School of Mechanical & Manufacturing Engineering (SMME),

National University of Science and Technology (NUST),

Sector H-12, Islamabad

Assignment no. 1

Course Title: Fundamentals of Programming (CS-109)

Name: Al Ameerah Bint E Waqar

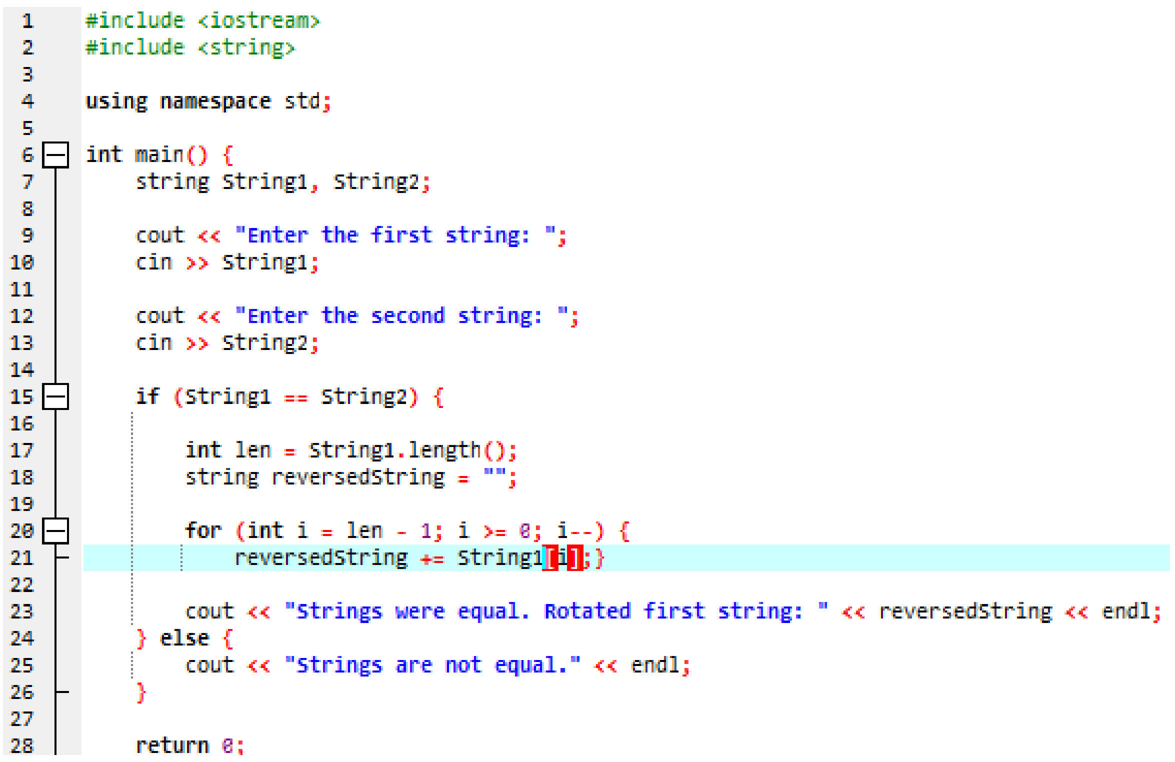
Qalam ID: 480966

Program: BE-Aerospace

Section: AE-01

Session: Fall 2023, 1st Semester

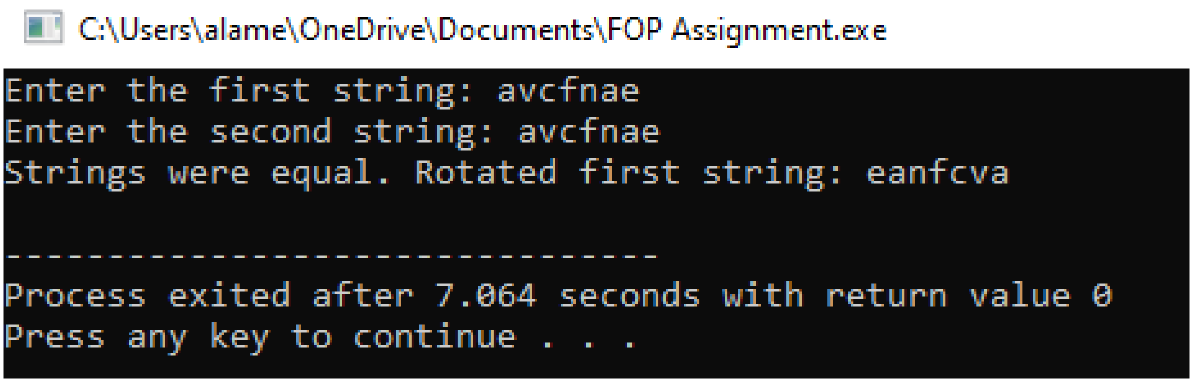
- **Question 01**



```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
    string String1, String2;

    cout << "Enter the first string: ";
    cin >> String1;

    cout << "Enter the second string: ";
    cin >> String2;

    if (String1 == String2) {

        int len = String1.length();
        string reversedString = "";

        for (int i = len - 1; i >= 0; i--) {
            reversedString += String1[i];}

        cout << "Strings were equal. Rotated first string: " << reversedString << endl;
    } else {
        cout << "Strings are not equal." << endl;
    }

    return 0;
```

