School of Mechanical & Manufacturing Engineering (SMME),

National University of Science and Technology (NUST),

Sector H-12, Islamabad

Program: BE-Aerospace Section: AE-01

Session: Fall 2023 Semester: 1st

Course Title: Fundamentals of Programming (CS-114)

Assignment: # 01

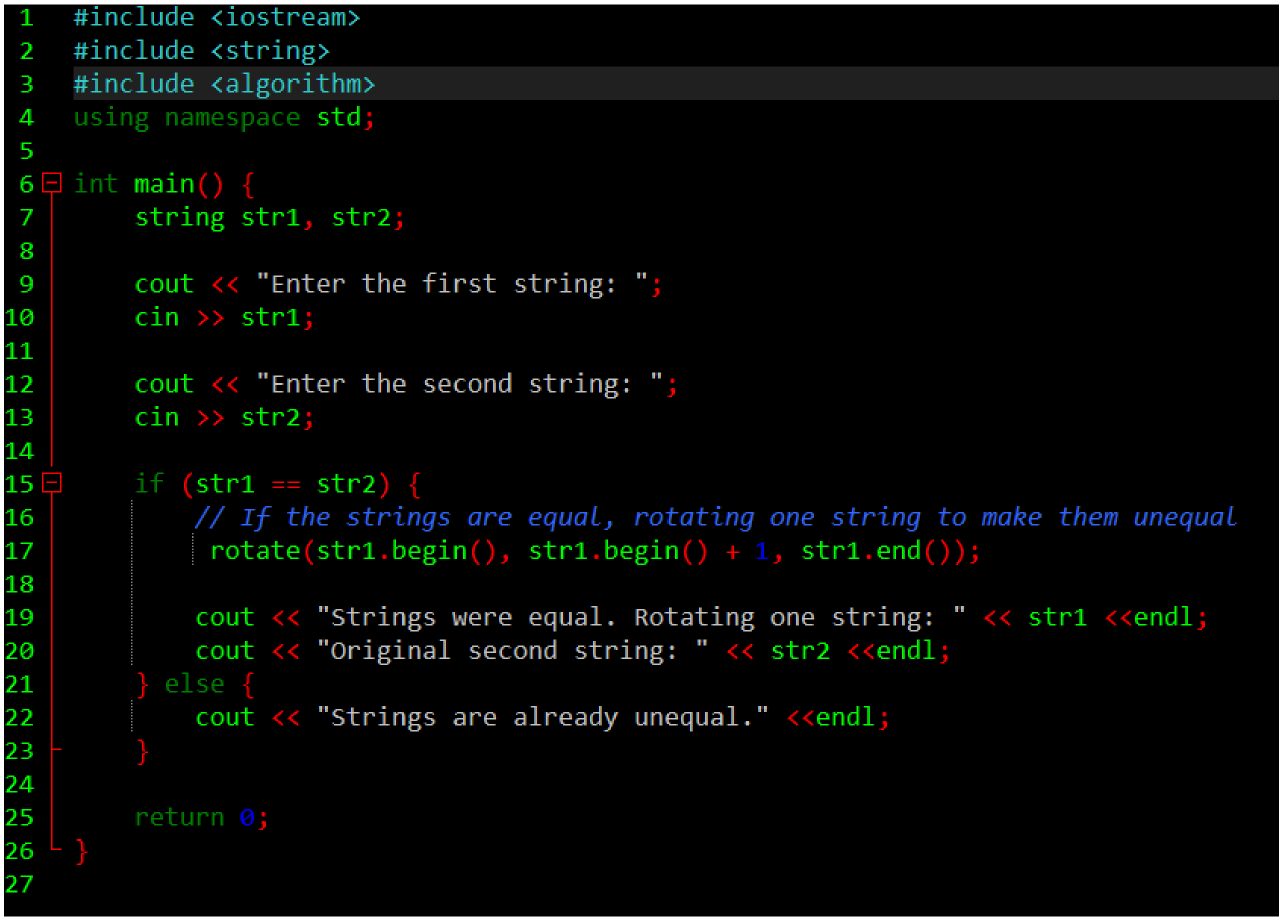
Fundamentals of Programming

Name: ***Ahmed Mukhtar Bukhari***

CMS: 461298

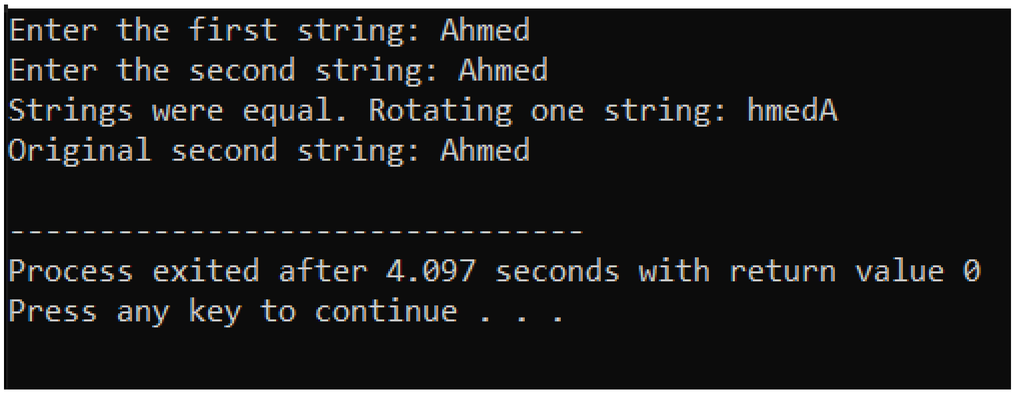
Question 1:

Write a C++ program, take two strings as input from user and check if both strings are equal or not. If they are equal make them unequal by rotating string. e.g., Hello is turned into olleH etc.



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

int main() {
    string str1, str2;

    cout << "Enter the first string: ";
    cin >> str1;

    cout << "Enter the second string: ";
    cin >> str2;

    if (str1 == str2) {
        // If the strings are equal, rotating one string to make them unequal
         rotate(str1.begin(), str1.begin() + 1, str1.end());

        cout << "Strings were equal. Rotating one string: " << str1 <<endl;
        cout << "Original second string: " << str2 <<endl;
    } else {
        cout << "Strings are already unequal." <<endl;
    }

    return 0;
}
```

Output:



```
Enter the first string: Ahmed
Enter the second string: Ahmed
Strings were equal. Rotating one string: hmedA
Original second string: Ahmed

