2.3.1 Programming Basics

Oct/NOV 2009. P11

10. Part of the processing of the data is to calculate the amount of tax which needs to be paid.

- The person's total income for the year is input to the system
- The first $500 is not taxed
- The remainder is taxed at 10%
- Either the tax to be paid is output from the system OR a message is output to say there is no tax to pay

(a) Using I to stand for the total income and T to stand for the tax to be paid, produce an algorithm which will take I as its input and then calculate the tax.

(b) Explain why I and T are unsuitable as variable names and say how they can be improved. **[2]**

May/June 2011. P23

3. Kris has written a program that will work out the wages for her staff. The main steps for each employee are: to work out the hours worked, work out the total earnings, work out tax and finally print out how much will be taken home.

(e) (i) Explain how it is possible to use the same variable name for different variables in different modules. **[2]**

(ii) Explain how this can help when modules are programmed separately. **[3]**

Oct/Nov 2011. P21

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.

(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 **[3]**

(iii) Give an example of an invalid identifier. **[1]**

Oct/NOV 2012.P21

1 Soni works for a software house which has been asked to design software for a cycle hire company, Super Bikes.

Soni decides on the main tasks:

2.3.1 Programming Basics

- collecting the information about new bikes
- entering details of repairs
- entering details of hirer
- entering details of payment

(d) Each subtask is coded as a program module.

Explain how it is possible to use the same variable name for different variables in different modules. **[2]**

2 (b) (ii) Describe what is meant by a reserved word. **[2]**

May/June 2013. P21/22

3 Meena needs to be aware of her average grade and declares a variable `AvMark`, which she decides will be a global variable.

(a) State where in the program a global variable is declared. **[1]**

(b) Using only global variables is poor programming practice.

Give a possible problem that could result from this. **[1]**

(c) Good programming practice uses local variables.

What is the scope of a local variable? **[1]**

May/June 2013. P23

3 Meena needs to be aware of her average mark and declares a variable with identifier `MyAvMark` which she decides will be a global variable.

(a) State where in the program a global variable will be declared. **[1]**

(b) Using only global variables is poor programming practice.

Give a possible problem that could result from this. **[1]**

Oct/Nov 2013.P21

2 (a) Juan has little programming experience, but has to write code for this program. He has written the following pseudocode statements.

For each statement describe what is wrong and write a correct version.

(i) `IF Index > 5000 OR < 0 THEN OUTPUT "Error"` **[2]**
(ii) `NumberOfCopies[Index] + 1 ← NumberOfCopies[Index]` **[2]**
(iii) `NumberOfCopies[Index] ← "three"` **[2]**

2.3.1 Programming Basics

Oct/Nov 2013.P22

1 (c) (i) Jemma is using the variable `Percentage` in one module and a different variable called `Percentage` in another module.

Explain the feature of a high-level programming language that avoids any possible conflict. **[2]**

Oct/Nov 2013.P23

1 The Science Department has a problem keeping track of its equipment. Ashvin has been asked to design and program a solution as his Computing Project.

(a) Ashvin has little programming experience, but has to write code for this program. He starts by writing some pseudocode.
For each statement, describe what is wrong and write a correct version.

(i) `NoOfBalances + 1 ← NoOfBalances` **[2]**

(ii) `IF NoOfPipettes > NoOfBeakers OR < NoOfBottles THEN OUTPUT "Check the numbers"` **[2]**

4 Ashvin is also learning about recursion. He writes the pseudocode for a recursive function.

```
1   FUNCTION Calc(X)
2   DECLARE Temp
3       IF X > 0
4           THEN
5               Temp ← X + Calc(X – 2)
6           ELSE
7               Temp ← 0
8       ENDIF
9   RETURN Temp
10  ENDFUNCTION
```

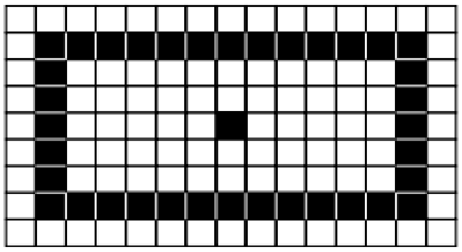
(a) What is the scope of the variable `Temp`? **[1]**

(b) State the line number which shows that this function is recursive. **[1]**

2.3.1 Programming Basics

Oct/Nov 2014. P21

4 Ahmed combines white tiles with tiles of one other colour to make a pattern. He draws a design. Here is one example:



Ahmed wants a program to calculate how many tiles he needs of each colour.

