**Q1.**

The algorithm, represented using pseudo-code below, outputs a series of integers or the message No result. The output depends upon the value entered by the user.

OUTPUT "Enter an integer greater than 1: "

INPUT X

Product ← 1

Factor ← 0

WHILE Product < X

  Factor ← Factor + 1

  Product ← Product \* Factor

ENDWHILE

IF X = Product THEN

  Product ← 1

  FOR N ← 1 TO Factor

    Product ← Product \* N

    OUTPUT N

  ENDFOR

ELSE

  OUTPUT "No result"

ENDIF

**What you need to do:**

**Task 1**

Write a program to implement the algorithm.

**Task 2**

Test that your program works:

•   run your program, then enter the number 720

•   run your program, then enter the number 600

**Evidence that you need to provide**

(a)  Your PROGRAM SOURCE CODE for **Task 1**.

**(9)**

(b)  SCREEN CAPTURE(S) showing the tests described in **Task 2**.

**(1)**

(c)  What is true for all valid inputs for X that output a number of numbers which is not true for all other valid inputs that output No result?

**(1)**

**(Total 11 marks)**

**Q2.**

The algorithm below, represented using pseudo-code, outputs a numeric result.

The numeric result depends upon the value entered by the user.

OUTPUT "Enter a positive whole number: "

INPUT NumberIn

NumberOut ← 0

Count ← 0

WHILE NumberIn > 0

  Count ← Count + 1

  PartValue ← NumberIn MOD 2

  NumberIn ← NumberIn DIV 2

  FOR i ← 1 TO Count – 1

    PartValue ← PartValue \* 10

  ENDFOR

  NumberOut ← NumberOut + PartValue

ENDWHILE

OUTPUT "The result is: " NumberOut

The table below lists the MOD and DIV operators for each of the available programming languages. You should refer to the row for your programming language.

|  |  |  |
| --- | --- | --- |
| **Programming language** | **MOD** | **DIV** |
| C# | % | / |
| Java | % | / |
| Pascal | mod | div |
| Python | % | // |
| VB.Net | Mod | \ |

**What you need to do:**

**Task 1**

Write a program to implement the algorithm above.

**Task 2**

Test that your program works:

•   run your program, then enter the number 22

•   run your program, then enter the number 29

•   run your program, then enter the number -1

**Evidence that you need to provide**

(a)  Your PROGRAM SOURCE CODE for **Task 1**.

**(11)**

(b)  SCREEN CAPTURE(S) showing the test described in **Task 2**.

**(1)**

(c)  What is the purpose of this algorithm?

**(1)**

**(Total 13 marks)**

**Q3.**

The algorithm represented using pseudo-code in **Figure 1** describes a method to find the greatest common factor (GCF) of two whole numbers (integers) entered by the user.

**Figure 1**

OUTPUT "Enter a whole number: "

INPUT Number1

OUTPUT "Enter another whole number: "

INPUT Number2

Temp1 ← Number1

Temp2 ← Number2

WHILE Temp1 ≠ Temp2

  IF Temp1 > Temp2 THEN

    Temp1 ← Temp1 – Temp2

  ELSE

    Temp2 ← Temp2 – Temp1

  ENDIF

ENDWHILE

Result ← Temp1

OUTPUT Result, " is GCF of ", Number1, " and ", Number2

**What you need to do:**

**Task 1**

Write a program to implement the algorithm in **Figure 1**.

**Task 2**

Test the program by showing the result of entering 12 and then 39.

**Evidence that you need to provide**

(a)  Your PROGRAM SOURCE CODE for **Task 1**.

**(6)**

(b)  SCREEN CAPTURE(S) showing the test described in **Task 2**.

**(1)**

The algorithm copies the values of Number1 and Number2 into Temp1 and Temp2 respectively.

(c)  Explain why the WHILE loop was written using Temp1 and Temp2 instead of Number1 and Number2.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 8 marks)**

**Q4.**

One method that can be used to compress text data is run length encoding (RLE). When RLE is used the compressed data can be represented as a set of character/frequency pairs. When the same character appears in consecutive locations in the original text it is replaced in the compressed text by a single instance of the character followed by a number indicating the number of consecutive instances of that character. Single instances of a character are represented by the character followed by the number 1.

**Figure 1** and **Figure 2** show examples of how text would be compressed using this method.

**Figure 1**

Original text: AAARRRRGGGHH

Compressed text: A 3 R 4 G 3 H 2

**Figure 2**

Original text: CUTLASSES

Compressed text: C 1 U 1 T 1 L 1 A 1 S 2 E 1 S 1

**What you need to do**

**Task 1**

Write a program that will perform the compression process described above. The program should display a suitable prompt asking the user to input the text to compress and then output the compressed text.

**Task 2**

Test the program works by entering the text AAARRRRGGGHH.

**Task 3**

Test the program works by entering the text A.

**Evidence that you need to provide**

(a)     Your PROGRAM SOURCE CODE.

**(12)**

(b)     SCREEN CAPTURE(S) for the test showing the output of the program when AAARRRRGGGHH is entered.

