CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTING **9691/21**

Paper 2 **October/November 2011**

2 hours

Candidates answer on the Question Paper.

No additional materials are required.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name on all the work you hand in.
Write in dark blue or black pen.
You may use a soft pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, highlighters, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.

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.

1 Ahmed is writing a program to record the data of members of the school football squad. The input data will need to be validated. One input will be the position of each player. Is he/she a forward, defender or goalkeeper? These will be entered as f, d or g. The flowchart for the validation of the position is shown below.
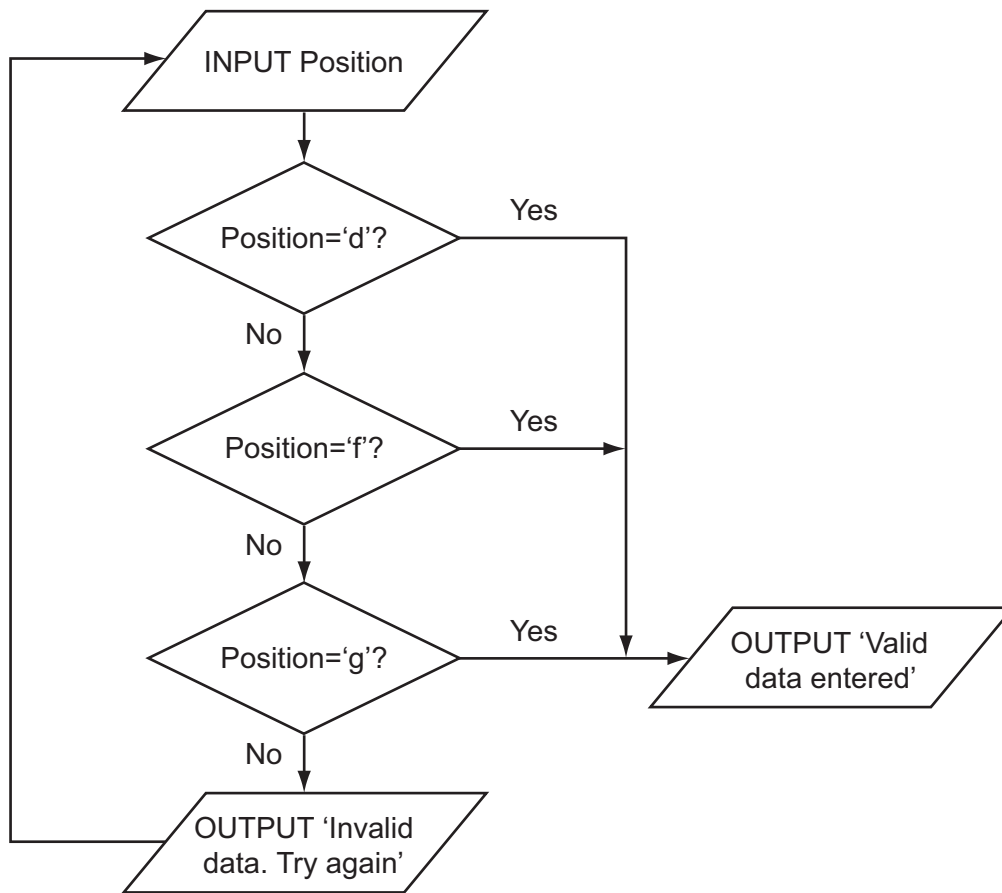
(a) (i) What is the output when the input is 'g'?

.... [1]

(ii) What is the output when the input is 'x'?

.... [1]

(b) In a high-level language, write the code that will produce the validation process shown in the flowchart.

Language ....

Code ....

....

....

....

....

....

....

....

.... [5]

(c) The three basic programming constructs used to control the flow of information are: sequence, selection and iteration.

State the **two** constructs that are used in your code.

1 ....

2 .... [1]

(d) Describe what is meant by iteration.

....

....

....

.... [2]

(e) Ahmed thinks it will be a good idea to allow only three attempts at getting the input data correct. If it is not a valid entry after three attempts, a message 'Please check which values are allowed' should be output.

