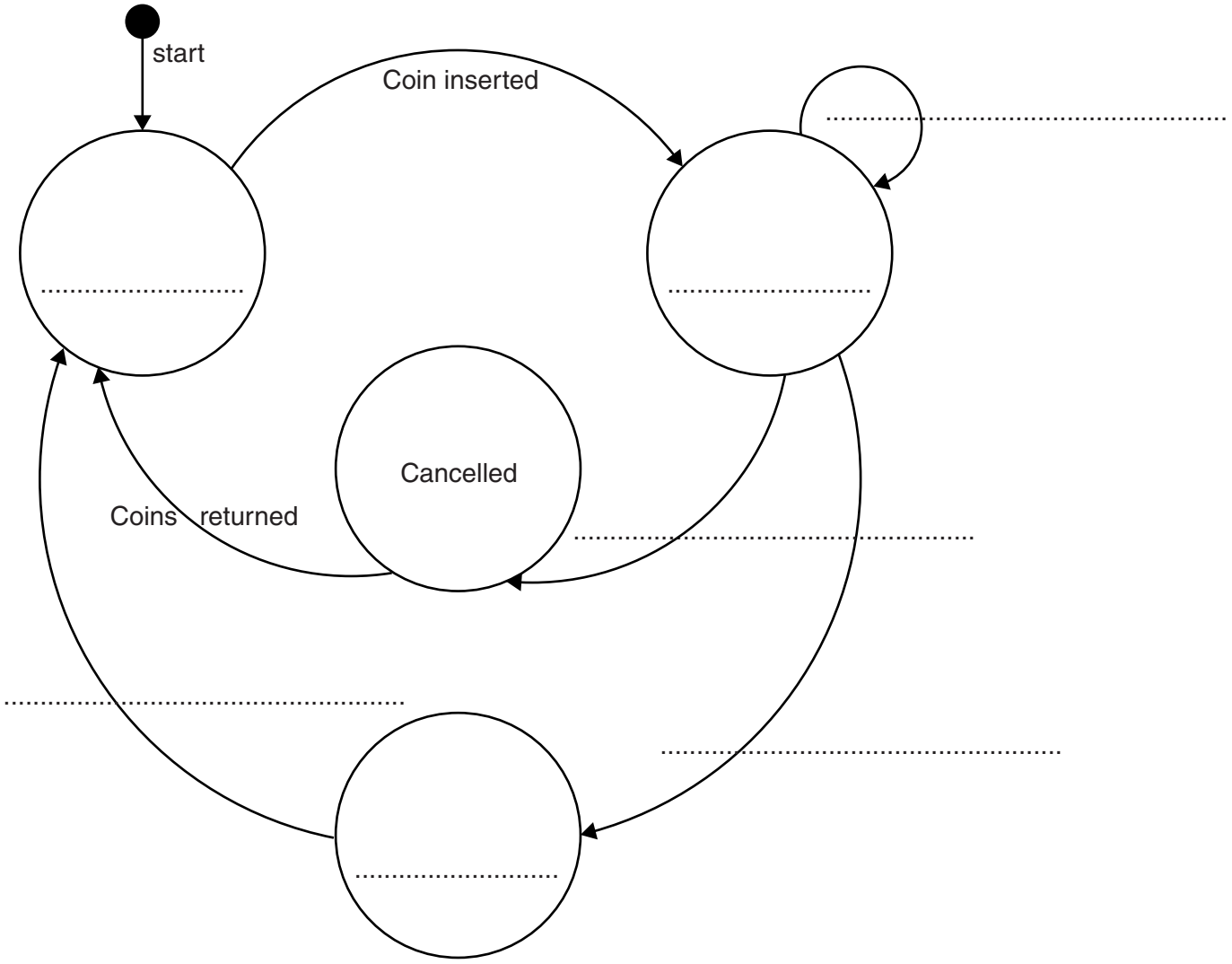
To cancel the transaction, the user can press button **C**. This makes the machine return the coins.

Invalid coins have no effect.