Figure 1: Code for Question 1

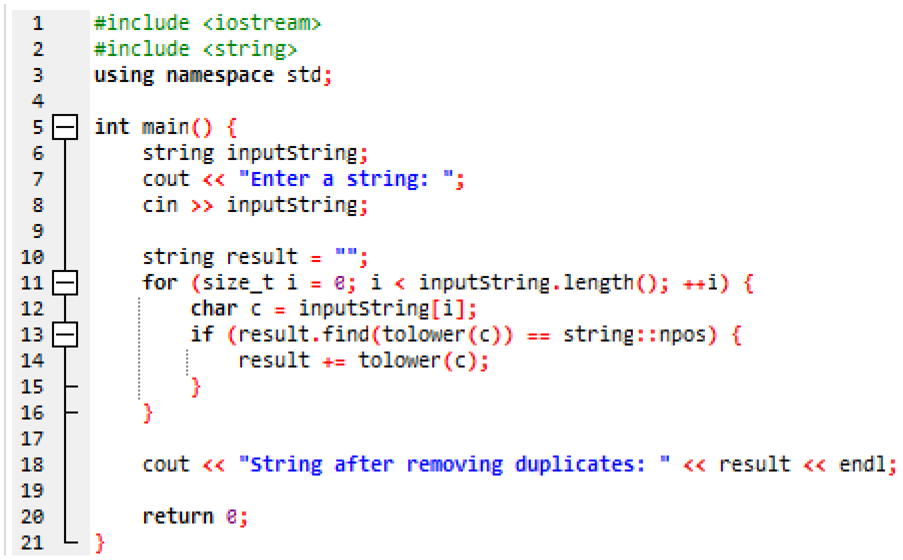
This C++ code prompts users to input two strings, compares them, and if they are identical, it reverses the first string and displays the reversed version. It checks for string equality using ==, then reverses String1 character by character using a for loop. The reversed string is stored in reversedString and printed if the strings are equal; otherwise, it declares that the strings are not equal.



```
C:\Users\alame\OneDrive\Documents\FOP Assignment.exe
Enter the first string: avcfnae
Enter the second string: avcfnae
Strings were equal. Rotated first string: eanfcva

--------------------------------
Process exited after 7.064 seconds with return value 0
Press any key to continue . . .
```

Figure 2: Output window for Question 1

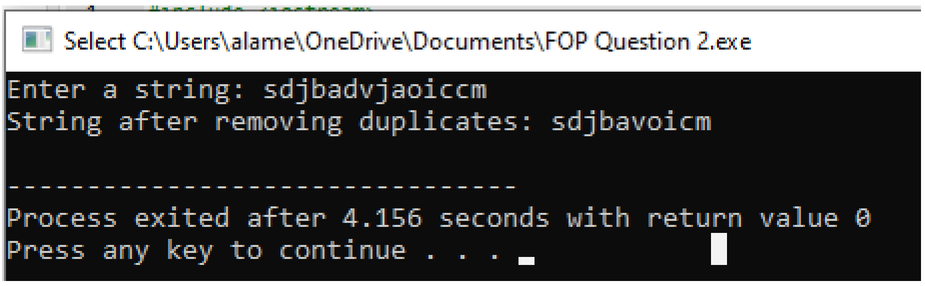
- **Question 02**



```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string inputString;
    cout << "Enter a string: ";
    cin >> inputString;

    string result = "";
    for (size_t i = 0; i < inputString.length(); ++i) {
        char c = inputString[i];
        if (result.find(tolower(c)) == string::npos) {
            result += tolower(c);
        }
    }

    cout << "String after removing duplicates: " << result << endl;

    return 0;
}
```