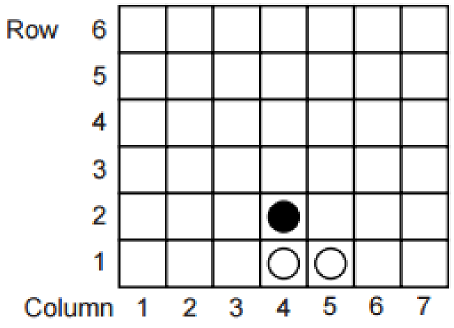
(a) Ali stores the design in a 2-dimensional array, `FloorDesign`. The length and width of the room will be no more than 35 tiles each.

(i) Suggest a value Ali should use to represent the white and coloured tiles. **[1]**

Oct/Nov 2014.P22

3 A game is played by two players. Player A uses white tokens (○). Player B uses black tokens (●). The players take turns dropping tokens into a vertical grid. The tokens fall straight down and occupy the next available space in the chosen column. The aim of the game is to connect four of one's own colour tokens. This must be done in a vertical, horizontal or diagonal line.

Here is one example after Player A has had 2 turns and Player B has had 1 turn:



2.3.1 Programming Basics

Nathan wants to write a program to allow two users to play the game on the computer.

The program will display a simplified version of the above grid which is redrawn after every turn.

(a) Before any tokens have been dropped into the grid, all grid cells are empty.

(i) Suggest values Nathan should use to represent:

Empty cell ....

White token ....

Black token .... **[2]**

(iii) Write the statement to assign the value for a black token to the cell shown in the diagram. **[2]**

May/June 2015.P21/P22

3 A board game is designed for two players, O and X.

At the beginning, all cells of a 3 x 3 grid are empty.

The players take turns in placing their marker in an empty cell of the grid; player O always starts.

The game ends when one player completes a row, column or diagonal or the grid is full.

Here is one example after three turns:

		O
	O	X

Ali wants to write a program to play the game.

(a) The array `Grid` is to be used to represent the contents of the grid.

Rows and columns are to be numbered from 1 to 3.

(i) To take their turn, the player inputs a row number and a column number to place their marker in an empty cell.
Write the values player X has input to place their marker, 'X', in the above diagram:
Row ....

Column .... [1]

(ii) State the value Ali could use to represent an empty cell. [1]

(c) (c) Ali uses the top-down design approach for his overall program solution.

His design is as follows:

```
01 GameEnd ← FALSE
02 CurrentPlayer ← 'O'
03 CALL DisplayGrid()
04
05 REPEAT
06    CALL PlayerTakesTurn(CurrentPlayer)
07    CALL DisplayGrid()
08    IF HasPlayerWon() = TRUE
09      THEN
10         GameEnd ← TRUE
11         OUTPUT "Player", CurrentPlayer, "has won"
12      ELSE
13         IF GridFull() = TRUE
14           THEN
15              GameEnd ← TRUE
16              OUTPUT "Draw"
17           ELSE
18              CALL SwapPlayer(CurrentPlayer)
19         ENDIF
20    ENDIF
21 UNTIL GameEnd = TRUE
```

(v) Complete the identifier table below.

Identifier	**Variable or Procedure or Function or Array**	**Data type**	**Description**
`GameEnd`	Variable	`BOOLEAN`	`FALSE` if game in progress `TRUE` if there is a winner or the grid is full
`Grid`	`ARRAY`		To store the current state of the game
`CurrentPlayer`			The marker value ('O' or 'X') of the current player
`PlayerTakesTurn`			Current player chooses cell Program checks if it is valid and stores marker
`DisplayGrid`			Outputs the contents of the grid
`HasPlayerWon`			Checks if the current player has completed a row, column or diagonal
`GridFull`			Checks if the grid is full
`SwapPlayer`	`PROCEDURE`		Swaps the value of `CurrentPlayer`

[5]