The following state transition table shows the transition from one state to anot
machine:

Current state	Event	Next state
Idle	Coin inserted	Counting
Counting	Coin inserted	Counting
Counting	Button **C** pressed	Cancelled
Cancelled	Coins returned	Idle
Counting	Button **A** pressed	Accepted
Accepted	Ticket printed	Idle

(a) Complete the state-transition diagram.



[7]

(b) A company wants to simulate the use of a ticket machine. It will do this wit
programming (OOP).

The following diagram shows the design for the class `TicketMachine`. This inc
attributes and methods.

TicketMachine

`Amount : INTEGER // total value of coins inserted in cents`
`State : STRING // "Idle", "Counting", "Cancelled"`
`// or "Accepted"`

`Create()` // method to create and initialise an object
// if using Python use __init__
`SetState()` // set state to parameter value
// and output new state
`StateChange()` // insert coin or press button,
// then take appropriate action
`CoinInserted()` // parameter is a string
// change parameter to integer
// and add coin value to Amount
`ReturnCoins()` // output Amount, then set Amount to zero
`PrintTicket()` // print ticket, then set Amount to zero

Write **program code** for the following methods.

Programming language .....

(i) `Create()`

.....

.....

.....

.....

.....[3]

(ii) `SetState()`

.....

.....

.....

.....[2]

(iii) `ReturnCoins()`

.....

.....

.....

..... [2]

(iv) Each coin inserted must be one of the following: 10, 20, 50 or 100 cents.

Write **program code** for a function `ValidCoin(s : STRING)` that returns:

- `TRUE` if the input string is one of `"10"`, `"20"`, `"50"` or `"100"`
- `FALSE` otherwise

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[3]

(v) Write **program code** for the method `CoinInserted()`

.....

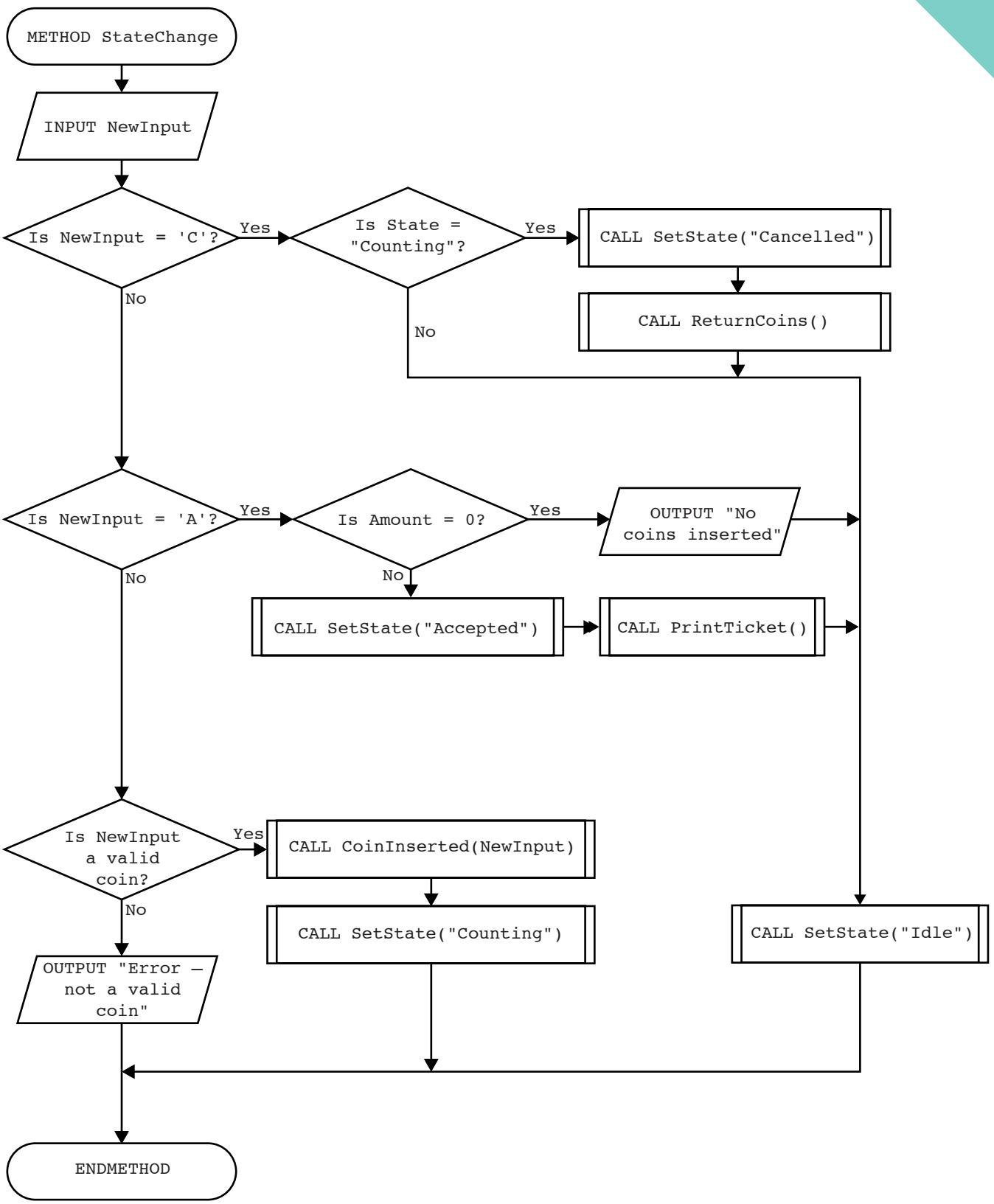
.....