Figure 3 Output window for Question 2

This C++ code receives a string input from the user, then iterates through each character of the input string. It checks if the lowercase version of the character exists in the result string using find() combined with tolower(). If the character isn't found in the result string, it appends the lowercase version of the character to result. This process effectively removes duplicate characters, case-insensitively, from the input string. Finally, it displays the modified string without duplicates as the output.

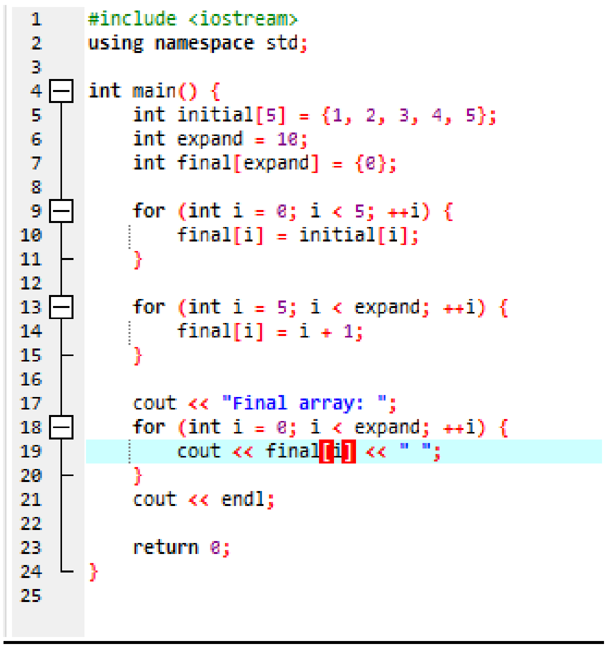


Select C:\Users\alame\OneDrive\Documents\FOP Question 2.exe

```
Enter a string: sdjbadvjaoiccm
String after removing duplicates: sdjbavoicm

--------------------------------
Process exited after 4.156 seconds with return value 0
Press any key to continue . . .
```