May/June 2015.P23

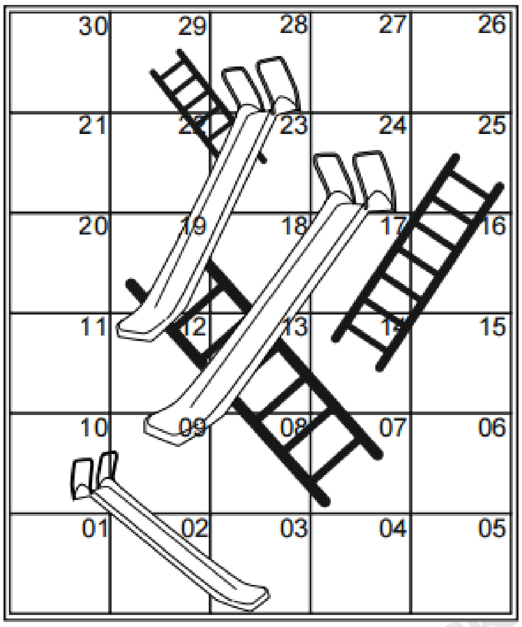
2 Ali has designed a board game.

The board consists of numbered squares.

Slides and ladders connect some of the squares.

An example layout of a board is:

2.3.1 Programming Basics



The object of the game is for a player to move from the bottom square (Square 01) to the top square (Square 30).

A roll of a 6-sided die determines how many squares the player moves.

The table below shows example moves. These use the board layout above, with the player currently on Square 04.

Number rolled by die	Square moved to	Consequence	Final position
2	06	–	06
3	07	The base of a ladder, so climb to top of ladder	19
6	10	The top of a slide, so move down the slide	03

Ali plans to write a program to test different board designs. Each board design will have different numbers, positions and lengths for the slides and ladders.

The program is to test each board design to check that it is possible to complete the game in a reasonable number of moves.

For each board design, the program will simulate playing the game 1000 times and report the average number of moves.

(a) Ali starts the high level design of his program using pseudocode.

```
01 CALL InitialiseArray() // blank board
02 CALL InputBoardDesign() // add slides and ladders data
```

```
03 TotalMoves ← ..........................................................................
04 FOR Game ← .............................................................................
05     // play next game and update TotalMoves
06     TotalMoves ← TotalMoves + NumberOfMovesInThisGame()
07  ..........................................................................................
08 AverageMovesPerGame ← ...............................................................
09 OUTPUT AverageMovesPerGame
```

(a) (v) List the variable identifiers used in the pseudocode. **[2]**

(d) (i) A high-level programming language has a built-in function that generates a random number. The function is defined as follows:

> `RANDOM(n : INTEGER) RETURNS INTEGER`
>
> returns an integer value in the range `0` to `n` inclusive.
>
> For example: `RANDOM(4)` returns `0`, `1`, `2`, `3` or `4`
>
> If the function call is not properly formed an error is generated.

Ali's program is to use this random number generator to simulate the rolling of a die. Complete the statement to assign a number between `1` and `6` to `NumberRolled`.

`NumberRolled` ← ....

.... **[1]**

Computer Science (9608)

May/June 2015.P21/P22

1 A marathon runner records their time for a race in hours, minutes and seconds.

An algorithm is shown below in structured English.

```
INPUT race time as hours, minutes and seconds
CALCULATE race time in seconds
STORE race time in seconds
OUTPUT race time in seconds
```

(a) The identifier table needs to show the variables required to write a program for this algorithm. Complete the table.

Identifier	Data type	Description
`RaceHours`	`INTEGER`	The hours part of the race time.

[3]

(b) Before the program is written, the design is amended.

The new design includes input of the runner's current personal best marathon time (in seconds).

The output will now also show one of the following messages:

- "Personal best time is unchanged"
- "New personal best time"
- "Equals personal best time"

(i) Show the additional variable needed for the new design.

Identifier	Data type	Description

[1]

2.3.1 Programming Basics

5 A company creates two new websites, Site X and Site Y, for selling bicycles.

Various programs are to be written to process the sales data.

These programs will use data about daily sales made from Site X (using variable `SalesX`) and Site Y (using variable `SalesY`).

