







Syllabus Content:

12.2 Program Design

Candidates should be able to:

Use a structure chart to decompose a problem into sub-tasks and express the parameters passed between the various modules / procedures / functions which are part of the algorithm design

Notes and guidance

-  Describe the purpose of a structure chart
-  Construct a structure chart for a given problem
-  Derive equivalent pseudocode from a structure chart
-  Show understanding of the purpose of state-transition diagrams to document an algorithm

12.2.1 Structure charts:

An alternative approach to modular design is to choose the sub-tasks and then construct a structure chart to show the interrelations between the modules. Each box of the structure chart represents a module. Each level is a refinement of the level above.

A structure chart also shows the interface between modules, the variables. These variables are referred to as 'parameters'.

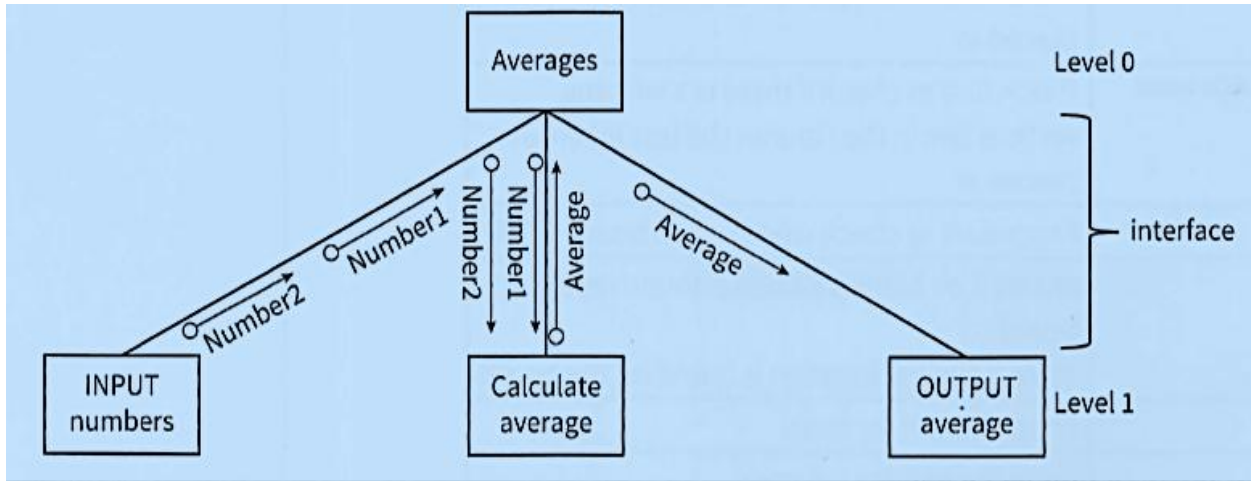
A **parameter** supplying a value to a lower-level module is shown as a downwards pointing arrow. A parameter supplying a new value to the module at the next higher level is shown as an upward pointing arrow.

Structure chart: a graphical representation of the modular structure of a solution

Parameter: a value passed between modules

Figure below shows a structure chart for a module that calculates the average of two numbers. The top-level box is the name of the module, which is refined into the three subtasks of Level 1. The input numbers (parameters **Number1** and **Number2**) are passed into the '**Calculate Average**' sub-task and then the **Average parameter** is passed into the '**OUTPUT Average**' sub-task. The arrows show how the parameters are passed between the modules.

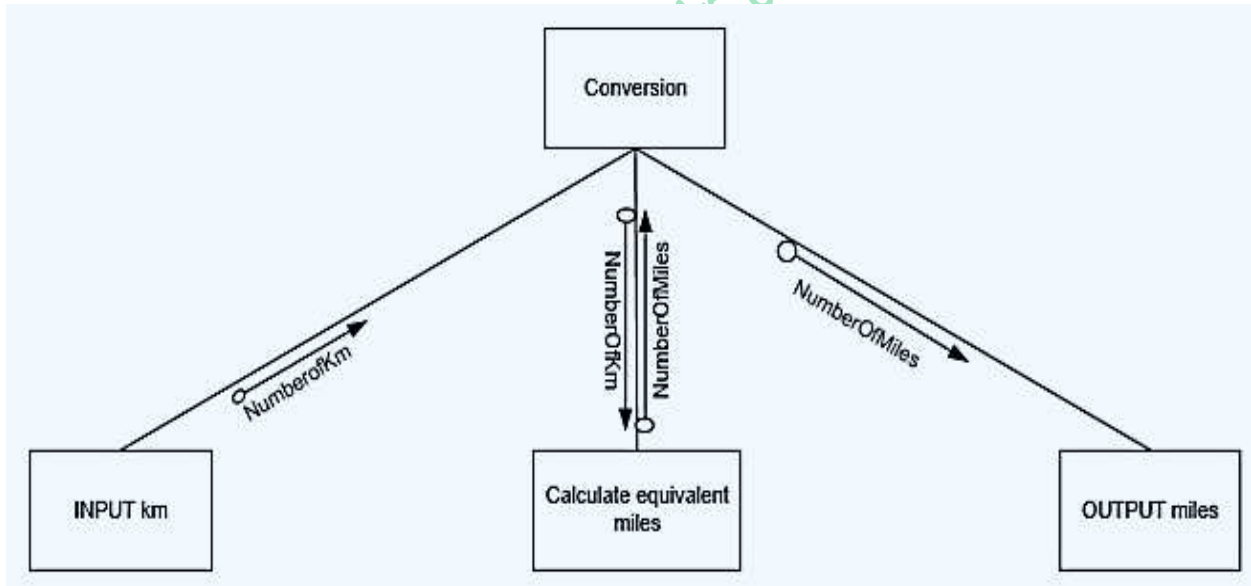
This parameter passing is known as the '**interface**'.