Figure 4: Output window for Question 2

- **<u>Question 3</u>**



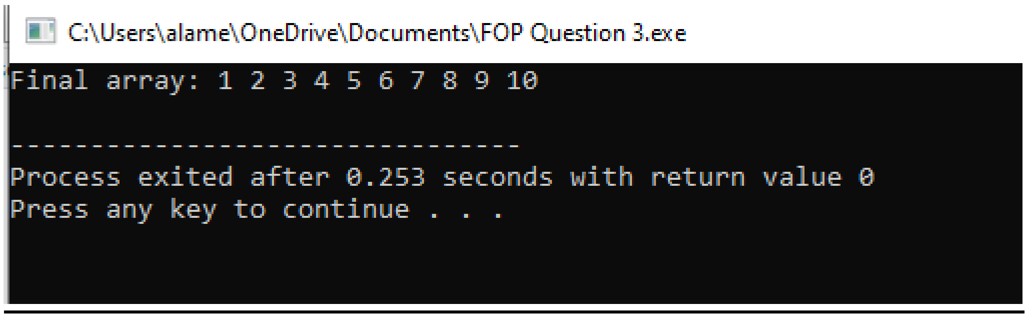
```cpp
#include <iostream>
using namespace std;

int main() {
    int initial[5] = {1, 2, 3, 4, 5};
    int expand = 10;
    int final[expand] = {0};

    for (int i = 0; i < 5; ++i) {
        final[i] = initial[i];
    }

    for (int i = 5; i < expand; ++i) {
        final[i] = i + 1;
    }

    cout << "Final array: ";
    for (int i = 0; i < expand; ++i) {
        cout << final[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Figure 5: Output window for Question 3

This C++ code demonstrates an array manipulation process. It initializes an array named initial with five elements (1, 2, 3, 4, 5). Then, it declares an integer variable expand set to 10, which determines the size of the final array.

The code proceeds to copy the elements from the initial array to the start of the final array. Subsequently, it populates the remaining elements of the final array with consecutive numbers starting from 6 up to the value of expand (which is 10 in this case).

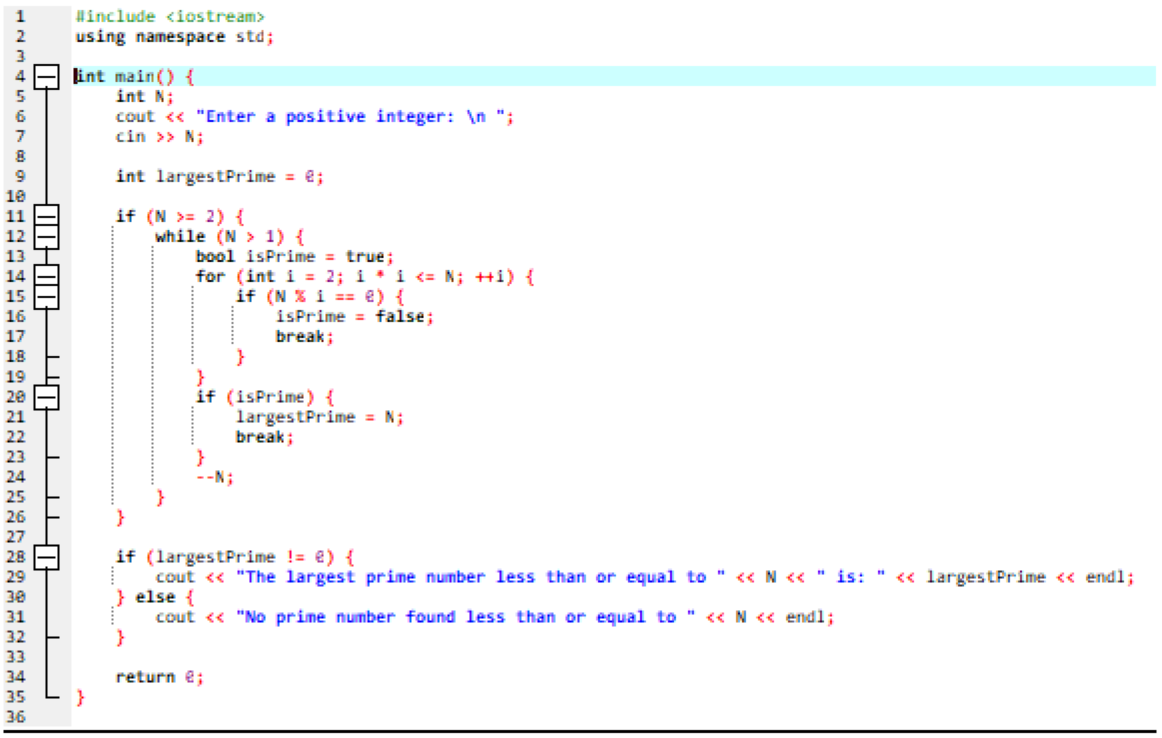
Finally, it displays the content of the final array, showing the merged and expanded array containing elements from initial followed by additional numbers from 6 to 10.



```
C:\Users\alame\OneDrive\Documents\FOP Question 3.exe
Final array: 1 2 3 4 5 6 7 8 9 10

--------------------------------
Process exited after 0.253 seconds with return value 0
Press any key to continue . . .
```

Figure 6: Output window for Question 3

- **<u>Question 4:</u>**



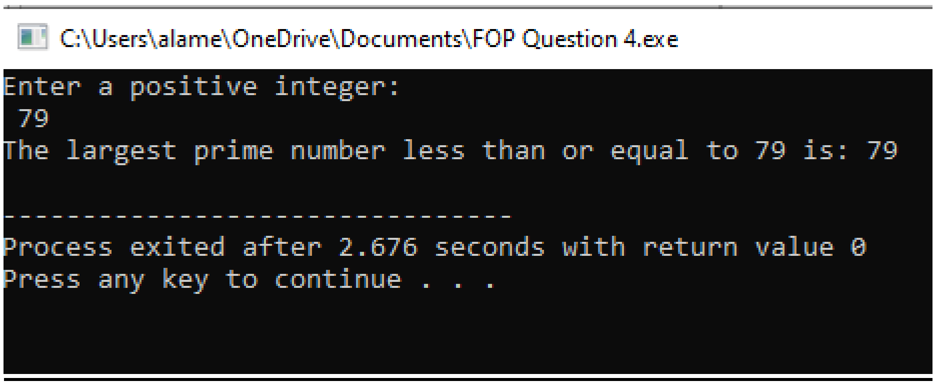
```cpp
#include <iostream>
using namespace std;