Data for the first 28 days is shown below:

	SalesDate	SalesX	SalesY
1	03/06/2015	0	1
2	04/06/2015	1	2
3	05/06/2015	3	8
4	06/06/2015	0	0
5	07/06/2015	4	6
6	08/06/2015	4	4
7	09/06/2015	5	9
8	10/06/2015	11	9
9	11/06/2015	4	1
...	∫	∫	∫
28	01/07/2015	14	8

(e) The `DISCOUNT_DATES` text file is successfully created.

The company now wants a program to:

- key in a date entered by the user
- search the text file for this date
- if found, output one of the following messages:
  - "No discount on this date"
  - "This is a discount date"
- if not found, output "Date not found"

(i) Add to the identifier table to show the variables you need for this new program

2.3.1 Programming Basics

Identifier	Data type	Description
DISCOUNT_DATES	FILE	Text file to be used

[3]

May/June 2015.P23

1 Horses are entered for a horse race. A horse may have to carry a penalty weight in addition to the rider. This weight is added to the saddle. The penalty weight (if any) depends on the number of wins the horse has achieved in previous races.

The penalty weight is calculated as follows:

Number of previous wins	Penalty weight (kg)
0	0
1 or 2	4
Over 2	8

A program is to be written from the following structured English design.

```
1 INPUT name of horse

2 INPUT number of previous wins

3 CALCULATE penalty weight

4 STORE penalty weight

5 OUTPUT name of horse, penalty weight
```

(a) Complete the identifier table showing the variables needed to code the program.

Identifier	Data type	Description

[3]

5 A firm employs workers who assemble amplifiers. Each member of staff works an agreed number of hours each day.

The firm records the number of completed amplifiers made by each employee each day.

Management monitor the performance of all its workers.

Production data was collected for 3 workers over 4 days.

	Daily hours worked
Worker 1	5
Worker 2	10
Worker 3	10

Production data

	Worker 1	Worker 2	Worker 3
Day 1	10	20	9
Day 2	11	16	11
Day 3	10	24	13
Day 4	14	20	17

A program is to be written to process the production data.

(c) An experienced programmer suggests that the pseudocode would be best implemented as a procedure `AnalyseProductionData`.

Assume that both arrays, `DailyHoursWorked` and `ProductionData`, are available to the procedure from the main program and they are of the appropriate size.

```
PROCEDURE AnalyseProductionData(NumDays : INTEGER, NumWorkers : INTEGER)

DECLARE ..........................................................................

DECLARE ..........................................................................

DECLARE ..........................................................................

DECLARE ..........................................................................
```

2.3.1 Programming Basics

```
FOR WorkerNum ← 1 TO 3
    WorkerTotal[WorkerNum] ← 0
ENDFOR

FOR WorkerNum ← 1 TO 3
   FOR DayNum ← 1 TO 4
     WorkerTotal[WorkerNum] ← WorkerTotal[WorkerNum] +
                                         ProductionData[DayNum, WorkerNum]
   ENDFOR
ENDFOR

FOR WorkerNum ← 1 TO 3
   WorkerAverage ← WorkerTotal[WorkerNum]/ (4 * DailyHoursWorked [WorkerNum])
   IF WorkerAverage < 2
      THEN
        OUTPUT "Investigate", WorkerNum
   ENDIF
ENDFOR
ENDPROCEDURE
```

(i) Complete the declaration statements showing the local variables. **[4]**

2.3.1 Programming Basics

Oct/Nov 2015.P21/P23

1 Computer programs have to evaluate expressions.
Study the sequence of pseudocode statements.
Write down the value assigned to each variable.

```
DECLARE h, w, r, Perimeter, Area : REAL
DECLARE A, B, C, D, E            : BOOLEAN

h ← 13.6
w ← 6.4
Perimeter ← (h + w) * 2

r ← 10
Area 3.142 * r^2

Z ← 11 + r / 5 + 3

A ← NOT(r > 10)
```

(i) Perimeter .... **[1]**
(ii) Area .... **[1]**
(iii) Z .... **[1]**
(iv) A .... **[1]**

6 A firm employs five staff who take part in a training programme. Each member of staff must complete a set of twelve tasks which can be taken in any order. When a member of staff successfully completes a task, this is recorded.
A program is to be produced to record the completion of tasks for the five members of staff.
To test the code, the programmer makes the program generate test data.
The program generates pairs of random numbers:

- the first, in the range, 1 to 5 to represent the member of staff
- the second, in the range, 1 to 12 to represent the task

Each pair of numbers simulates the completion of one task by one member of staff.

