0478

COMPUTER SCIENCE

TOPIC QUESTION PAPER-2 SET-3

SMART EXAM RESOURCES

Unit 7 Algorithm Design and Problem Solving

1 (a) Draw the most appropriate flowchart symbol for each pseudocode statement.

Pseudocode statement	Flowchart symbol
`IF Number = 20`	
`PRINT Number`	
`Number ← Number + 1`	

[3]

(b) State the type of each pseudocode statement. For example, `X ← X + Y` is totalling.

`IF Number = 20` .....

`PRINT Number` .....

`Number ← Number + 1` .....

[3]

2. This algorithm checks passwords.

- Each password must be 8 or more characters in length; the predefined function `Length` returns the number of characters.
- Each password is entered twice, and the two entries must match.
- Either `Accept` or `Reject` is output.
- An input of 999 stops the process.

```
REPEAT
  OUTPUT "Please enter password"
  INPUT Password
  IF Length(Password) >= 8
    THEN
      INPUT PasswordRepeat
      IF Password <> PasswordRepeat
        THEN
          OUTPUT "Reject"
        ELSE
          OUTPUT "Accept"
      ENDIF
    ELSE
      OUTPUT "Reject"
  ENDIF
UNTIL Password = 999
```

(a) Complete the trace table for the algorithm using this input data:
Secret, Secret, VerySecret, VerySecret, Pa55word, Pa55word, 999, 888

Password	PasswordRepeat	OUTPUT

[3]

(b) Explain how the algorithm could be extended to allow three attempts at inputting the matching password. Any pseudocode statements used in your answer must be fully explained.

.....

.....

.....

.....

.....

.....

.....

..... [4]

3. Identify a suitable validation check that could be used for each piece of normal test data and describe how it would be used. Each validation check must be different.

Test data for entering an email address: **id27@cambridgeuniversity.com**

Validation check name.....

Description of use.....

.....

.....

Test data for entering a year: **2021**

Validation check name.....

Description of use.....

.....

.....

Test data for entering a name: **Ericson-Bower**

Validation check name.....

Description of use.....

.....

.....[6]

4. The pseudocode algorithm should allow a user to input the number of scores to be entered and then enter the scores. The scores are totalled, the total is output and the option to enter another set of scores is offered.

```
1   Count ← 0
2   REPEAT
3     FullScore ← 20
4     INPUT Number
5     FOR StoreLoop ← 1 TO Number
6       INPUT Score
7       FullScore ← FullScore
8     UNTIL StoreLoop = Number
9     OUTPUT "The full score is ", FullScore
10    OUTPUT "Another set of scores (Y or N)?"
11    OUTPUT Another
12    IF Another = "N"
13      THEN
14        Count ← 1
15    ENDIF
16  UNTIL Count = 1
```

(a) Identify the **four** errors in the pseudocode and suggest a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....[4]

(b) Show how you could change the algorithm to store the individual scores in the array ScoreArray[ ], then find and print the average score once the scores have all been entered.

.....

.....

.....

.....

.....

.....

.....

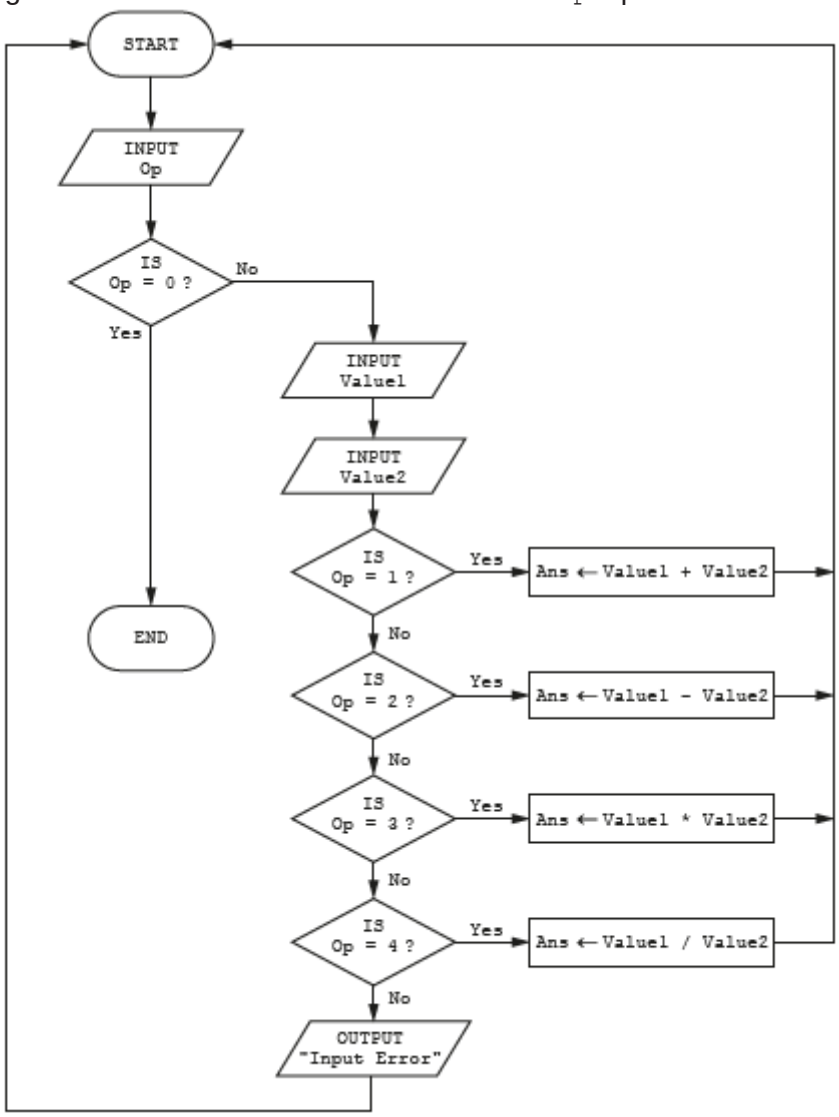
.....

.....

.....[4]

5. The flowchart represents an algorithm.

The algorithm will terminate if 0 is entered at the `Op` input.



(a) Complete the trace table for the algorithm using this input data:
1, 87, 14, 3, 2, 30, 5, 10, 6, 4, 10, 2, 0, 2, 90, 6

Op	Value1	Value2	Ans	OUTPUT

[5]

(b) State the purpose of the algorithm.

.....

.....

.....

..... [1]

(c) Suggest an addition that could be made to the algorithm to make it more useful.

.....

.....

.....[1]

6. An algorithm has been written in pseudocode to generate 50 positive random integers with values less than or equal to 100. These random integers are stored in the array RandNum[ ]

The function `Rand(X, Y)` generates a random integer greater than or equal to `X` and less than `Y`. For example, `Rand(1, 4)` generates `1` or `2` or `3`.

```
1 Count ← 0
2 REPEAT
3     RandNum[Counter] ← Rand(1, 100)
4     Count ← Count + 2
5 UNTIL Count <= 50
```

(a) Find the **four** errors in the pseudocode and write a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

.....[4]

(b) The pseudocode for this algorithm could be shortened by the use of a FOR … NEXT loop.

Rewrite the algorithm using a `FOR … NEXT` loop.

.....

.....

.....

..... [3]

(c) Identify another loop structure available in pseudocode.

..... [1]