int main() {
    int N;
    cout << "Enter a positive integer: \n ";
    cin >> N;

    int largestPrime = 0;

    if (N >= 2) {
        while (N > 1) {
            bool isPrime = true;
            for (int i = 2; i * i <= N; ++i) {
                if (N % i == 0) {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime) {
                largestPrime = N;
                break;
            }
            --N;
        }
    }

    if (largestPrime != 0) {
        cout << "The largest prime number less than or equal to " << N << " is: " << largestPrime << endl;
    } else {
        cout << "No prime number found less than or equal to " << N << endl;
    }

    return 0;
}
```

Figure 7: Output window for Question 4

This C++ code aims to find the largest prime number less than or equal to a given positive integer N. It first prompts the user to input a positive integer, then initializes a variable largestPrime to store the largest prime number found.

The code checks if the input N is greater than or equal to 2. If it is, it enters a loop where it iterates down from N. Within this loop, it checks for primality using a nested loop: it verifies whether each number from 2 up to the square root of N is a factor of N. If it finds a factor, it marks isPrime as false and breaks the loop. If no factors are found, it updates largestPrime with the current value of N and breaks out of the loop.

Finally, it checks if largestPrime is not zero. If so, it displays the largest prime number found less than or equal to the original input N. Otherwise, it indicates that no prime number was found within the given range.

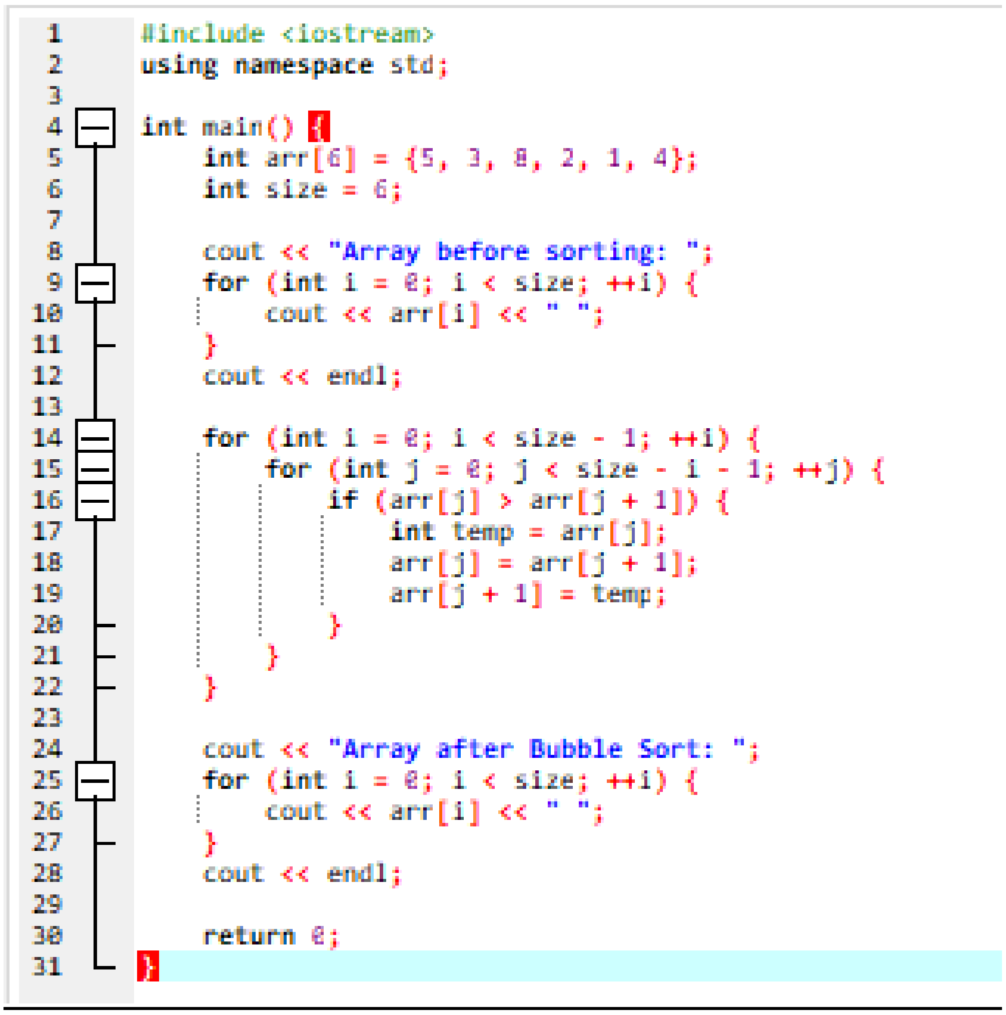


C:\Users\alame\OneDrive\Documents\FOP Question 4.exe