TASK 12.03

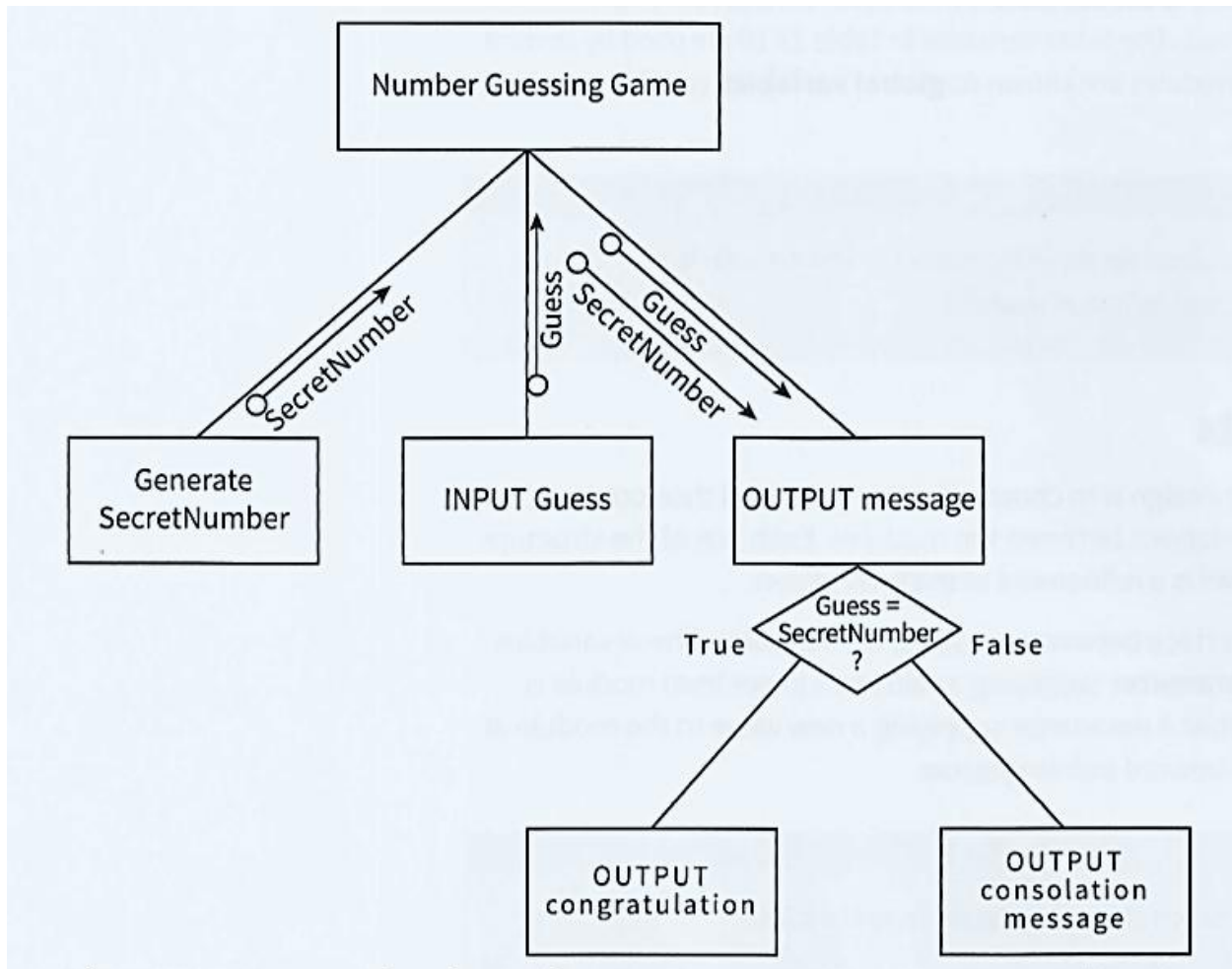
Draw a structure chart for the following module: Input a number of km, output the equivalent number of miles.

Answer:



Structure charts can also show control information: selection and repetition.

The simple number-guessing game could be modularised and presented as a structure chart, as shown in Figure below



Structure chart for number-guessing game with only one guess allowed

Past paper questions on structure charts:

MJ 2015/ 21

Q 3:

When the guarantee on a computer runs out, the owner can take out insurance to cover breakdown and repairs.