.....

.....

.....[2]

(vi) Convert the flowchart to **program code** for the method `StateChange()`.
Use the attributes and methods in the original class definition and the `V`
function from **part (iv)**.



Programming language ....

....

....[12]

(vii) The company needs to write a program to simulate a parking met
will create an object with identifier `ParkingMeter`, which is an instanc
`TicketMachine`.

The main program design is:

```
instantiate ParkingMeter (create and initialise ParkingMeter)
loop forever (continually use ParkingMeter)
   call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

(c) It is possible to declare attributes and methods as either public or private.

A programmer has modified the class design for `TicketMachine` as follows.

TicketMachine
PRIVATE Amount : INTEGER State : STRING
PUBLIC Create() StateChange() PRIVATE SetState() CoinInserted() ReturnCoins() PrintTicket()

(i) Describe the effects of declaring the `TicketMachine` attributes as private.

.....

.....

.....

.....[2]

(ii) Describe the effects of declaring two methods of the class as public and the other four as private.

.....

.....

.....

.....[2]

QUESTION 4.

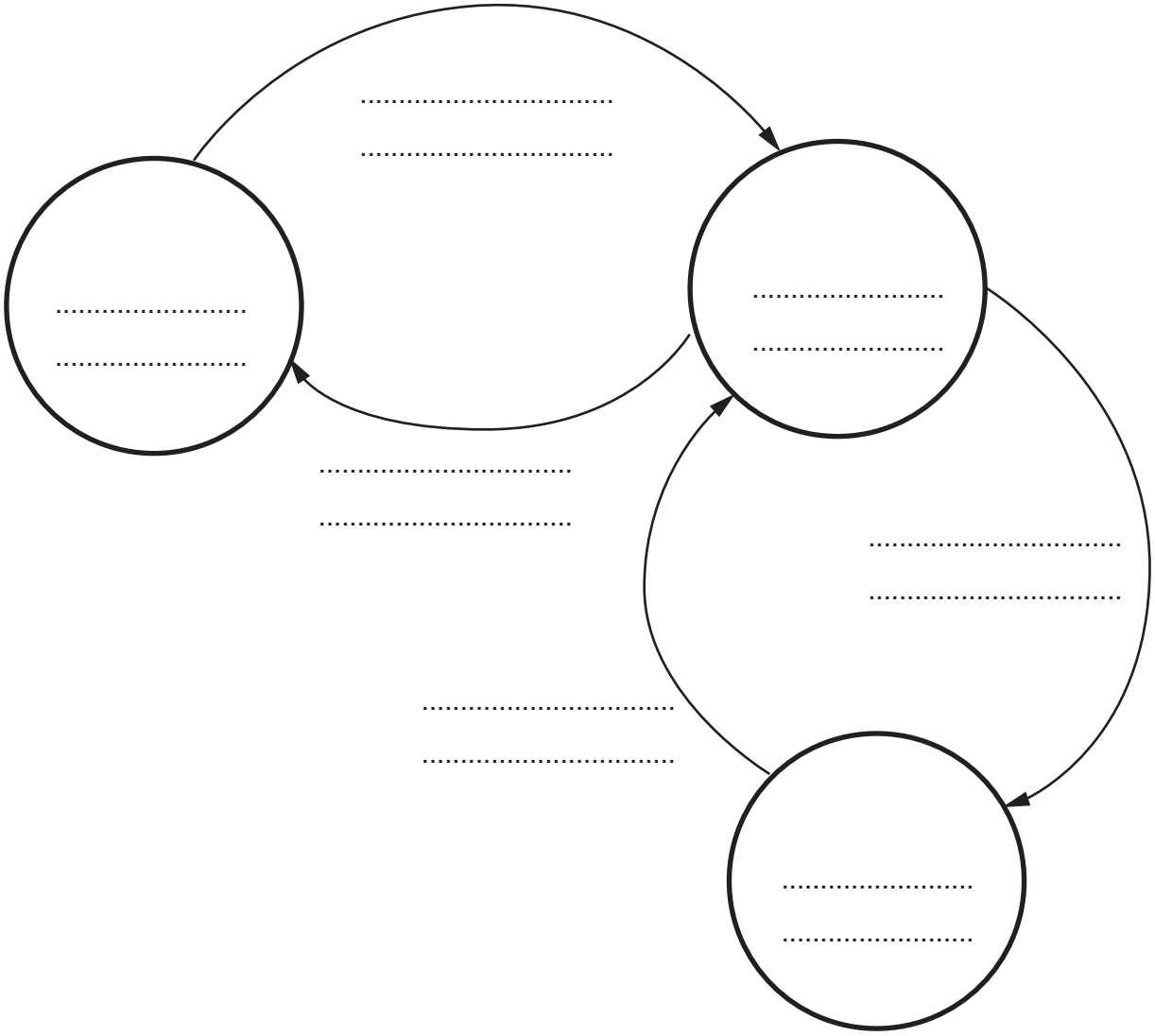
1 A greenhouse has a window that automatically opens and closes depending temperature.