(a) Explain why the generation of 60 (5 staff x 12 tasks) pairs of random numbers will not simulate all tasks completed by all staff. **[2]**

(b) Data is currently recorded manually as shown.

Staff number	Task number 1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3				✓								
4												
5								✓				

2.3.1 Programming Basics

The table shows that two members of staff have each successfully completed one task.
The program must use a suitable data structure to store, for all staff:

- tasks successfully completed
- tasks not yet successfully completed

The program will output the staff number and task number in the order in which tasks are completed.

The program design in pseudocode is produced as follows:

```
01 DECLARE StaffNum : INTEGER
02 DECLARE TaskNum : INTEGER
03 DECLARE ..................................................................................................
04 DECLARE NewStaffTask : BOOLEAN
05
06 CALL InitialiseTaskGrid
07 Completed ← 0
08 WHILE Completed <> 60
09 NewStaffTask ← FALSE
10 WHILE NewStaffTask = FALSE
11 StaffNum ← RANDOM(1,5) //generates a random number
12 TaskNum ← RANDOM(1,12) //in the given range
13 IF TaskGrid[StaffNum, TaskNum] = FALSE
14 THEN
15 TaskGrid[StaffNum, TaskNum] ← TRUE
16 NewStaffTask ← TRUE
17 OUTPUT StaffNum, TaskNum
18 ENDIF
19 ENDWHILE
20 Completed ← Completed + 1
21 ENDWHILE
22 OUTPUT "Staff Task Count", Completed
23
24 // end of main program
25
26 PROCEDURE InitialiseTaskGrid()
27 DECLARE i : INTEGER
28 DECLARE j : INTEGER
29 FOR i ← 1 TO 5
30 FOR j ← 1 TO 12
31 TaskGrid[i, j] ← FALSE
32 ENDFOR
33 ENDFOR
34 ENDPROCEDURE
```

2.3.1 Programming Basics

Study the pseudocode and answer the questions below.
Give the line number for:

(i) The declaration of a BOOLEAN global variable. **[1]**
(ii) The declaration of a local variable. **[1]**
(iii) The incrementing of a variable used as a counter, but not to control a 'count controlled'loop. **[1]**

(c) (iv) Give the global variable that needs to be declared at line 03. **[2]**

Oct/Nov 2015.P22

1 Computer programs have to evaluate expressions.
Study the sequence of pseudocode statements.
Give the value assigned to each variable.
The statement may generate an error. If so, write ERROR.
The & operator is used to concatenate strings.

```
DECLARE N1        : INTEGER
DECLARE N2        : INTEGER
DECLARE Answer    : REAL
DECLARE Found     : BOOLEAN
DECLARE IsValid   : BOOLEAN
N1 ← 3
N2 ← 9
Answer ← (N1 + N2) / 6
Answer ← 3 * (N1 - 2) + N2 / 2
IsValid ← (N1 > N2) AND (N2 = 9)
Found ← FALSE
IsValid ← (N1 > N2 / 2) OR (Found = FALSE)
Answer ← "1034" & " + " & "65"
```

(i) Answer .... **[1]**
(ii) Answer .... **[1]**
(iii) IsValid .... **[1]**
(iv) IsValid .... **[1]**
(v) Answer .... **[1]**

4 The standard pack of playing cards has four suits – called Clubs, Diamonds, Hearts and Spades.
Each card has a value shown by its number or a name: 1 (Ace), 2, 3, … 10, 11 (Jack), 12 (Queen),13 (King). The pack of cards has one combination for each suit and value.
A program is to be written which simulates a magician dealing all 52 cards from the card pack.
The program generates pairs of random numbers:

- the first, in the range 1 to 4, to represent the suit
- the second, in the range 1 to 13, to represent the card value

(a) Explain why the generation of 52 (4 suits x 13 card values) pairs of random numbers will not simulate the dealing of the complete pack. **[2]**

2.3.1 Programming Basics

(b) A representation of dealing out the cards is shown below:

Suit number	Card value												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1 (Clubs)	F	F	F	F	F	F	F	F	F	F	T	F	F
2 (Diamonds)	F	F	F	F	F	F	F	F	F	F	F	F	F
3 (Hearts)	F	F	T	F	F	F	F	F	F	F	F	F	F
4 (Spades)	F	F	F	F	F	F	F	F	F	F	F	F	F

The table shows two cards have been dealt so far; the 3 of Hearts and the Jack of Clubs.
When each card is dealt, the appropriate cell changes from F to T.
The program will output the suit and the card value in the order in which the cards are dealt.

