An ARRAY ADT

```
Function INITIALIZE (A)
 1. Initialize Array, Set A[0] to Delimiter (say MIN_VAL)
        A(0) := MIN_VAL
 2. Terminate
        Return.

Function DISPLAY (A, LEN)
1.  Display the Prompt
        Print(' Array Contents Are ')
2.  Iterate to print array elements
        For  I := 0 to LEN-1
            print( A(I) )
3.  Return to Terminate
        Return
```

```
Function INSERT_ELEMENT (A, INDEX, VALUE)
1.  Parameter Initialization
        LEN := A.Length
2.  Check if Array is Full
        If LEN = MX_SIZE
            Print (' Arry Bounds Reached.. ')
            Return LEN.
3.  Locate the Array Index for Insertion by moving elements
        For I := LEN downto INDEX
            A(I+1) := A(I)
4.  Place the value at desired position
        A(INDEX) := VALUE
5.  Return Array Length
        Return LEN+1
```

```
Procedure CREATE (A, LEN)
    This procedure initializes array with element values. LEN represents array
    length and is passed by reference.
1.  Parameter Initialization
        LEN := A.Length
2.  Iterate to receive input and insert into array
        Repeat Step 2 thru Step 3  Until (LEN = MX_SIZE OR VALUE <> MIN_VAL)
3.  Check if Array is Full, Otherwise Accept and insert element value
        If LEN = MX_SIZE
            Print (' Arry Bounds Reached.. ')
            Break
        Else
            VALUE := Call GET_INPUT("Element Value")
            If VALUE = MIN_VAL [ Discard MIN_VALUE ]
                Continue
            Call INSERT_ELEMENT(A, (LEN), VALUE)
4.  Return Array and Length
        Return.
```

```
Function IS_EMPTY (A)
1.  If the array is Empty, return TRUE
        If A.Length = 0
            Return 1
2.  Otherwise, return FALSE
        Return 0


Function IS_FULL (A)
1.  If the array is Full, return TRUE
        If A.Length = MX_SIZE
            Return 1

2.  Otherwise, return FALSE
        Return 0
```

```
Function LENGTH (A)
1.  Local parameter
        LEN := 0
2.  Iterate Until MX_SIZE
        While LEN < MX_SIZE
        [If Element Value is MIN_VAL, Exit loop]
            If A(LEN) = MIN_VAL
                Break
            LEN := LEN+1
        End-While
3.  Return Length
        Return LEN.
```

```
Function SEARCH( A, VALUE)
1.  Set Length
        LEN := A.Length
2.  Iterate thru Array to locate element
        For I := 0 to LEN-1
        [If located, return index]
            If A(I) = VALUE
                Return I+1
3.  Unsuccessful Search
        Return 0.
```

```
Function DELETE_ELEMENT (A, INDEX)
1.  Set Array Length
        LEN := A.Length
2.  Is Array Empty??
        If LEN = 0
            Print (' Array Empty, Delete Failed ')
            Return LEN
3.  Iterate thru array to remove the element at INDEX position
        For I := INDEX to LEN-1
            A(I) := A (I+1)
4.  Was the array Full, before removal of element? Set Delimiter.
        If LEN = MX_SIZE
            A(LEN-1) := MIN_VAL
5.  Return Updated Length
        Return LEN-1
```

```
Function TRAVERSE( A )
1.  Set Length
        LEN := A.Length
2.  Is Array Empty? Generate Error.
        If LEN = 0
            Print (' Array Empty, Failed.. ')
3.  Otherwise, Execute Operation
        Else
            For I := 0 to LEN-1
                A(I) := A(I) + 200
        End-If
```

```
Function SORT (A, LEN)
1.  Is Array Empty??
        If LEN = 0
            Print (' Empty Array, Sort Failed ..')
            Return
2.  Iterate thru Array to Order It
        For I:= 0 to LEN-2
            For J:= I+1 to LEN-1
                If A(I) > A(J)
                    Call SWAP( A(I), A(J) )
                End-If
            End-For
        End-For
3. Successful Ordering
        Return
```

```
Function COPY (A, B)
1.  Iterare through Array A
        For I := 0 to A.Length-1
            B(I) := A(I)
2.  Set Array Boundary
        If A.Length < MX_SIZE
            B(I) := MIN_VAL
3.  Return Length of Copied Array
        Return A.Length
```