--------------------------------
Process exited after 4.097 seconds with return value 0
Press any key to continue . . .
```

1- Input: Requests two strings from the user.

2- Comparison: Checks if the entered strings are equal.

3- String Rotation:
Adding #include <algorithm> at the beginning includes the necessary functionalities like rotate to perform the string rotation,

If the strings are equal:

Uses rotate to rotate the first string, moving its first character to the end.

Displays the rotated first string and the original second string.

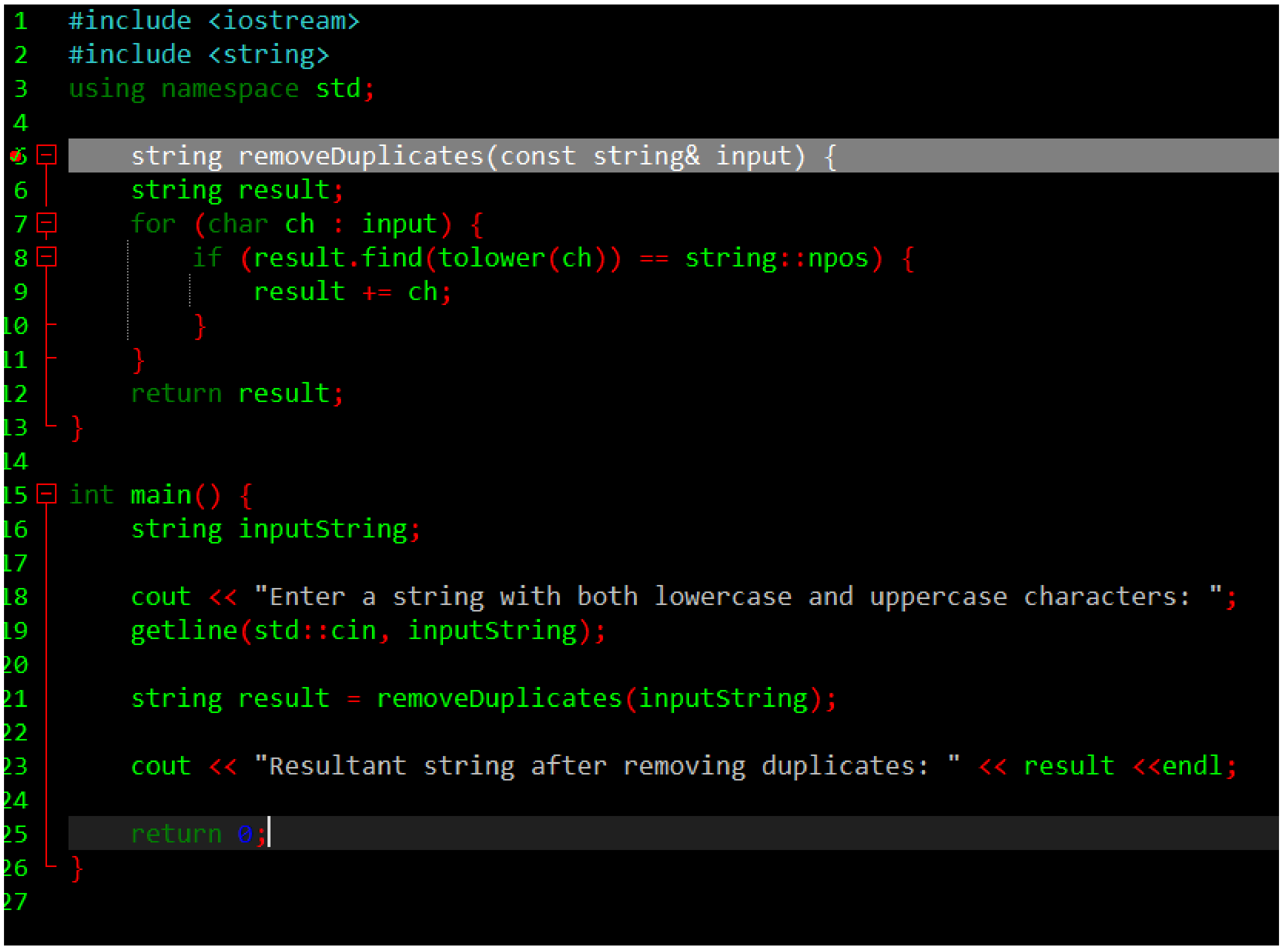
If the strings are already unequal, it informs the user.

4- Output: Shows either the rotated strings (if they were equal) or a message stating that the strings are already different.

5- End: Program completes after this check and output.

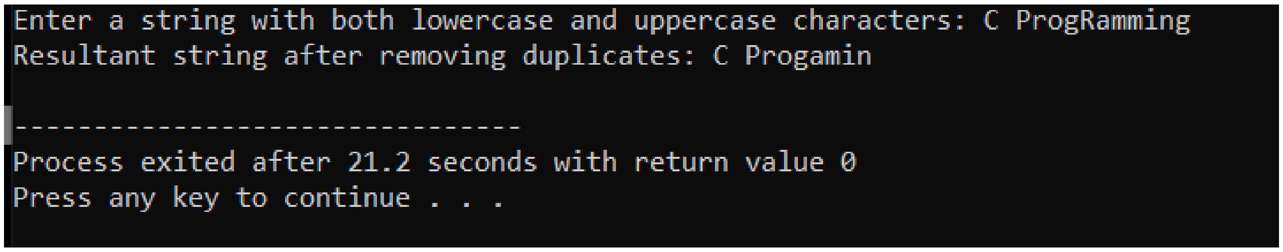
Question 2:

Write a C++ program for a string which may contain lowercase and uppercase characters. The task is to remove all duplicate characters from the string and find the resultant string.



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

    string removeDuplicates(const string& input) {
    string result;
    for (char ch : input) {
        if (result.find(tolower(ch)) == string::npos) {
            result += ch;
        }
    }
    return result;
}

int main() {
    string inputString;

    cout << "Enter a string with both lowercase and uppercase characters: ";
    getline(std::cin, inputString);

    string result = removeDuplicates(inputString);

    cout << "Resultant string after removing duplicates: " << result <<endl;

    return 0;
}
```