The program design in pseudocode is produced as follows:

```
01 DECLARE SuitNum : INTEGER
02 DECLARE CardValue : INTEGER
03 DECLARE DealCount : INTEGER
04 DECLARE NewCard : BOOLEAN
05 DECLARE CardPack ..................................................................
06
07 CALL InitialiseCardPack
08 DealCount ← 0
09 WHILE DealCount <> 52
10    NewCard ← FALSE
11    WHILE NewCard = FALSE
12       SuitNum ← RANDOM(1,4) // generates a random number
13       CardValue ← RANDOM(1,13) // in the range given
14       IF CardPack[SuitNum, CardValue] = FALSE
15         THEN
16           CardPack[SuitNum, CardValue] ← TRUE
17           NewCard ← TRUE
18           OUTPUT SuitNum, CardValue
19       ENDIF
20    ENDWHILE
21    DealCount ← DealCount + 1
22 ENDWHILE
23
24 // end of main program
25
```

```
26 PROCEDURE InitialiseCardPack
27    DECLARE i : INTEGER
28    DECLARE j : INTEGER
29    FOR i ← 1 TO 4
30      FOR j ← 1 TO 13
31         CardPack[i, j] ← FALSE
32      ENDFOR
33    ENDFOR
34 ENDPROCEDURE
```

Study the pseudocode and answer the questions below:
Give the line number for:

(ii) The declaration of a local variable. **[1]**
(iii) The initialisation of a variable used as a counter, but not to control a 'count controlled'loop. **[1]**
(f) Complete the declaration of the global variable at line 05.

`05 DECLARE CardPack` .... **[1]**

5 A program is to process a set of integers.
The integers are stored in an array, `Num`. The first `N` elements are to be processed.
The pseudocode for this program is shown below:

```
FOR i ← 1 TO (N – 1)
  j ← 1
  REPEAT
    IF Num[j] > Num[j + 1]
      THEN
        Temp ← Num[j]
        Num[j] ← Num[j + 1]
        Num[j + 1] ← Temp
    ENDIF
    j ← j + 1
  UNTIL j = (N – i + 1)
ENDFOR
```

(b) Complete the identifier table documenting the use of each of the variables.

Identifier	Data type	Description
`Num`	`ARRAY[1:100] OF INTEGER`	The array of numbers.
`N`		
`i`		
`j`		
`Temp`		

[5]

2.3.1 Programming Basics

6 Some pseudocode statements follow which use the following built-in functions:

`ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR`
returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.
For example: `ONECHAR("Barcelona", 3)` returns 'r'.

`CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER`
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("South Africa")` returns 12.

(b) A program is to be written as follows:

- the user enters a string
- the program will form a new string with all <Space> characters removed
- the new string is output

```
NewString ← " "
INPUT InputString

j ← CHARACTERCOUNT(InputString)

FOR i ← 1 TO j

   NextChar ← ONECHAR(InputString, i)
   IF NextChar <> " "
      THEN
        // the & character joins together two strings

        NewString ← NewString & NextChar
   ENDIF
ENDFOR
OUTPUT NewString
```

(i) Complete the identifier table below.

Identifier	Data type	Description
`InputString`	`STRING`	The string value input by the user

[4]

2.3.1 Programming Basics

Oct/Nov 2016. P21/P23

5 Study the following pseudocode statements.

```
CONST Pi = 3.1 : REAL
DECLARE Triangle, Base, Height, Radius, Cone : REAL
DECLARE a, b, c, Answer2 : INTEGER
DECLARE Answer1 : BOOLEAN
Base ←2.6
Height ←10
Triangle ←(Base * Height) / 2
Radius ← 1
Height ← 2
Cone ← 2 * Pi * Radius * (Radius + Height)
a ← 13
b ← 7
c ← 3
Answer1 ←  NOT((a + b + c) > 28)
Total ←  34
Total ← Total - 2
Answer2 ← a + c * c
```

Give the final value assigned to each variable.