If the temperature rises above 20 °C, the window half opens. If the temperature rises abo the window fully opens. If the temperature drops below 25 °C, the window returns to bei open. If the temperature drops below 15 °C, the window fully closes.

The window has three possible states: **Closed**, **Half Open** and **Fully Open**.

Current state	Event	Next state
Closed	Temperature rises above 20 °C	Half Open
Half Open	Temperature drops below 15 °C	Closed
Half Open	Temperature rises above 30 °C	Fully Open
Fully Open	Temperature drops below 25 °C	Half Open

Complete the state-transition diagram for the window:



[7]

QUESTION 5.

1 Each student at CIE University needs a printing account to print documents computers.

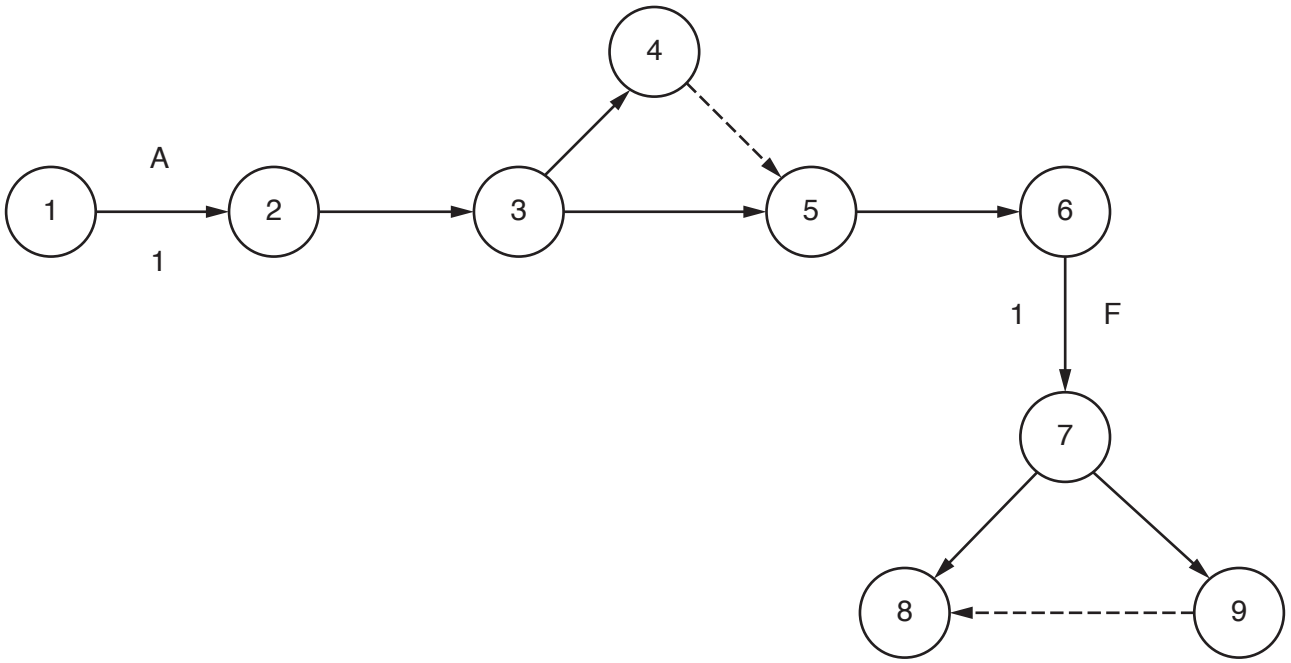
The university is developing software to manage each student's printing account and the process.

(a) Developing the software will include the following activities.

Activity	Description	Time in weeks	Predecessor
A	Identify requirements	1	-
B	Produce design	3	A
C	Write code	10	B
D	Test modules	7	B
E	Final system black-box testing	3	C, D
F	Install software	1	E
G	Acceptance testing	2	F
H	Create user documentation	2	F

(i) Add the correct activities and times to the following Program Evaluation Review Technique (PERT) chart for the software development.