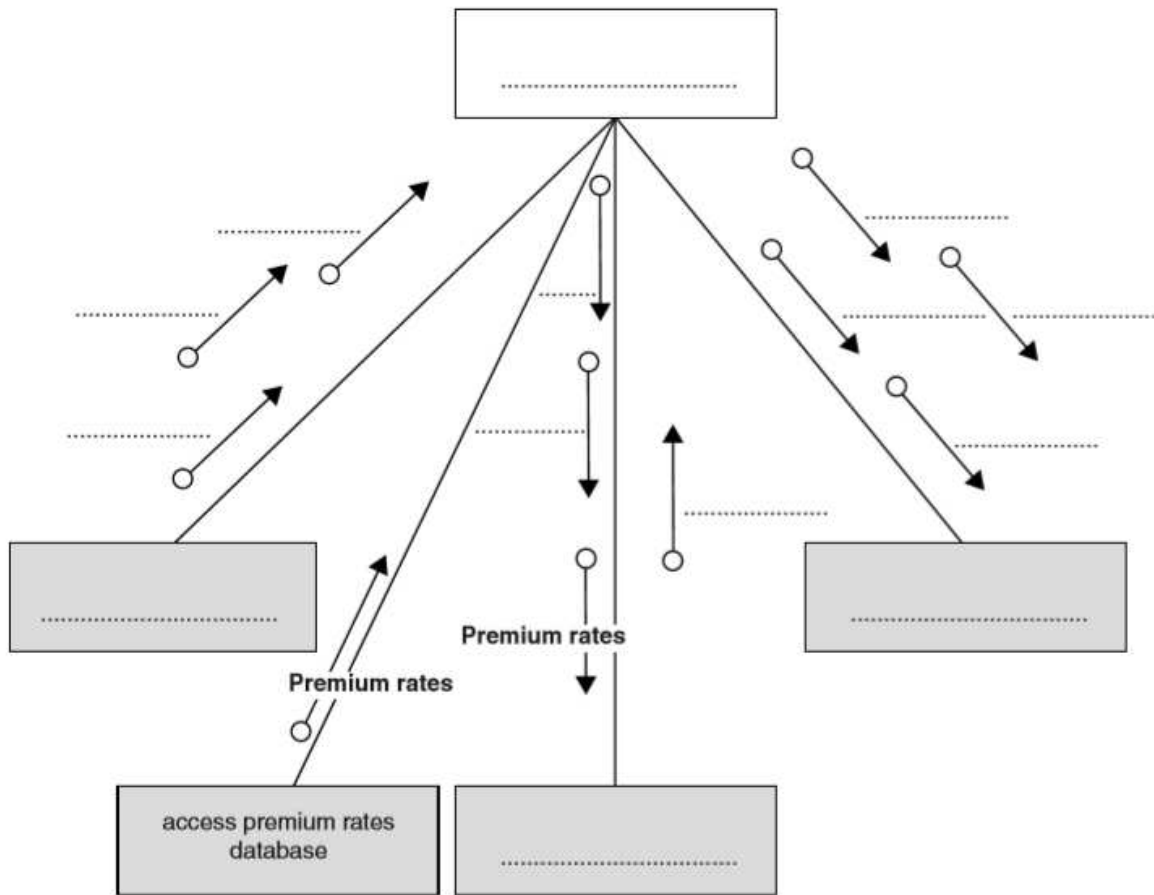
The price of the insurance is calculated from:

- the model of the computer
- the age of the computer
- the current insurance rates

Following an enquiry to the insurance company, the customer receives a quotation letter with the price of the insurance.

A program is to be produced.

The structure chart below shows the modular design for this process:



(a) Using the letters A to D, add the labelling to the chart boxes on the opposite page.

Modules	
A	Send quotation letter
B	Calculate price
C	Produce insurance quotation
D	Input computer details

[2]

(b) Using the letters E to J, complete the labelling on the chart opposite.

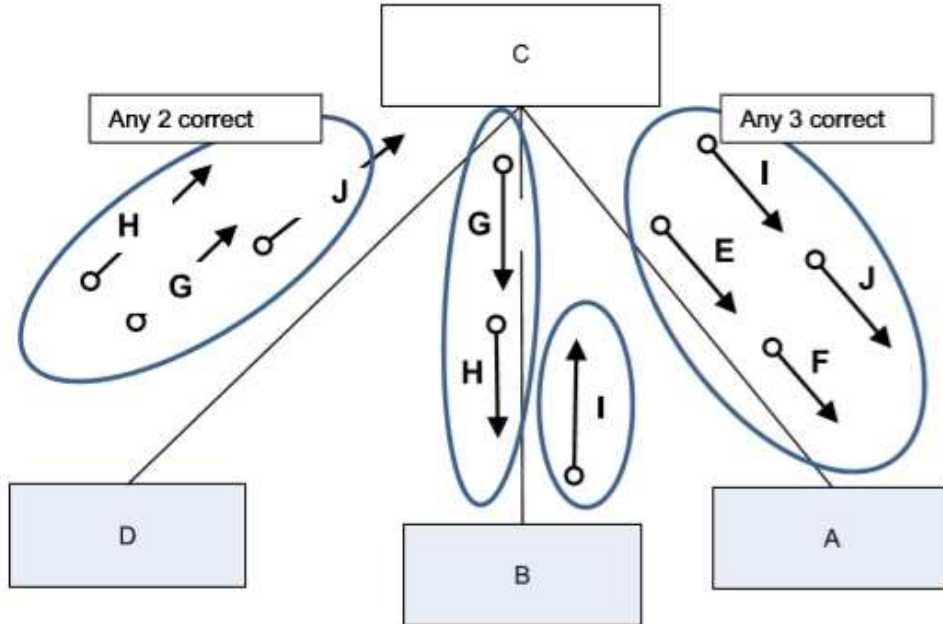
Some of these letters will be used more than once.