```
Enter a positive integer:
 79
The largest prime number less than or equal to 79 is: 79

--------------------------------
Process exited after 2.676 seconds with return value 0
Press any key to continue . . .
```

Figure 8: Output window for Question 4

- **Question 5:**



```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[6] = {5, 3, 8, 2, 1, 4};
    int size = 6;

    cout << "Array before sorting: ";
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;

    for (int i = 0; i < size - 1; ++i) {
        for (int j = 0; j < size - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    cout << "Array after Bubble Sort: ";
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

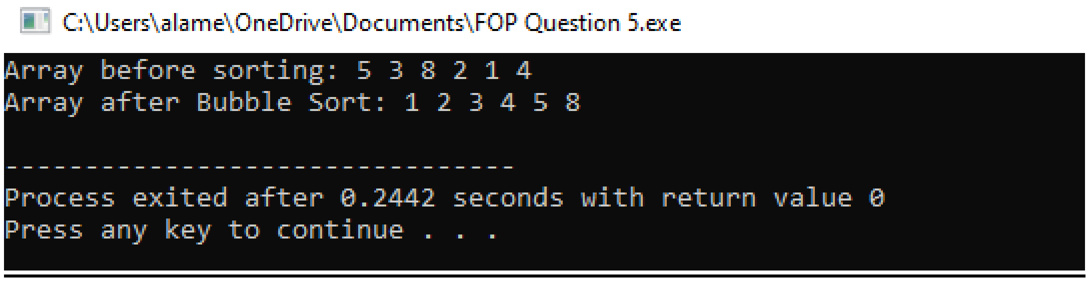
Figure 9: Output window for Question 5

This C++ code demonstrates a sorting technique known as "Bubble Sort" applied to an array of integers. It initializes an array arr with six elements and a variable size set to 6 to represent the size of the array.

The code first displays the contents of the array before sorting. It then employs a nested loop structure to perform the Bubble Sort algorithm. Within the nested loops, it compares adjacent elements in the array and swaps them if they are in the wrong order, moving through the array multiple times until it's sorted.

During each iteration of the outer loop (i), the inner loop (j) performs comparisons and swaps to gradually move larger elements towards the end of the array.

Once the sorting process is complete, it displays the sorted array using cout. Bubble Sort repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order, gradually moving the larger elements towards the end of the array, resulting in an ascending order of elements in this case.