Two activities and times have been done for you.



[6]

(ii) State what is meant by the **critical path** in a PERT chart.

.....

.....

(iii) Identify **and** describe a project planning technique, other than a PERT chart.

.....

.....

.....

..... [2]

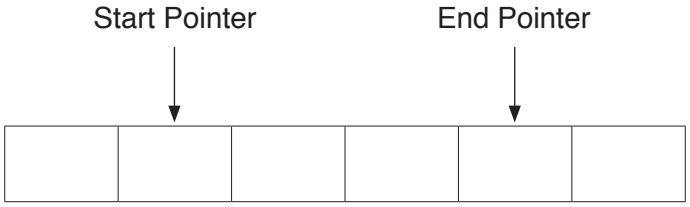
(b) When a student prints a document, a print job is created. The print job is sent to a print server.

The print server uses a queue to hold each print job waiting to be printed.

(i) The queue is circular and has six spaces to hold jobs.

The queue currently holds four jobs waiting to be printed. The jobs have arrived in the order A, B, D, C.

Complete the diagram to show the current contents of the queue.



[1]

(ii) Print jobs A and B are now complete. Four more print jobs have arrived in the order E, F, G, H.

Complete the diagram to show the current contents and pointers for the queue.

[3]

(iii) State what would happen if another print job is added to the queue in the status in **part (b)(ii)**.

.....

..... [1]

(iv) The queue is stored as an array, `Queue`, with six elements. The fol
removes a print job from the queue and returns it.

Complete the following **pseudocode** for the function `Remove`.