Data items	
E	CustomerName
F	CustomerEmail
G	Model
H	Age
I	PolicyCharge
J	PolicyNumber

[4]

Answer:

(b)



Data items	
E	CustomerName
F	CustomerEmail
G	Model
H	Age
I	PolicyCharge
J	PolicyNumber

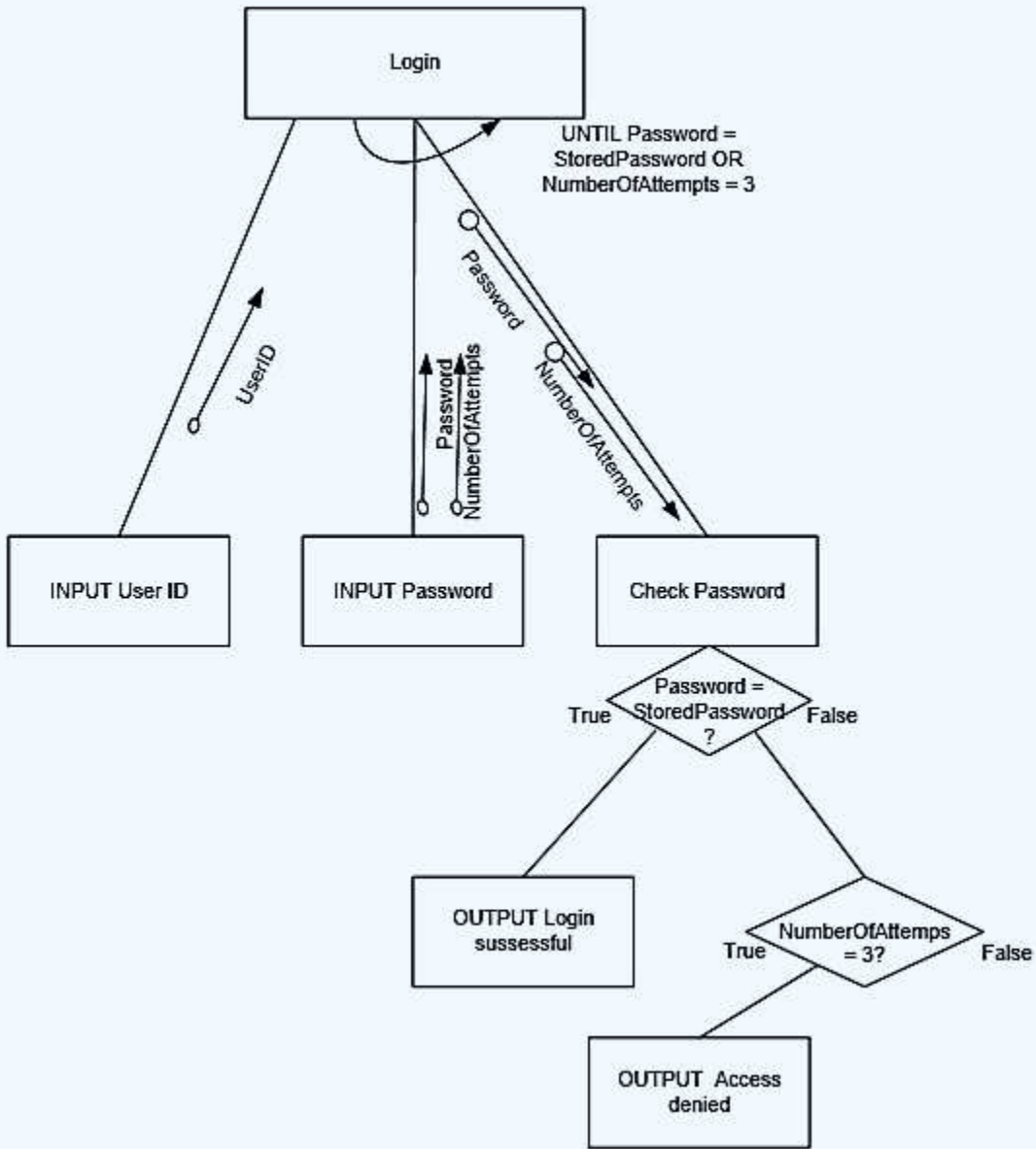
[4]

TASK 12.05

Draw a structure chart for the following problem: A user attempts to log on with a user ID. User IDs and passwords are stored in two 1D arrays (lists). The algorithm searches the list of user IDs and looks up the password in the password list. The user is given three chances to input the correct password. If the correct password is entered, a suitable message is output. If the third attempt is incorrect, a warning message is output.

Answer:

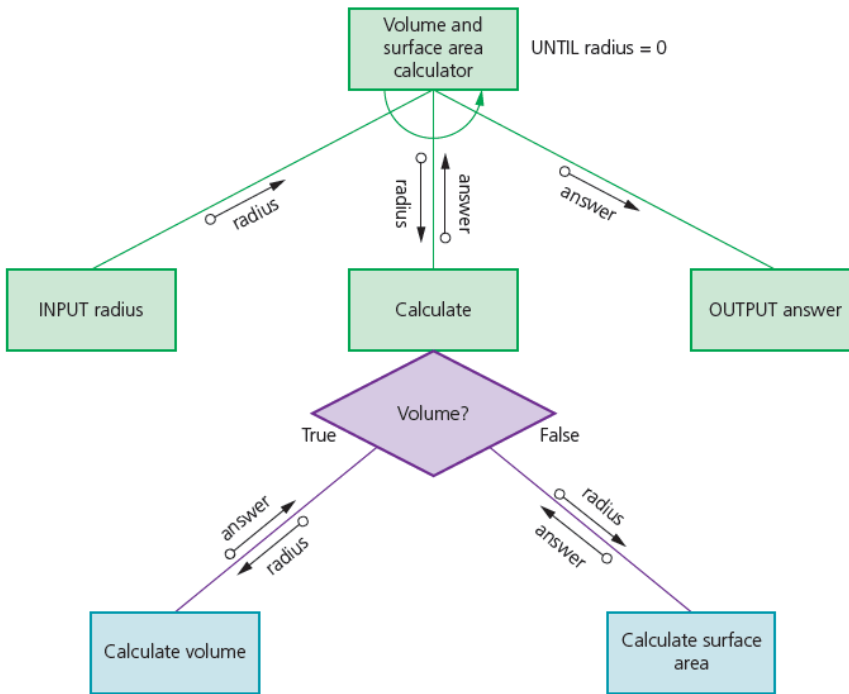
Task 12.05



Comp



Consider structure chart:



To derive **pseudocode** from **structure chart**, we need to make identifiers:

Identifier name	Description
radius	Stores radius input
answer	Stores result of calculation
pi	Constant set to 3.142

Declaring constants and variables in Pseudocodes:

```

DECLARE radius: REAL
DECLARE radius: REAL
CONSTANT pi ← 3.142

```

Making PSEUDOCODE for Structure Chart above:

```

FUNCTION calculateVolume (radius: REAL ) RETURNS : REAL
RETURN (4/3) * pi * radius * radius * radius
END FUNCTION

```

```

FUNCTION calculateSurfaceArea (radius: REAL ) RETURNS : REAL
RETURN (4/3) * pi * radius * radius
END FUNCTION

```

The Input/Output modules can be made as Procedures

```

PROCEDURE InputRadius
OUTPUT ("Please Enter radius of Sphere")
INPUT radius
WHILE radius < 0 Do
OUTPUT ("Please Enter a positive number")
INPUT radius
END WHILE
END PROCEDURE

```





```
PROCEDURE outputAnswer
    OUTPUT answer
END PROCEDURE
```

Combining all the program

```
DECLARE radius: REAL
DECLARE radius: REAL
CONSTANT pi ← 3.142
```

```
FUNCTION calculateVolume (radius: REAL ) RETURNS : REAL
RETURN (4/3) * pi * radius * radius * radius
END FUNCTION
```

```
FUNCTION calculateSurfaceArea (radius: REAL ) RETURNS : REAL
RETURN (4/3) * pi * radius * radius
END FUNCTION
```

```
PROCEDURE InputRadius
    OUTPUT ("Please Enter radius of Sphere")
    INPUT radius
        WHILE radius < 0 Do
            OUTPUT ("Please Enter a positive number")
            INPUT radius
        END WHILE
END PROCEDURE
```

```
PROCEDURE outputAnswer
    OUTPUT answer
END PROCEDURE
```

```
CALL inputRadius
```

```
WHILE radius <> 0 Do
    OUTPUT ("Do you want to Calculate Volume (V) or Surface Area (S)")
    INPUT reply
        IF reply = "V"
            THEN
                answer ← calculateVolume(radius)
                OUTPUT "Volume"
            ELSE
                answer ← calculateSurfaceArea(radius)
                OUTPUT "Surface Area"
            END IF
        CALL outputAnswer
        CALL inputRadius
    END WHILE
```


Syllabus Content:

12.2 Program Design

State-transition diagrams



use state-transition diagrams to document an algorithm



use state-transition diagrams to show the behaviour of an object

12.2.2: State- transition diagrams:

A computer system can be seen as a finite state machine (FSM). An FSM has a start state. An input to the FSM produces a transformation from one state to another state.




The information about the states of an FSM can be presented in a state-transition table.

KEY TERMS

Finite state machine (FSM): a machine that consists of a fixed set of possible states with a set of inputs that change the state and a set of possible outputs


State-transition table: a table that gives information about the states of an FSM

Table shows an example FSM represented as a state-transition table. If the FSM is in state S1, an input of a causes no change of state.

-  If the FSM is in state S1, an input of b transforms S1 to S2.
-  If the FSM is in state S2, an input of b causes no change of state.
-  If the FSM is in state S2, an input of a transforms S2 to S1.

		current state	
		S1	S2
input	a	S1	S1
	b	S2	S2

A state-transition diagram can be used to describe the behaviour of Table 24.05 State-transition table an FSM.

Figure STD shows the start state as S1 (denoted by ). If the FSM has a final state (also known as the halting state), this is shown by a double-circled state (S1 in the example).

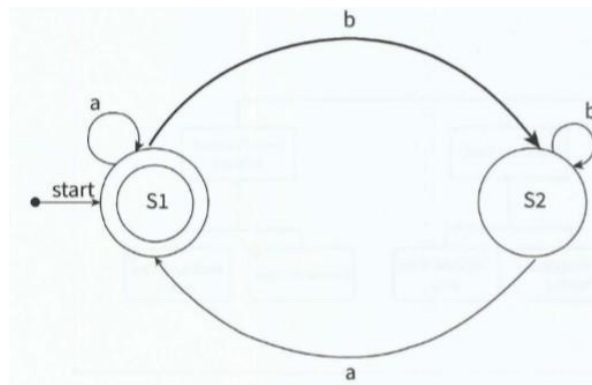


Figure STD

If an input causes an output this is shown by a vertical bar, For example, if the current state is S1, an input of b produces output c and transforms the FSM to state S2.