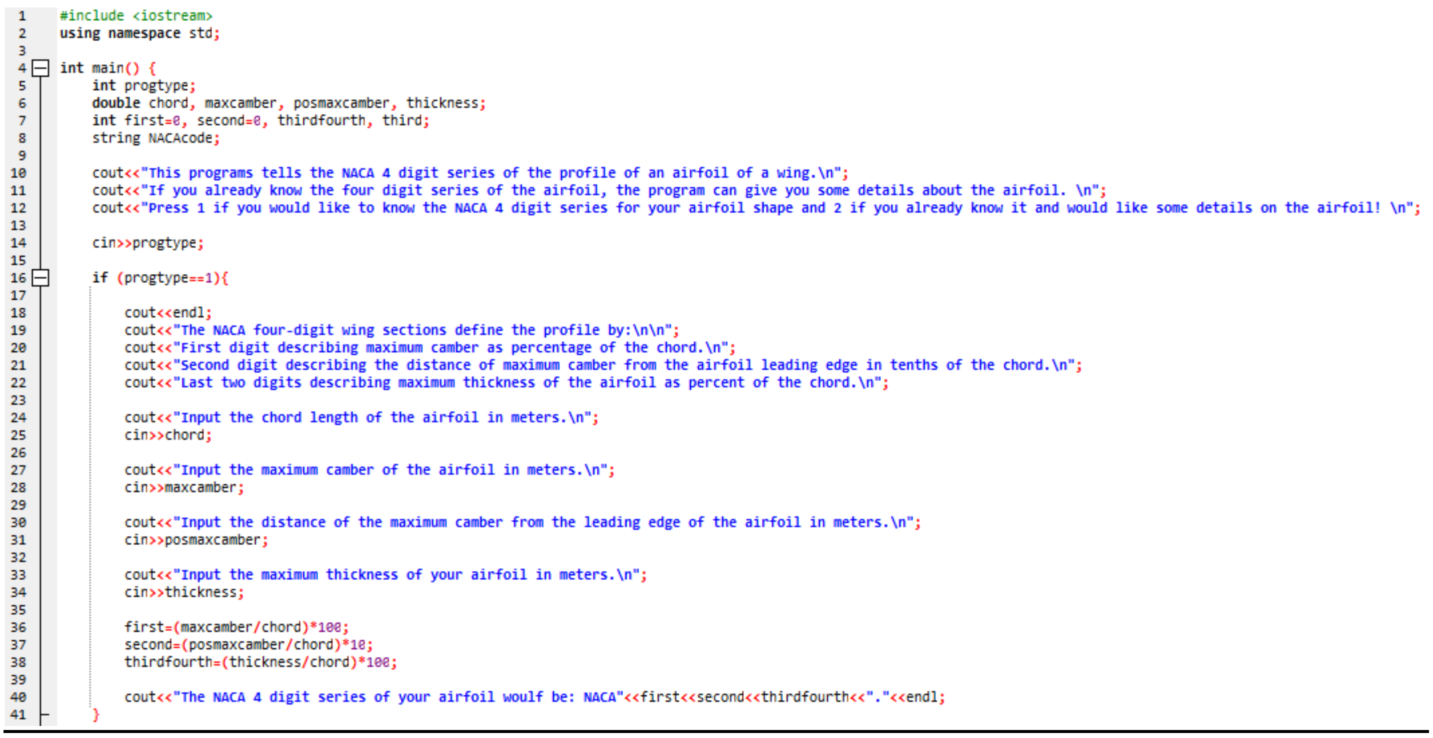


C:\Users\alame\OneDrive\Documents\FOP Question 5.exe

```
Array before sorting: 5 3 8 2 1 4
Array after Bubble Sort: 1 2 3 4 5 8

--------------------------------
Process exited after 0.2442 seconds with return value 0
Press any key to continue . . .
```