```
FUNCTION Remove RETURNS STRING
   DECLARE PrintJob : STRING
   IF ..... = EndPointer
      THEN
         RETURN "Empty"
      ELSE
         PrintJob ← Queue[.....]
         IF StartPointer = .....
            THEN
               StartPointer ← .....
            ELSE
               StartPointer ← StartPointer + 1
         ENDIF
         RETURN PrintJob
   ENDIF
ENDFUNCTION
```

[4]

(v) Explain why the circular queue could not be implemented as a stack.

.....

.....

.....

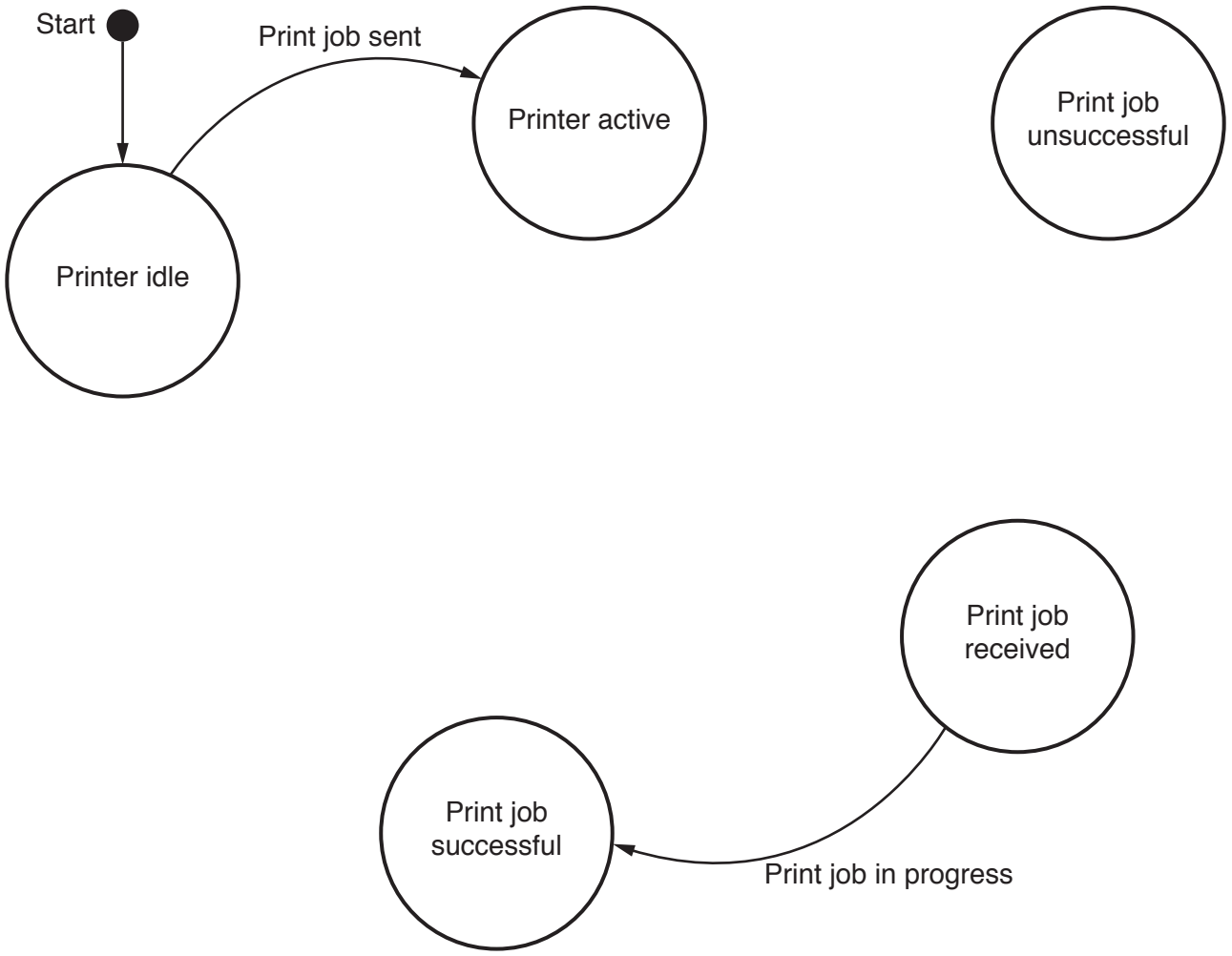
..... [2]

(c) The university wants to analyse how a printer and a print server deal with the

The following table shows the transitions from one state to another for the process

Current state	Event	Next state
Printer idle	Print job sent	Printer active
Printer active	Print job added to queue	Print job received
Print job received	Print job in progress	Print job successful
Print job received	Print job in progress	Print job unsuccessful
Print job successful	Check print queue	Printer active
Print job unsuccessful	Error message displayed	Printer active
Printer active	Timeout	Printer idle

Complete the state-transition diagram for the table.



[5]

(d) The university wants to assess troubleshooting issues with a printer. It wants ... table to do this.

The troubleshooting actions are:

- check the connection from computer to printer, if the error light is flashing **and** ... document has not been printed
- check the ink status, if the quality is poor
- check whether there is a paper jam, if the error light is flashing **and** the document has not been printed
- check the paper size selected, if the paper size is incorrect.

(i) Describe the purpose of a decision table.

.....

.....

.....

..... [2]

(ii) Complete the rules for the actions in the following decision table.

		Rules							
Conditions	Document printed but the quality is poor	Y	Y	Y	Y	N	N	N	N
	Error light is flashing on printer	Y	Y	N	N	Y	Y	N	N
	Document printed but paper size is incorrect	Y	N	Y	N	Y	N	Y	N
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[4]

(iii) Simplify your solution by removing redundancies.

		Rules							
Conditions	Document printed but the quality is poor								
	Error light is flashing on printer								
