**Q1.**

The contents of the arrays Items and NewItems are shown in **Figure 1**.

A pseudo-code representation of an algorithm is given in **Figure 2**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Figure 1** | | | |
| Items | | | |
| [0] | [1] | [2] | [3] |
| 12 | 25 | 12 | 53 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewItems | | | |
| [0] | [1] | [2] | [3] |
| 0 | 0 | 0 | 0 |

**Figure 2**

ItemsCount ← 4

NewItems[0] ← Items[0]

NewItemsCount ← 1

FOR LoopA ← 1 TO ItemsCount - 1

    Done ← False

    For LoopB ← 0 TO NewItemsCount - 1

        IF Items[LoopA] = NewItems[LoopB] THEN

            Done ← True

        ENDIF

    ENDFOR

    IF Done = False THEN

        NewItems[NewItemsCount] ← Items[LoopA]

        NewItemsCount ← NewItemsCount + 1

    ENDIF

ENDFOR

(a)     Dry run the algorithm in **Figure 2** by completing the table. The first row has been completed for you. You may not need to use all of the rows provided in the table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ItemsCount | NewItemsCount | LoopA | Done | LoopB | NewItems | | | |
| [0] | [1] | [2] | [3] |
| 4 | 1 |  |  |  | 12 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**(5)**

(b)     Explain the purpose of the algorithm in **Figure 2**.

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

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

**(1)**

**(Total 6 marks)**

**Q2.**

The following algorithm uses an array Values that contains the integers 4,7,9.

(a)     Dry run this algorithm by using the trace table below.

Last ← 3  
New ← 6  
Ptr ← 1  
WHILE(New > Values[Ptr])  
  Ptr ← Ptr + 1  
ENDWHILE  
WHILE (Last >= Ptr)  
  Values[Last+1] ← Values[Last]  
  Last ← Last – 1  
ENDWHILE  
Values[Ptr] ← New

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| New | Last | Ptr | Values | | | | |
|  |  |  | [1] | [2] | [3] | [4] | [5] |
| 6 | 3 | 1 | 4 | 7 | 9 |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**(6)**

(b)     What is the purpose of this algorithm?

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

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

**(1)**

**(Total 7 marks)**

**Q3.**

The following section of pseudo-code processes a one-dimensional integer array called *List.* The numbers in *List* are stored in ascending order, and x, *Low, High, Middle* are all integer variables. (The function Int returns the whole number part of its parameter.)

Proc Process(Low, High, x)

     Found ← False  
Repeat  
     Middle ← Int((Low **+** High)/2)  
     If List(Middle) = x  
     Then Found ← True  
     Else If List(Middle) **>** x  
        Then High ← Middle –1  
        Else Low ← Middle +1 {List(Middle) <x}  
Until Found = True

(a)     Complete the following dry-run table for Process (1, 10, 19), given that the integers in the list are:

2,4, 6, 7, 11, 13, 19, 21, 27, 29

|  |  |  |  |
| --- | --- | --- | --- |
| **Low** | **High** | **Middle** | **Found** |
| 1 | 10 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**(7)**

(b)     What type of routine does this pseudo-code define?

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

**(1)**

**(Total 8 marks)**

**[8]**