Modify the flowchart to include this additional check.

```
                    ┌────────────────────┐
          ┌────────▶│  INPUT Position    │
          │         └────────────────────┘
          │                   │
          │                   ▼
          │              ╱─────────╲        Yes
          │             ╱ Position= ╲──────────────┐
          │             ╲  'd'?     ╱              │
          │              ╲─────────╱               │
          │                  │ No                  │
          │                  ▼                     │
          │              ╱─────────╲        Yes    │
          │             ╱ Position= ╲──────────▶   │
          │             ╲  'f'?     ╱              │
          │              ╲─────────╱               │
          │                  │ No                  │
          │                  ▼                     │
          │              ╱─────────╲   Yes    ┌──────────────────┐
          │             ╱ Position= ╲────────▶│  OUTPUT 'Valid   │
          │             ╲  'g'?     ╱         │  data entered'   │
          │              ╲─────────╱          └──────────────────┘
          │                  │ No
          │                  ▼
          │         ┌────────────────────┐
          └─────────│  OUTPUT 'Invalid   │
                    │  data. Try again'  │
                    └────────────────────┘
```
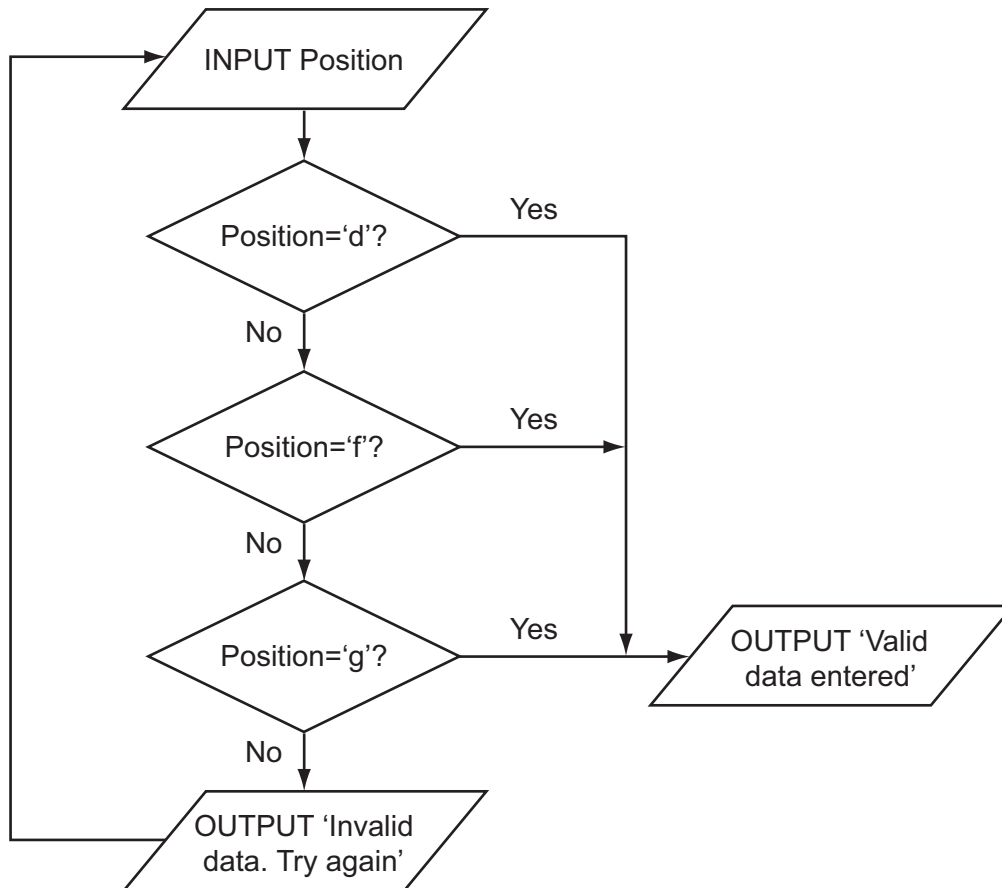
[5]

(f) Ahmed needs to store more information about the players. He creates a record structure that contains PlayerID (a whole number between 1 and 50), Sex (m or f), PlayerName, Position (f, d or g), and DateOfBirth.

Complete the table.

Field Name	Data Type	Field Size (bytes)
PlayerID		
Sex		
PlayerName		
Position		
DateOfBirth		

[10]

2 Nathan is designing a software solution for stock control in a mobile phone shop. He has a colleague, called Andre, who will help him write the program. Nathan decides to modularise the solution.

(a) State why modularisation is a good idea.

....

.... [1]

(b) As the first step in his design he splits the solution into the following main areas:

Initialisation, PhoneSales, StockOrdering, Accounts.

