National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Programming Fundamentals**	**Course Code:**	**CS1002**
	Program:	**Electrical Engineering**	**Semester:**	**Fall 2023**
	Assigned on:	**22 November 2023**	**Total Marks:**	**50**
	Deadline:	**30 November 2023**	**Weight:**	**3.33**
	Section:	**EE-1A and EE-1C**	**Page(s):**	**2**
	Exam Type:	**Assignment-3 Solution**	**CLO #**	**4**

Instruction:
1. Do not forget to write your Name and Roll Numbers.
2. Submit hand-written hard copy at the Start of the Class on Thursday, 30 November.
3. **No Late submissions**. Plagiarism/copying cases to be referred to the DC.

Question No. 1 **Marks: 10+5**

Task 1: Ask the user to enter two sets (of integers). First ask them to enter the size of each set (i.e. the number of elements in each set), then input these elements. As you know, a set cannot contain duplicate elements. If the user, while entering the numbers, repeats a number, tell them it's already been entered and ask for a different number. By the end of this process, you will have two sets, say s1 and s2, both containing numbers without repetitions. The numbers can be both positive and negative.

You may assume that neither set will contain more than 500 elements. So you can create arrays of capacity 500 each.

Task 2: Print s1 and s2 on the screen in the standard way of writing sets. For example, if s1 contains 1, 5 and 11, and s2 contains -3, 0, 5 and 9, the program should display:

s1 = {1, 5, 11}

s2 = {-3, 0, 5, 9}

For an empty set, print { }

Solution

```
#include <iostream>
#include <unordered_set>

int main() {
   const int maxSize = 500;

   // Task 1: Input two sets
   int sizeS1, sizeS2;

   std::cout << "Enter the size of set s1 (up to 500): ";
   std::cin >> sizeS1;

   std::unordered_set<int> s1;
   int num;
   std::cout << "Enter elements for set s1 (without repetitions):\n";
   for (int i = 0; i < sizeS1; ++i) {
      std::cin >> num;
      if (s1.find(num) != s1.end()) {
         std::cout << "Number already entered. Enter a different number.\n";
```

```
            --i; // Decrement i to re-enter the current element
        } else {
            s1.insert(num);
        }
    }

    std::cout << "Enter the size of set s2 (up to 500): ";
    std::cin >> sizeS2;

    std::unordered_set<int> s2;
    std::cout << "Enter elements for set s2 (without repetitions):\n";
    for (int i = 0; i < sizeS2; ++i) {
        std::cin >> num;
        if (s2.find(num) != s2.end()) {
            std::cout << "Number already entered. Enter a different number.\n";
            --i; // Decrement i to re-enter the current element
        } else {
            s2.insert(num);
        }
    }

    // Task 2: Print sets s1 and s2
    std::cout << "s1 = {";
    for (int num : s1) {
        std::cout << num << ", ";
    }
    std::cout << "}\n";

    std::cout << "s2 = {";
    for (int num : s2) {
        std::cout << num << ", ";
    }
    std::cout << "}\n";

    return 0;
}
```

Question No. 2 **Marks: 10+10**

You learnt and implemented two algorithms in class; **Sequential/Linear Search** and **Selection Sort**.
Write a C++ program that inputs a N size array from user.
Sorts it using **Bubble Sort** and searches a value from it using **Binary Search**.
Hint: For solving this problem, you need to find the Bubble Sort algorithm, understand its pseudo-code and implement it in C++. You need to find the Binary Search algorithm, understand its pseudo-code and implement it in C++.

