Algorithms on 1D Array

Algorithm_Traverse_Array()

Input: The elements of an array, say A.
Output: The elements of A are traversed in order.
Data Structure Used: An array A[L..U], where L = Lower Index and U = Upper Index of the array.
Steps: Begin

```
Set i = L
While (L <= U)
Begin
        Process A[i]
        Set i = i + 1
End While
```

Algorithm_Find_Min_Array()

Input: The elements of a 1D array, say A.
Output: The minimum of the elements of A, say MIN.
Data Structure used: An array A[L .. U], where L = Lower Index and U = Upper Index of the array.
Steps: Begin

```
Set MIN = A[L]
Set i = A[L + 1]
While (i <= U)
Begin
        If (MIN > A[i])
        Then
             Set MIN = A[i]
        End If
        Set i = i + 1
End While
End
```

Algorithm_Find_Max_Array()

Input: The elements of a 1D array, say A.
Output: The maximum of the elements of A, say MAX.
Data Structure used: An array A[L .. U], where L = Lower Index and U = Upper Index of the array.

Steps: Begin

```
Set MAX = A[L]
Set i = A[L + 1]
While (i <= U)
Begin
        If (MAX > A[i])
        Then
              Set MAX = A[i]
        End If
        Set i = i + 1
End While
End
```

Algorithm_Insert_Array()

Input: The elements of an array, say A and an item, say ITEM to insert in A at a position, say POS.

Output: ITEM inserted successfully or suitable unsuccessful message.

Data Structure Used: An array A[L..U], where L = Lower Index and U = Upper Index of the array A with SIZE = U – L + 1.

Steps:

```
Begin
If No_of_elements = SIZE
Then
      Print “Array is full!!”
Else
      If (POS < 1 || POS > SIZE)
      Then
            Print “Invalid Location specified!!”
      Else
            Set i = No_of_elements – L + 1
            While (i >= (POS – L + 1))
            Begin
                  A[i+1] = A[i]
                  Set i = i – 1
            End While
            Set A[i] = ITEM
      End If
End If
End
```

Algorithm_Delete_Array()

Input: The elements of an array, say A and a position, say POS from where item has to be deleted.

Output: An Item, say ITEM deleted successfully from POS or suitable unsuccessful message.

Data Structure Used: An array A[L..U], where L = Lower Index and U = Upper Index of the array A with SIZE = U – L + 1.

```
Steps:    Begin
          If (No_of_elements = 0), Then
               Print “Array is empty”
          Else
               If (POS < 1 || POS > SIZE)
               Then
                    Print "Item can’t be deleted"
               Else
                    Set i = POS – L
                    While (i < No_of_elements)
                    Begin
                         Set A[i-1] = A[i]
                         Set i = i + 1
                    End While
               End If
          End If
          End
```

Algorithm_Linear_Search_Array()

Input: The elements of an array, say A and an Item, say ITEM to search in A.

Output: ITEM inserted successfully or suitable unsuccessful message.

Data Structure Used: An array A[L..U], where L = Lower Index and U = Upper Index of the array A with SIZE = U – L + 1.

```
Steps:    Begin
          Set flag = 0
          Set i = L
          While (i <= U)
          Begin
               If (A[i] = ITEM)
               Then
                    Set flag = 1
                    Set POS = i – L + 1
                    Break
               End If
               Set i = i + 1
          End While
          If flag = 0
```

```
Then
      Print “ITEM not found”
Else
      Print “ITEM found at POS”
End If
End
```

Algorithm_Binary_Search_Array()

Input: The elements of an array, say A available in sorted order and an Item, say ITEM to search in A.

Output: ITEM inserted successfully or suitable unsuccessful message.

Data Structure Used: An array A[L..U], where L = Lower Index and U = Upper Index of the array A with SIZE = U – L + 1.

Steps:

```
Begin
Set flag = 0
While (i <= U)
Begin
      Set mid = ↳ (L + U)/ 2 ↲
      If (ITEM = A[mid])
      Then
            Set flag = 1
            Set POS = mid – L + 1
            Break
      Else If (ITEM < A[mid])
      Then
            Set U = mid – 1
      Else
            Set L = mid + 1
      End If
End While
If flag = 0
Then
      Print “ITEM not found”
Else
      Print “ITEM found at POS”
End If
End
```

Algorithm_Merging_1D_Arrays()

Input: The elements of two arrays, say A and B.

Output: An array, say C containing the elements of **A** followed by the elements of B.

Data Structure used: An array A[L .. U1], B[L .. U2] and C[L .. U] where L = Lower index of the arrays, U1 = Upper Index of the array A, U2 = Upper Index of the array B and U = Upper index of the array C

Steps:

```
Begin
Set i = L
Set j = L
While (i <= U1)
Begin
     Set C[j] = A[i]
     Set i = i + 1
     Set j = j + 1
End While
Set i = L
While (i <= U2)
Begin
     Set C[j] = B[i]
     Set i = i + 1
     Set j = j + 1
End While
End
```