	Document printed but paper size is incorrect								
Actions	Check connection from computer to printer								
	Check ink status								
	Check if there is a paper jam								
	Check the paper size selected								

[5]

(e) There are 1000 students at the university. They will each require a printing a

Students need to buy printing credits that will be added to their account. Each p uses one printing credit.

The university needs software to keep track of the number of printing credits each student in their account. The university has decided to implement the software using object-oriente programming (OOP).

The following diagram shows the design for the class `PrintAccount`. This includes the attributes and methods.

PrintAccount
FirstName : STRING // parameter sent to Constructor()
LastName : STRING // parameter sent to Constructor()
PrintID : STRING // parameter sent to Constructor()
Credits : INTEGER // initialised to 50

Constructor() // instantiates an object of the PrintAccount class,
// and assigns initial values to the attributes

GetName() // returns FirstName and LastName concatenated
// with a space between them

GetPrintID() // returns PrintID
SetFirstName() // sets the FirstName for a student
SetLastName() // sets the LastName for a student
SetPrintID() // sets the PrintID for a student
AddCredits() // increases the number of credits for a student
RemoveCredits() // removes credits from a student account

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

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) Write **program code** for the `SetFirstName()` method.

Programming language .....

Program code

.....

.....

.....

.....

..... [2]

(iii) Write **program code** for the `GetName()` method.

Programming language .....

Program code

.....

.....

.....

.....

.....

..... [2]

(iv) The method `AddCredits()` calculates the number of printing credits and adds the printing credits to the student's account.

- Credits cost $1 for 25 credits.
- If a student buys $20 or more of credits in a single payment, they receive an extra 50 credits.
- If a student buys between $10 and $19 (inclusive) of credits in a single payment, they receive an extra 25 credits.