Solution

```
#include <iostream>

// Function to perform Bubble Sort on an array
void bubbleSort(int arr[], int size) {
   for (int i = 0; i < size - 1; ++i) {
      for (int j = 0; j < size - i - 1; ++j) {
         if (arr[j] > arr[j + 1]) {
            // Swap arr[j] and arr[j+1] if they are in the wrong order
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
         }
      }
   }
}

// Function to perform Binary Search on a sorted array
int binarySearch(int arr[], int size, int target) {
   int left = 0, right = size - 1;

   while (left <= right) {
      int mid = left + (right - left) / 2;

      if (arr[mid] == target) {
         return mid; // Element found, return its index
      } else if (arr[mid] < target) {
         left = mid + 1; // If target is greater, ignore the left half
      } else {
         right = mid - 1; // If target is smaller, ignore the right half
      }
   }

   return -1; // Element not found
}
```

```
int main() {
    int N;

    // Input the size of the array
    std::cout << "Enter the size of the array: ";
    std::cin >> N;

    int arr[N];

    // Input array elements from the user
    std::cout << "Enter " << N << " elements for the array:\n";
    for (int i = 0; i < N; ++i) {
        std::cin >> arr[i];
    }

    // Sort the array using Bubble Sort
    bubbleSort(arr, N);

    // Display the sorted array
    std::cout << "Sorted array: {";
    for (int i = 0; i < N; ++i) {
        std::cout << arr[i] << (i < N - 1 ? ", " : "");
    }
    std::cout << "}\n";

    // Input a value to search
    int searchValue;
    std::cout << "Enter a value to search: ";
    std::cin >> searchValue;

    // Perform Binary Search
    int index = binarySearch(arr, N, searchValue);

    // Display the result of the search
    if (index != -1) {
        std::cout << "Value " << searchValue << " found at index " << index << ".\n";
    } else {
        std::cout << "Value " << searchValue << " not found in the array.\n";
    }

    return 0;
}
```

Question No. 3 **Marks: 5**

Implement a function weightedSum that take an integer array A and size S as input parameters and returns the weighted sum given by: sum = 0*A[0] + 1*A[1] + 2*A[2] … (S-1) * A[S-1] where every element in the array is multiplied with its index number. Sample example for S = 4:

Array:	8	3	1	2
Index:	0	1	2	3

Weighted Sum = 0*8 + 1*3 + 2*1 + 3*2 = 11

Solution

```
#include <iostream>

// Function to calculate the weighted sum of an integer array
int weightedSum(int A[], int S) {
   int sum = 0;

   for (int i = 0; i < S; ++i) {
      sum += i * A[i];
   }

   return sum;
}

int main() {
   // Example usage
   const int S = 4;
   int A[S] = {1, 2, 3, 4};

   // Calculate and display the weighted sum
   int result = weightedSum(A, S);
   std::cout << "Weighted Sum: " << result << std::endl;

   return 0;
}
```

Question No. 4 **Marks: 5**

What is stored in `myList` after the following C++ code executes?

```
double myList[6];

myList[0] = 2.5;

for (int i = 1; i < 6; i++)
{
    myList[i] = i * myList[i - 1];
    if (i > 3)
        myList[i] = myList[i] / 2;
}
```

Solution

2.5 2.5 5 15 30 75

Question No. 5 **Marks: 5**

What is the output of the following C++ code?

```
#include <iostream>

using namespace std;

int main()
{
    int alpha[10];
    int beta[15];

    for (int i = 0; i < 5; i++)
    {
        alpha[i] = 2 * i + 1;
        alpha[5 + i] = 3 * i - 1;
        beta[i] = 5 * i - 2;
    }

    cout << "alpha: ";
    for (int i = 0; i < 10; i++)
        cout << alpha[i] << " ";
    cout << endl;

    for (int i = 5; i < 10; i++)
    {
        beta[i] = alpha[9 - i] + beta[9 - i];
        beta[i + 5] = beta[9 - i] + beta[i];
    }

    cout << "beta: ";
    for (int i = 0; i < 15; i++)
        cout << beta[i] << " ";
    cout << endl;

    return 0;
}
```

Solution

1 3 5 7 9 -1 2 5 8 11

-2 3 8 13 18 27 20 13 6 -1 45 33 21 9 -3