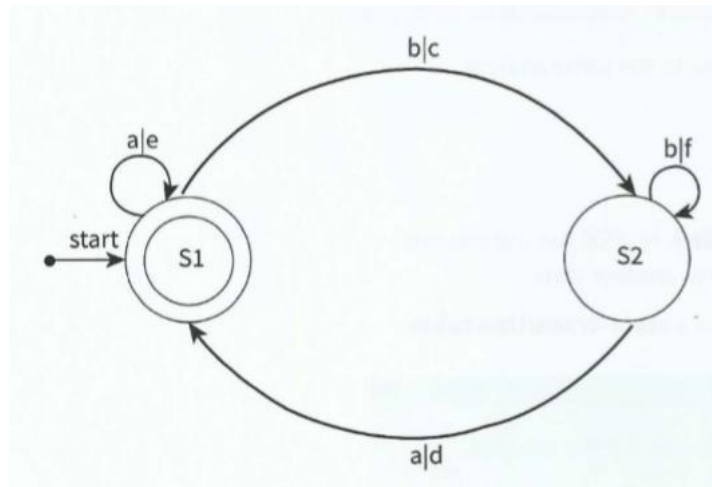


Figure State-transition diagram with outputs

Creating a state-transition diagram for an intruder detection system

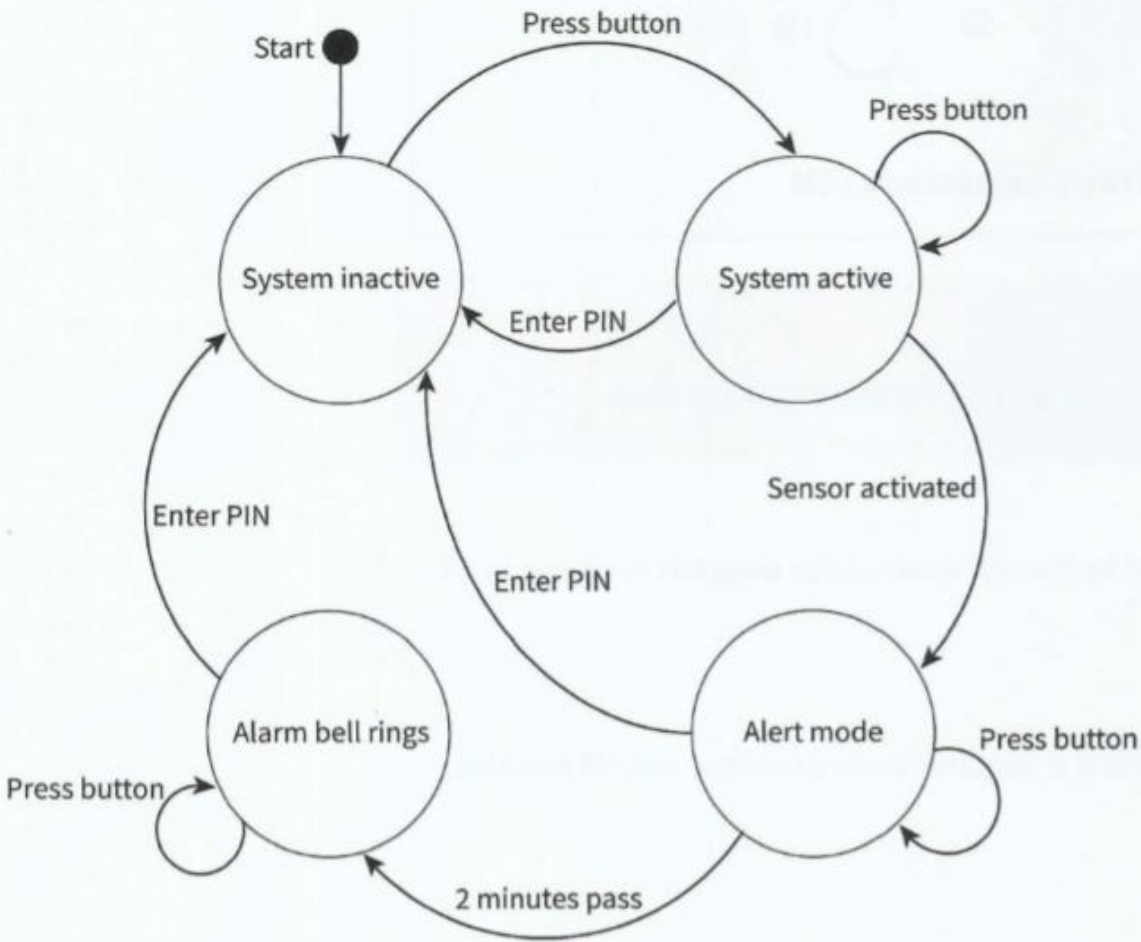
A program is required that simulates the behaviour of an intruder detection system.

Description of the system: The system has a battery power supply. The system is activated when the start button is pressed. Pressing the start button when the system is active has no effect. To de-activate the system, the operator must enter a PIN. The system goes into alert mode when a sensor is activated. The system will stay in alert mode for two minutes. If the system has not been de-activated within two minutes an alarm bell will ring.

We can complete a state-transition table (Table) using the information from the system description.

Current state	Event	Next state
System inactive	Press start button	System active
System active	Enter PIN	System inactive
System active	Activate sensor	Alert mode
System active	Press start button	System active
Alert mode	Enter PIN	System inactive
Alert mode	2 minutes pass	Alarm bell ringing
Alert mode	Press start button	Alert mode
Alarm bell ringing	Enter PIN	System inactive
Alarm bell ringing	Press start button	Alarm bell ringing

The start state is 'System inactive'. We can draw a state-transition diagram (Figure 24.07) from the information in Table



