Object Oriented Programming (BCS-2E, BCS-2F) – Assignment 1

**Important Instructions:**

* Submit only one running .cpp file carefully. After deadline no updated files will be accepted.
* Do not submit compressed files (.zip or .rar etc.).
* Name of your submission file should be your roll number.

# Exercise 1 [Bubble Sort]

Write a C++ program that takes size of an array and array’s elements from file “Data1.txt” and displays the sorted list (in ascending order) of elements after applying Bubble Sort.

## Input File

**Important Note:** File name in your code should be “Data1.txt”, it will be evaluated accordingly.

**Data1.txt** (Create file and save following data in it.)

|  |
| --- |
| 6  8 6 11 3 15 5  10  25 83 55 27 1 3 62 49 44 7  -2  33 76 89 |

**Format of data file** is explained below:

|  |
| --- |
| 6 //Size of Array for Test Case 1  8 6 11 3 15 5 // Data for Array 1  10 //Size of Array for Test Case 2  25 83 55 27 1 3 62 49 44 7 //Data for array 2  -2 //Size of Array for Test Case 3  33 76 89 //Data for Array 3 |

There will be total three test cases in input file but the data i.e. size and elements of the array may differ, so your code should be generic.

## Required Output

|  |
| --- |
| Test Case 1:  Array before Sorting: 8, 6, 11, 3, 15, 5  Sorted Array:  3, 5, 6, 8, 11, 15  Test Case 2:  Array before Sorting: 25, 83, 55, 27, 1, 3, 62, 49, 44, 7  Sorted Array:  // Print your sorted array here  Test Case 3:  Error: Array Size should be greater than zero. |

## Bubble Sort Explanation:

A bubble sort starts at the top of the list. Each element is compared to the next. If it is greater than the next element, then swap the two. Pass through the list as many times as necessary to sort it. Usually the number of passes required is equal to (**number of elements – 1**). The smallest value bubbles up to the top of the list while the largest value sinks to the bottom.

**8 6 11 3 15 5**

**swap**

**6 8 11 3 15 5**

**okay**

**6 8 11 3 15 5**

**swap**

**6 8 3 11 15 5**

**okay**

**6 8 3 11 15 5**

**swap**

**6 8 3 11 5 15**

**Pass 1:**

**Pass2:**

**6 8 3 11 5 15**

**okay**

**6 8 3 11 5 15**

**swap**

**6 3 8 11 5 15**

**okay**

**6 3 8 11 5 15**

**swap**

**6 3 8 5 11 15**

**Pass4:**

**3 6 5 8 11 15**

**okay**

**3 6 5 8 11 15**

**swap**

**3 5 6 8 11 15**

**Pass3:**

**swap**

**6 3 8 5 11 15**

**okay**

**3 6 8 5 11 15**

**swap**

**3 6 8 5 11 15**

**3 6 5 8 11 15**

**Scan 5 will not do swapping and the algorithm terminates after this pass.**

**Pass5:**

# Exercise 2 [Merge Arrays]

Write a program that takes two sorted arrays and merges the arrays (in a new third array) in sorted order. You are not allowed to use any sorting algorithm.

## Input File

Data2.txt (Create a file and save following data in it) Name of the file in your program should be “Data2.txt”, it will be evaluated accordingly.

|  |
| --- |
| 3  2 5 9  4  1 2 3 6  4  1 2 3 6  3  2 5 9  5  2 5 9 14 19  2  6 8 |

**Format of data input file** is explained below

|  |
| --- |
| 3 //Size of Array 1 of Test Case 1  2 5 9 //Data of Array 1  4 //Size of Array 2 of Test Case 1  1 2 3 6//Data of Array 2  4//Size of Array 1 of Test Case 2  1 2 3 6//Data of Array 1  3//Size of Array 2 of Test Case 2  2 5 9//Data of Array 2  5//Size of Array 1 of Test Case 3  2 5 9 14 19//Data of Array 1  2//Size of Array 2 of Test Case 3  6 8//Data of Array 2 |

There will be total three test cases but the data i.e. size and elements of the array may differ, so your code should be generic.

## Required Output

|  |
| --- |
| Test Case 1:  Array 1: 2, 5, 9  Array 2: 1, 2, 3, 6  Sorted Merged Array:  1, 2, 2, 3, 5, 6, 9  Test Case 2:  Array 1: 1, 2, 4, 6  Array 2: 2, 5, 8  Sorted Merged Array:  1, 2, 2, 4, 5, 6, 8  Test Case 3:  Array 1: 2, 5, 9, 14, 19  Array 2: 6, 8  Sorted Merged Array:  2, 5, 6, 8, 9, 14, 19 |

**Important Note:**

Violation of following will result in **ZERO** credit:

* **Use pointers to scan/traverse the array. Any loop iterating from index zero to size (i.e. using integer iterators to traverse the array) is not allowed.**
* You cannot use subscript operator [] to manipulate arrays in your program.
* There will be marks deduction for every single exception.
* Using Global Variables and “goto” in assignment is **NOT ALLOWED**.

**Marks Distribution for Exercise 1 (Tentative):**

There will be similar marks distribution for exercise 2.

|  |  |
| --- | --- |
| **Requirements** | **Marks** |
| User Interface (Printing Input and Output) | 2 + 2 |
| Sorted List Output | 12 |
| Program’s behavior on list size <= 0 (Display proper error message) | 2 |
| Dynamic Allocation | 2 |
| Memory Deallocation | 7 |
| Separate Input Function | 5 |
| Separate Output Function | 5 |
| Separate Bubble Sort Function | 2 |
| Coding Standards / Readability: |  |
| Variable Names | 2 |
| Comments | 2 |
| Other Coding Standards | 2 |
| **Total Marks** | **45** |

**Submit only one running .cpp file with names YourRollNumber.cpp, for example, L201234.cpp. Do not submit compressed files.**

**Final Output required on console (Your program should NOT take ANYTHING from user.):**

|  |
| --- |
| ----------------------------------------Bubble Sort Result-----------------------------------------  Test Case 1:  Array before Sorting: 8, 6, 11, 3, 15, 5  Sorted Array:  3, 5, 6, 8, 11, 15  Test Case 2:  Array before Sorting: 25, 83, 55, 27, 1, 3, 62, 49, 44, 7  Sorted Array:  // Print your sorted array here  Test Case 3:  Error: Array Size should be greater than zero.  ----------------------------------------Merge Arrays Result-----------------------------------------  Test Case 1:  Array 1: 2, 5, 9  Array 2: 1, 2, 3, 6  Sorted Merged Array:  1, 2, 2, 3, 5, 6, 9  Test Case 2:  Array 1: 1, 2, 4, 6  Array 2: 2, 5, 8  Sorted Merged Array:  1, 2, 2, 4, 5, 6, 8  Test Case 3:  Array 1: 2, 5, 9, 14, 19  Array 2: 6, 8  Sorted Merged Array:  2, 5, 6, 8, 9, 14, 19 |