**(1)**

(c)     SCREEN CAPTURE(S) for the test showing the output of the program when A is entered.

**(1)**

**(Total 14 marks)**

**Q5.**

The algorithm, represented using pseudo-code in the figure below, describes a method to calculate the additive or multiplicative persistence of a two-digit integer. The examples below illustrate how additive and multiplicative persistence are calculated.

**Example**: calculating the additive persistence of 87

8 + 7 = 15

1 + 5 = 6

After 2 steps the method results in a one digit answer so the additive persistence of 87 is 2.

**Example**: calculating the multiplicative persistence of 39

3 \* 9 = 27

2 \* 7 = 14

1 \* 4 = 4

After 3 steps the method results in a one digit answer so the multiplicative persistence of 39 is 3.

OUTPUT "Enter integer (0-99):"

INPUT Value

OUTPUT "Calculate additive or multiplicative persistence (a or m)?"

INPUT Operation

Count ← 0

WHITE Value > 9

    IF Operation = "a" THEN

        Value ← (Value DIV 10) + (Value MOD 10)

    ELSE

        Value ← (Value DIV 10) \* (Value MOD 10)

    ENDIF

    Count ← Count + 1

ENDWHILE

OUTPUT "The persistence is: "

OUTPUT Count

The MOD operator calculates the remainder resulting from an integer division, for example, 10 MOD 3 = 1.

The DIV operator calculates integer division, for example 10 DIV 3 = 3.

**What you need to do**

**Task 1**

Write a program for the algorithm in the figure.

**Task 2**

Test the program by showing the result of entering 47, followed by m when prompted by the program.

**Task 3**

Test the program by showing the result of entering 77, followed by a when prompted by the program.

**Evidence that you need to provide**

(a)     Your PROGRAM SOURCE CODE

**(8)**

(b)     SCREEN CAPTURE(S) showing the tests described.

**(1)**

The part of the program where the calculations are performed uses a WHILE repetition structure.

(c)     Explain why a WHILE repetition structure was chosen instead of a FOR repetition structure.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

**(Total 10 marks)**

**Q6.**

**Figure 1** contains the pseudo-code for a program to output a sequence according to the ‘Fizz Buzz’ counting game.

**Figure 1**

OUTPUT "How far to count?"  
INPUT HowFar  
WHILE HowFar < 1  
  OUTPUT "Not a valid number, please try again."  
  INPUT HowFar  
ENDWHILE

FOR MyLoop ← 1 TO HowFar  
  IF MyLoop MOD 3 = 0 AND MyLoop MOD 5 = 0  
  THEN  
      OUTPUT "FizzBuzz"  
    ELSE  
      IF MyLoop MOD 3 = 0  
        THEN  
          OUTPUT "Fizz"  
        ELSE  
          IF MyLoop MOD 5 = 0  
            THEN  
              OUTPUT "Buzz"  
            ELSE  
              OUTPUT MyLoop  
          ENDIF  
      ENDIF  
  ENDIF  
ENDFOR

**What you need to do:**

Write a program that implements the pseudo-code as shown in **Figure 1**.

Test the program by showing the result of entering a value of 18 when prompted by the program.

Test the program by showing the result of entering a value of -1 when prompted by the program.

**Evidence that you need to provide**

(a)     Your PROGRAM SOURCE CODE for the pseudo-code in **Figure 1**.

**(8)**

(b)     SCREEN CAPTURE(S) for the tests conducted when a value of 18 is entered by the user and when a value of -1 is entered by the user.

**(1)**

The main part of the program uses a FOR repetition structure.

(c)     Explain why a FOR repetition structure was chosen instead of a WHILE repetition structure.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(1)**

(d)     Even though a check has been performed to make sure that the variable HowFar is greater than 1 there could be inputs that might cause the program to terminate unexpectedly (crash).

Provide an example of an input that might cause the program to terminate and describe a method that could be used to prevent this.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(3)**

(e)     Programs written in a high level language are easier to understand and maintain than programs written in a low level language.

The use of meaningful identifier names is one way in which high level languages can be made easier to understand.

State **three** other features of high level languages that can make high level language programs easier to understand.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

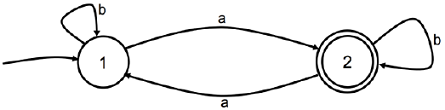
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(3)**

(f)      The finite state machine (FSM) shown in **Figure 2** recognises a language with an alphabet of a and b.

**Figure 2**



Input strings of a and aabba would be accepted by this FSM.

In the table below indicate whether each input string would be accepted or not accepted by the FSM in **Figure 2**.

If an input string would be accepted write YES.  
If an input string would **not** be accepted write NO.

|  |  |
| --- | --- |
| **Input string** | **Accepted by FSM?** |
| aaab |  |
| abbab |  |
| bbbbba |  |

**(2)**

(g)     In words, describe the language (set of strings) that would be accepted by this FSM shown in **Figure 2**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(2)**

**(Total 20 marks)**