Payment from a student is stored in the variable `MoneyInput`. This is passed as a parameter.

Write **program code** for `AddCredits()`. Use constants for the values that do not change.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

(v) A global array, `StudentAccounts`, stores 1000 instances of `PrintAc`

Write **pseudocode** to declare the array `StudentAccounts`.

.....

..... [2]

(vi) The main program has a procedure, `CreateID()`, that:

- takes the first name and last name as parameters
- creates `PrintID` that is a concatenation of:
  - the first three letters of the first name in lower case
  - the first three letters of the last name in lower case
  - the character '1'
    for example, the name Bill Smith would produce `"bilsmi1"`
- checks if the `PrintID` created already exists in the global array `StudentAccounts`:
  - If `PrintID` does not exist, it creates an instance of `PrintAccount` in the next free index in `StudentAccounts`.
  - If `PrintID` does exist, the number is incremented until a unique ID is created, for example, `"bilsmi2"`. It then creates an instance of `PrintAccount` in the next free index in `StudentAccounts`.

The global variable `NumberStudents` stores the number of print accounts that have currently been created.

Write **program code** for the procedure `CreateID()`. Do not write the procedure header.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

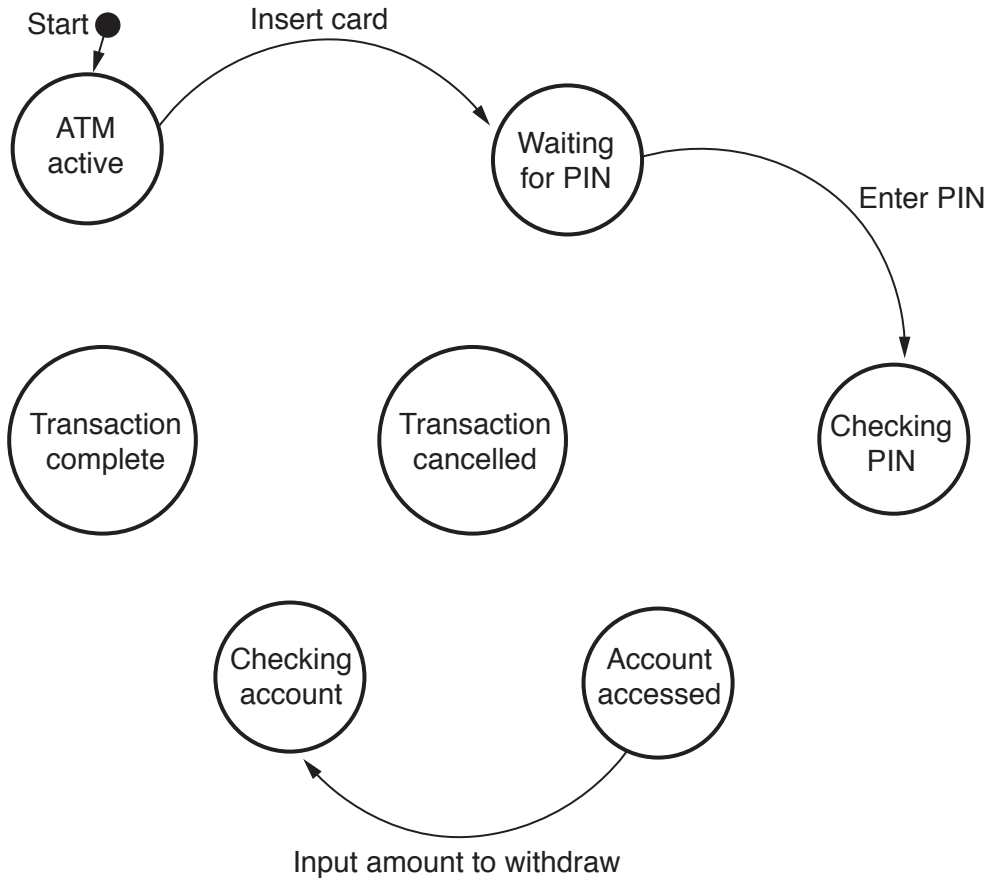
QUESTION 6.

4 A bank wants to analyse how an automated teller machine (ATM) deals with trans

The following state-transition table shows the transitions from one state to another for a t

Current state	Event	Next state
ATM active	Insert card	Waiting for PIN
Waiting for PIN	Enter PIN	Checking PIN
Waiting for PIN	Cancel selected	Transaction cancelled
Checking PIN	PIN valid	Account accessed
Checking PIN	PIN invalid	Waiting for PIN
Account accessed	Cancel selected	Transaction cancelled
Account accessed	Input amount to withdraw	Checking account
Checking account	Funds available	Transaction complete
Transaction complete	Return card and dispense cash	ATM active
Checking account	Funds not available	Account accessed
Transaction cancelled	Return card	ATM active

Complete the state-transition diagram to correspond with the table.



[8]