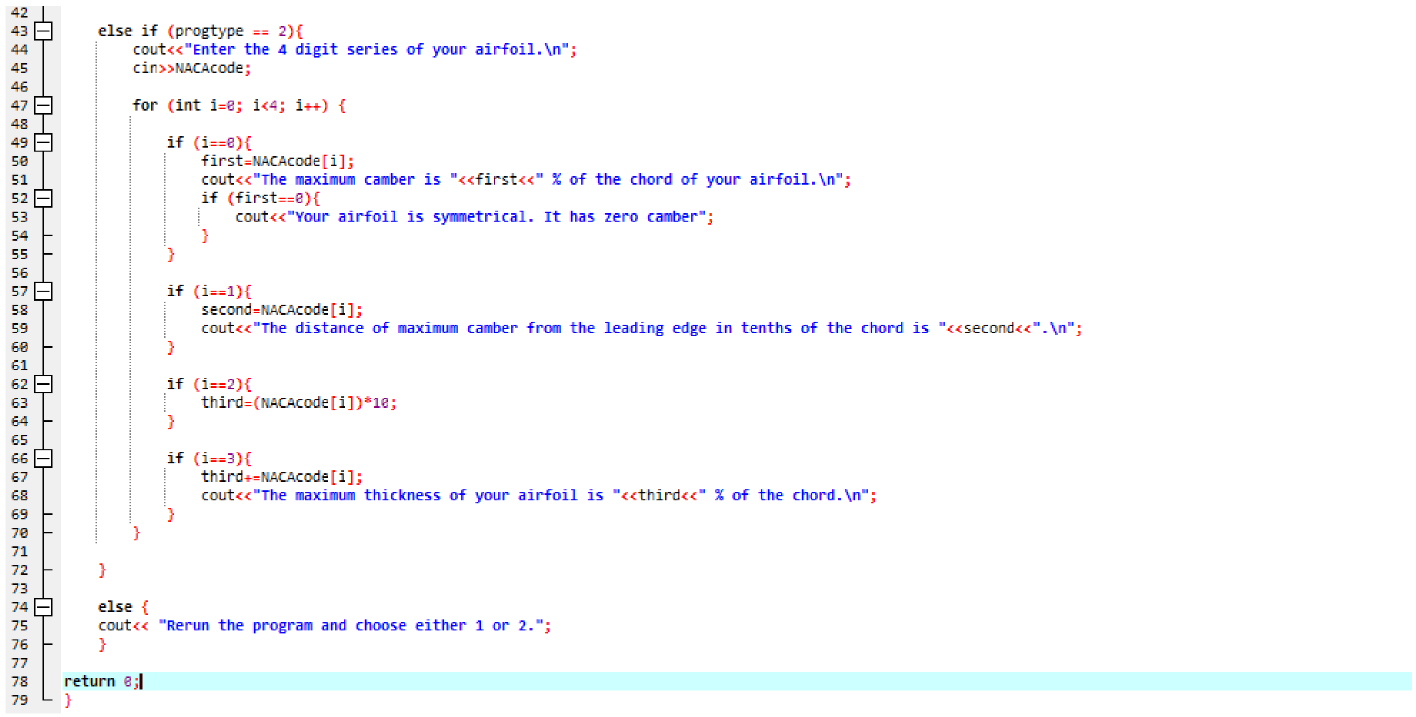
Figure 10: Output window for Question 5

- **<u>Question 6:</u>**



```cpp
#include <iostream>
using namespace std;

int main() {
    int progtype;
    double chord, maxcamber, posmaxcamber, thickness;
    int first=0, second=0, thirdfourth, third;
    string NACAcode;

    cout<<"This programs tells the NACA 4 digit series of the profile of an airfoil of a wing.\n";
    cout<<"If you already know the four digit series of the airfoil, the program can give you some details about the airfoil. \n";
    cout<<"Press 1 if you would like to know the NACA 4 digit series for your airfoil shape and 2 if you already know it and would like some details on the airfoil! \n";

    cin>>progtype;

    if (progtype==1){

        cout<<endl;
        cout<<"The NACA four-digit wing sections define the profile by:\n\n";
        cout<<"First digit describing maximum camber as percentage of the chord.\n";
        cout<<"Second digit describing the distance of maximum camber from the airfoil leading edge in tenths of the chord.\n";
        cout<<"Last two digits describing maximum thickness of the airfoil as percent of the chord.\n";

        cout<<"Input the chord length of the airfoil in meters.\n";
        cin>>chord;

        cout<<"Input the maximum camber of the airfoil in meters.\n";
        cin>>maxcamber;

        cout<<"Input the distance of the maximum camber from the leading edge of the airfoil in meters.\n";
        cin>>posmaxcamber;

        cout<<"Input the maximum thickness of your airfoil in meters.\n";
        cin>>thickness;

        first=(maxcamber/chord)*100;
        second=(posmaxcamber/chord)*10;
        thirdfourth=(thickness/chord)*100;

        cout<<"The NACA 4 digit series of your airfoil woulf be: NACA"<<first<<second<<thirdfourth<<"."<<endl;
    }
```

Figure 11: Code for Question 6 (part 1)



```
    else if (progtype == 2){
        cout<<"Enter the 4 digit series of your airfoil.\n";
        cin>>NACAcode;

        for (int i=0; i<4; i++) {

            if (i==0){
                first=NACAcode[i];
                cout<<"The maximum camber is "<<first<<" % of the chord of your airfoil.\n";
                if (first==0){
                    cout<<"Your airfoil is symmetrical. It has zero camber";
                }
            }

            if (i==1){
                second=NACAcode[i];
                cout<<"The distance of maximum camber from the leading edge in tenths of the chord is "<<second<<".\n";
            }

            if (i==2){
                third=(NACAcode[i])*10;
            }

            if (i==3){
                third+=NACAcode[i];
                cout<<"The maximum thickness of your airfoil is "<<third<<" % of the chord.\n";
            }
        }

    }

    else {
    cout<< "Rerun the program and choose either 1 or 2.";
    }

return 0;
}
```

Figure 12: Code for Question 6 (part 2)

This code finds the NACA 4-digit series of an airfoil for the user, and if the user already has it, it takes it as an input string and gives the user information on the airfoil. Hence solving an aerospace engineering related problem.