(i) `Triangle` .... **[1]**
(ii) `Cone` .... **[1]**
(iii) `Answer1` .... **[1]**
(iv) `Total` .... **[1]**
(v) `Answer2` .... **[1]**

Oct/Nov 2016. P22

2 You will need to refer to the list of pseudocode string-handling functions in the **Appendix**.
A computer program is to simulate the reading and processing of a string of characters from an input device.
The character string consists of:

- a number of digit characters
- one or more <*> characters, each used as a separator
- a final <#> character.

A typical input character sequence, stored as `InputString` is:

`13*156*9*86*1463*18*#`

Study this pseudocode.

```
01 DECLARE Numbers ARRAY [1:100] OF INTEGER
02 DECLARE InputString : STRING
03 DECLARE NextChar : CHAR
04 DECLARE NextNumberString : STRING
05 DECLARE i : INTEGER // Numbers array index
06 DECLARE j : INTEGER // InputString index
07
08 OUTPUT "String ... "
09 INPUT InputString
10 j ←1
```

2.3.1 Programming Basics

```
11 NextChar ← ONECHAR(InputString, j)
12
13 i ← 1
14 WHILE NextChar <> '#'
15   NextNumberString = ""
16         WHILE NextChar <> '*'
17               NextNumberString ← NextNumberString & NextChar
18               j ← j + 1
19               NextChar ← ONECHAR(InputString, j)
20         ENDWHILE
21
22         // store the next integer to the array
23         Numbers[i] ← TONUM(NextNumberString)
24         i ← i + 1
25         j ← j + 1
26         NextChar ← ONECHAR(InputString, j)
27 ENDWHILE
28
29 CALL DisplayArray()
```

(b) Write the line number for:

(i) A statement which declares a global variable used to store a single character. .... **[1]**

(ii) A statement which runs code written as a procedure. .... **[1]**

(iii) A statement which indicates the start of a 'pre-condition' loop. .... **[1]**

(iv) A statement which increments a variable. .... **[1]**

(c) Copy the condition which is used to control the inner loop. **[1]**

3 Radhika mostly studied the high-level programming language XYZ at university.
She has been working in her first job for two years using language XYZ.
She applied for a new job which stated:
"The majority of the development work is done using language ABC."

(a) Radhika was interviewed for the job. Part of the interview process was to study some program code written in language ABC.

```
11 settype($TimesTable, Integer);
12 settype($upTo, Integer);
13 settype($Posn, Integer);
14 settype($Product, Integer);
15 $TimesTable = 7;
16 $UpTo = 10;
17
18 $Posn = 1
19 While ($Posn < $UpTo + 1)
20 {
21 $Product = $Posn * $TimesTable;
22 Echo $Posn . ' X' . $TimesTable . ' = ' . $Product . "<br>";
23 $Posn = $Posn + 1;
24 }
```

Answer the following questions taken from the interview.

(i) State what the `settype` keyword does in this language. **[1]**

2.3.1 Programming Basics

(ii) Name **one** variable that the code uses. **[1]**

(iii) Give a line number for an assignment statement. **[1]**

(iv) Line `19` is the start of a pre-condition loop.
State the syntax that language ABC uses to indicate which statements must be executed inside a loop. **[1]**

6 Study the sequence of pseudocode statements.

```
CONST a = 3.2 : REAL
DECLARE x, y, z, Answer1, Answer2, Answer3 : REAL
DECLARE p, q : BOOLEAN
x ← 3
x ← x + 7
y ← 6
Answer1 ← 2 * (a + y)
z ← 6
Answer2 ← y ^ 2 + 5
p ← TRUE
q ← NOT(NOT(p))
Answer3 ← y + a * 2
```

Give the final value assigned to each variable.

(i) `x` .... **[1]**

(ii) `Answer1` .... **[1]**

(iii) `Answer2` .... **[1]**

(iv) `q` .... **[1]**

(v) `Answer3` .... **[1]**