Complete the following structure diagram.

```
                        ┌─────────────────┐
                        │  Stock Control  │
                        └────────┬────────┘
          ┌──────────────┬───────┴──────┬──────────────┐
   ┌──────┴──────┐ ┌─────┴──────┐ ┌─────┴──────┐ ┌─────┴──────┐
   │             │ │ PhoneSales │ │            │ │            │
   └─────────────┘ └────────────┘ └────────────┘ └────────────┘
```
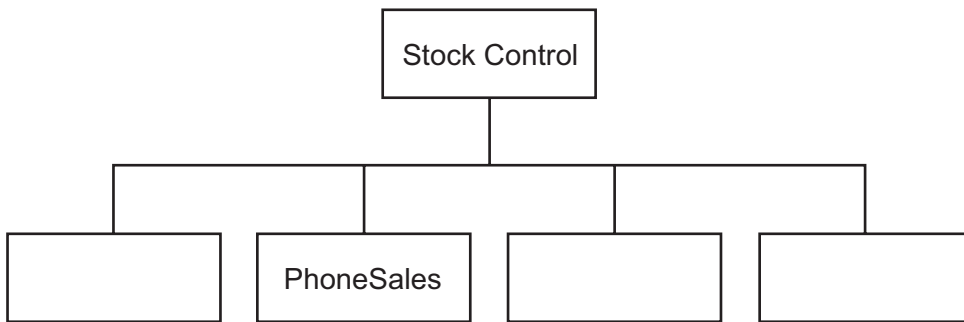
[1]

(c) PhoneSales is made up of two modules, ShopSales and OnlineSales. Add them to the structure diagram shown in **(b)**. [2]

(d) Nathan will write the ShopSales module and Andre will write the OnlineSales module. Nathan will use the identifier *Sale* for a sale in the shop, and Andre will use the identifier *Sale* for an online order.

Explain how they can both use the same identifier and not cause a problem when the program is run.

....

....

....

....

.... [2]

(e) Both programmers need to choose other identifiers that they will use.

   (i) Explain why there are some words that cannot be used as identifiers.

   ....

   ....

   ....

   .... [1]

   (ii) State **three** other rules of a high-level programming language that restrict the choice of identifiers.

   Language ....

   Rule 1 ....

   ....

   Rule 2 ....

   ....

   Rule 3 ....

   .... [3]

   (iii) Give an example of an **invalid** identifier.

   Language ....

   Example .... [1]

(f) One line in the program reads:

ForecastStock = 5*a-b.

    (i) Work out the value of ForecastStock when a is 4 and b is 3.

        ....

        ....

        .... [1]

    (ii) The programmer has made an error in the code. When a is 4 and b is 3 the expected result is 5.

        Rewrite the line of code with added parentheses to give the expected result.

        .... [1]

    (iii) Name the type of testing strategy which identified this error.

        .... [1]

(g) One type of test data is invalid data.

(i) Name the other **two** types.

1 ....

2 .... [2]

(ii) Andre has written the StockOrdering module, which now needs testing.

- The StockID is a whole number between 1000 and 9999
- The ReOrderLevel is between 10% and 20%

Give **six** different items of test data, other than invalid data, which thoroughly test the two rules given above. Give a reason for each choice.

StockID	ReOrderLevel	Reason
50		Invalid data for StockID
	21%	Invalid data for ReOrderLevel

[6]

(h) (i) When dealing with the Accounts module, an entry, 'y' or 'n' has to be input, indicating whether the accounts should be printed. The variable identifier is PrintAccounts.

Write a Boolean expression to validate an input for PrintAccounts.

....

....

.... [2]

(ii) Write a Boolean expression to validate StockID, as described in (g).

....

....

.... [2]

(iii) Describe a possible problem that could occur if the StockID is not validated.

....

....

....

....

....

.... [2]

3 Aisha is learning about manipulating strings in a high-level programming language.
She has an idea that she wants to try. She produces the following design in pseudocode:

```
FUNCTION Surprise(MyWord : STRING): STRING
   LOOP FOR Counter ← 1 TO LENGTH(MyWord)
      Temp[Counter] ← MyWord[LENGTH(MyWord) – (Counter-1)]
   ENDLOOP
   Surprise ← Temp
ENDFUNCTION
```

(a) Complete the trace of this function for the function call `Surprise('rac')`.

MyWord	Counter	LENGTH (MyWord)	Temp[1]	Temp[2]	Temp[3]	Surprise
rac						
	1					

[3]

(b) State the purpose of this function.

....

.... [1]

(c) Describe what `Surprise ← Temp` does.

....

....

....

.... [2]

(d) High-level languages also use procedures.

Describe the features of a procedure.

....

....

....

....

....

.... [3]

(e) Aisha has written her pseudocode so that it should be easily understood.

(i) State **two** techniques that she has used in her pseudocode to do this.

1 ....

....

2 ....

.... [2]

(ii) One other technique to help understanding is to annotate using comments.

Write a suitable comment to annotate the line:

```
Temp[Counter] ← MyWord[LENGTH(MyWord) – (Counter-1)]
```

....

....

....

.... [2]

(f) Aisha uses an expression:

```
String1 > String2
```

Explain how strings are compared by the processor.

....

....

....

....

....

.... [3]

BLANK PAGE

BLANK PAGE