Past Paper Questions:
9608/41/M/J/15

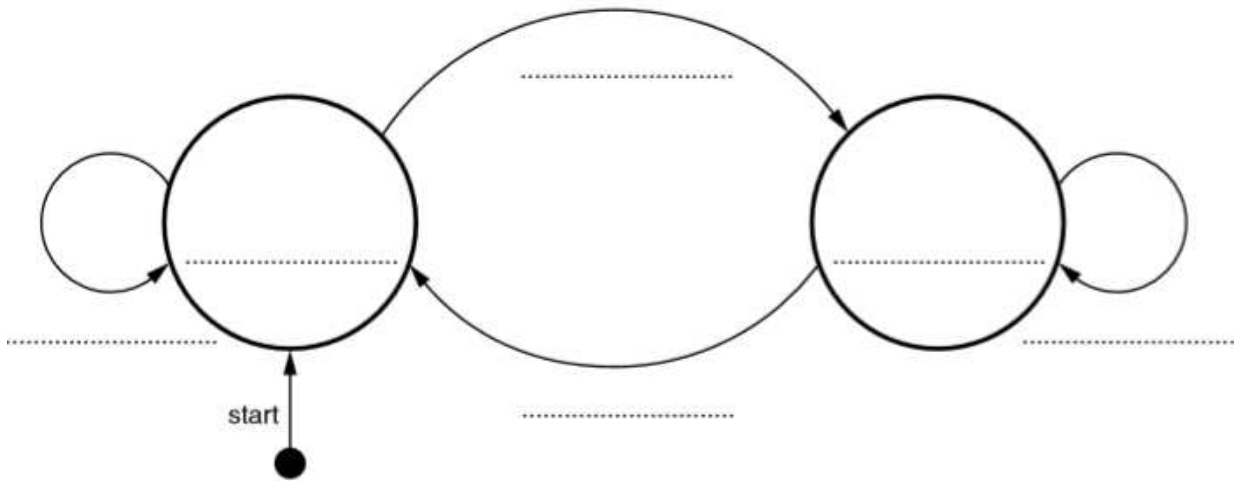
1/- A turnstile is a gate which is in a locked state. To open it and pass through, a customer inserts a coin into a slot on the turnstile. The turnstile then unlocks and allows the customer to push the turnstile and pass through the gate.

After the customer has passed through, the turnstile locks again. If a customer pushes the turnstile while it is in the locked state, it will remain locked until another coin is inserted.

The turnstile has two possible states: **locked** and **unlocked**. The transition from one state to another is as shown in the table below.

Current state	Event	Next state
Locked	Insert coin	Unlocked
Locked	Push	Locked
Unlocked	Attempt to insert coin	Unlocked
Unlocked	Pass through	Locked

Complete the state transition diagram for the turnstile:



[5]

(9608/43/M/J/15)

Q2/ - A petrol filling station has a single self-service petrol pump.

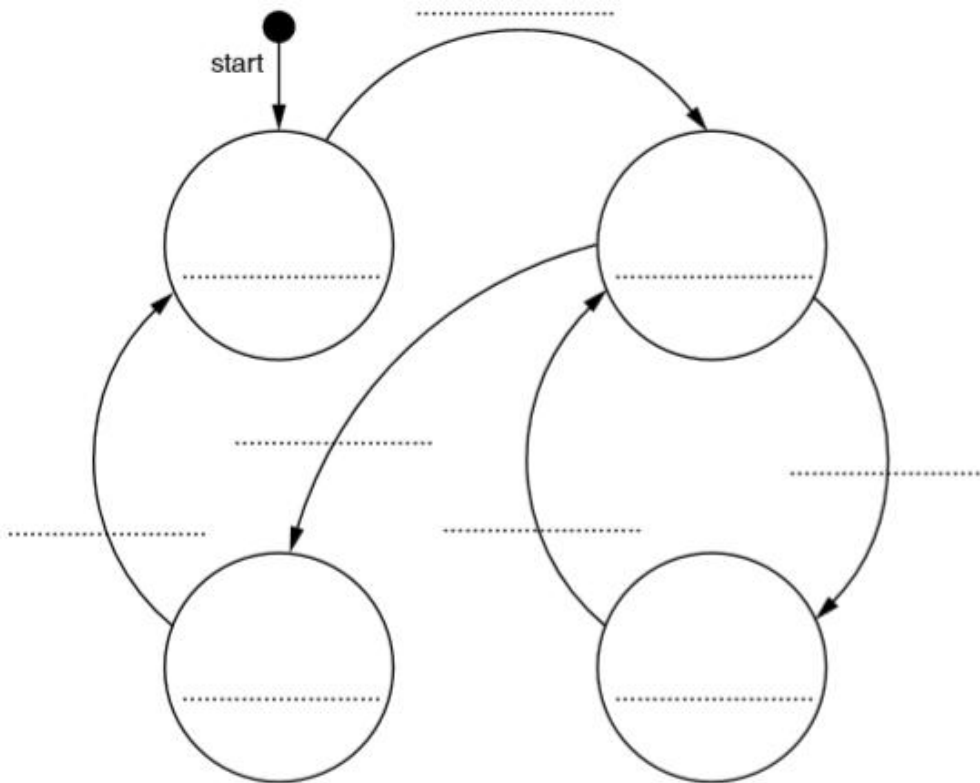
A customer can use the petrol pump when it is ready to dispense petrol. The pump is in use when the customer takes the nozzle from a holster on the pump. The pump dispenses petrol while the customer presses the trigger on the nozzle. When the customer replaces the nozzle into the holster, the pump is 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:



