Lab Manual – Dynamic 1D Array

Important Note:

- **There shouldn't be any memory leakage or dangling pointers in your program.**
- Make separate functions for input and output of arrays. Your main should be a sequence of function calls only.
- You are not allowed to use global variables and goto instruction.
- **Submit only one cpp file having main function testing all the following functions.**

Exercise 1 [Input Array]:

Write a function **int* InputArray(int& size)** that asks user to enter size of required array, allocates the memory on heap, takes input in array and returns its pointer.

Exercise 2 [Output Array]:

Write a program **void OutputArray(int* myArray, const int& size)** that takes a pointer to an integer array and prints its data.

Write main function to test above functionality.

Exercise 3 [Compress Array]:

Write a function **int* CompressArray(int* oiginalArr, int& size)** that takes a sorted array and removes duplicate elements from this array.

Sample Run:

```
//Input:
Enter Size of array: 10
Enter 10 elements: 1 2 2 2 3 3 3 3 3 7
//Output
Array after Compression: 1 2 3 7
```

Your function will compress the original array, allocate a new array of compressed size (compressed size is 4 in above example) on heap, copy updated array in new array and return the new array.

Take input from user by calling **int* InputArray(int& size)** (function you implemented in Exercise 1). Call CompressArray, call OutputArray(function you implemented in Exercise 2) to display the final output.

Note: Make appropriate functions for following problems yourself

Exercise 4 [Intersection]

Implement a function **int* Intersection(int* setA, int& size1, int* setB, int& size2, int& size3)** that finds intersection (common elements) of two sets (stored using arrays).

Sample Run:

```
//Input:
Enter Size of Array: 6
Enter 6 elements: 1 2 3 4 5 6
```

Enter Size of Array: 4
Enter 3 elements: 1 3 5 7

//Output
A = {1,2,3,4,5,6}
B = {1,3,5,7}
A Intersection B = {1, 3, 5}

Help: Note array3 should not have any duplicate elements. You must:

- Allocate the three arrays dynamically after inputting the size of array1 and array2 from the user. Statically allocated arrays are NOT allowed.
- Initially you can allocate elements = (size of array1 + size of array2) to array3. For example, you would allocate 6+4 to array3 for the above example. After finding the common elements, the allocated size of array3 may be more than what you need. (In the above example you require size 3 whereas you have allocated 10).