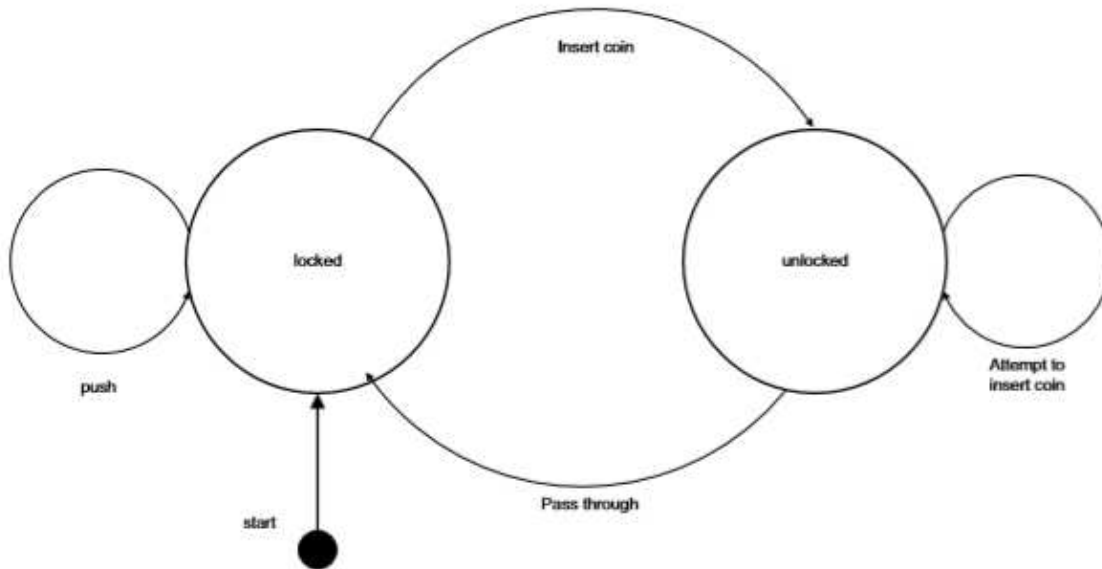
[9]

Answers
9608/41/M/J/15





1

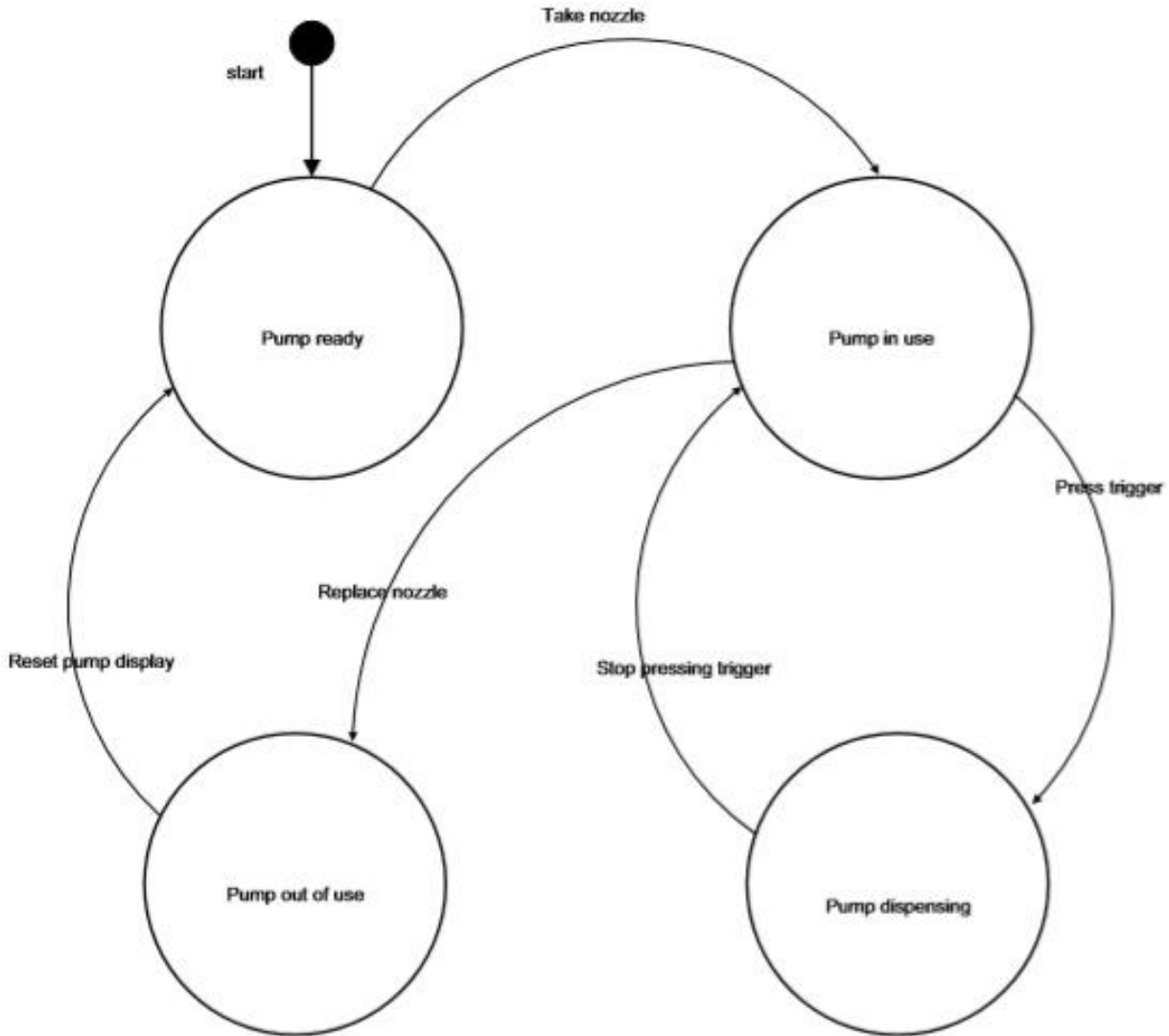


Mark as follows:
1 mark for both states correct
1 mark for each further label

[5]



(9608/43/M/J/15)

2



[9]

References:

-  AS & A level Course Book by Sylvia Langfield & Dave Duddell
-  A level 9608 Pastpapers