Output:



```
Enter a string with both lowercase and uppercase characters: C ProgRamming
Resultant string after removing duplicates: C Progamin

--------------------------------
Process exited after 21.2 seconds with return value 0
Press any key to continue . . .
```

1- Function to Remove Duplicates (removeDuplicates)

Parameters: const string & input (the input string)

Return Type: string (the resultant string after removing duplicates)

removeDuplicates Function Explanation:

1- Initialization:
Declares an empty string **result** to store the resultant string without duplicates.

2- Iterating through the Input String:
Utilizes a **for** loop to go through each character **(char ch)** in the input string.

3- Checking for Duplicates:
Uses **result.find(tolower(ch))** to search for the lowercase version of the current character in the **result** string.

If **tolower(ch)** (the lowercase version of ch) is not found in **result (returns string::npos** indicating "not found"):
Appends the current character ch to the result string.

4- Return Result:
Returns the result string containing unique characters.

2- main() Function:

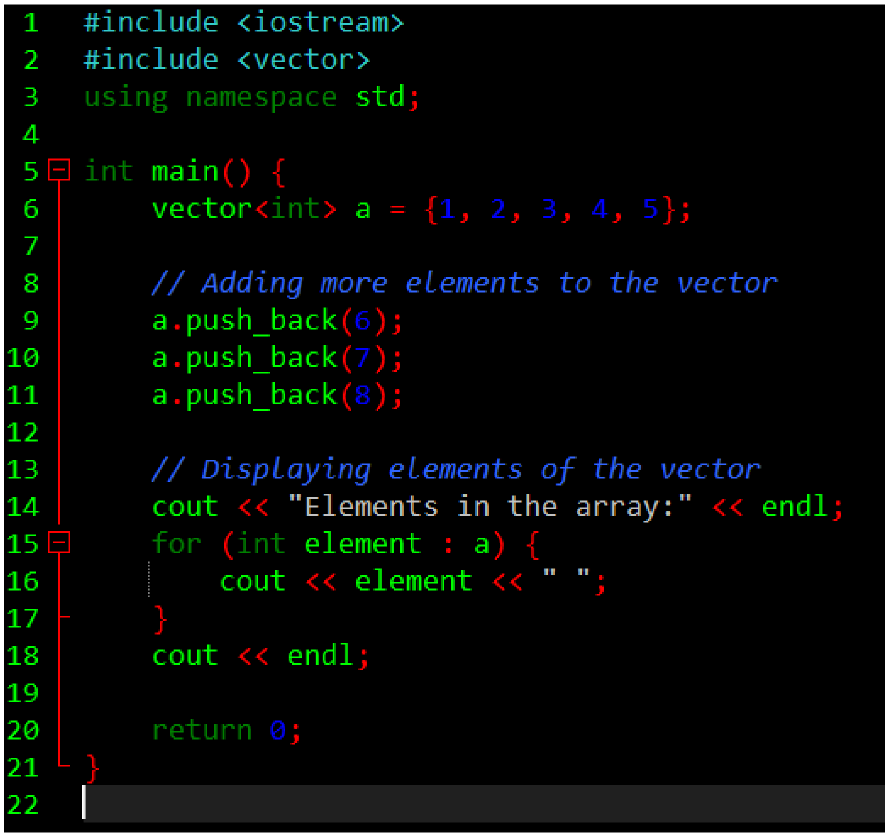
1- User Input:
Asks the user to input a string containing both lowercase and uppercase characters.
Reads the input string using getline to allow for spaces in the string.

2- Calling removeDuplicates:
Calls the removeDuplicates function with the user-input string.
Stores the resultant string in result.

3- Display Result:
Prints the resultant string (without duplicate characters) to the console.

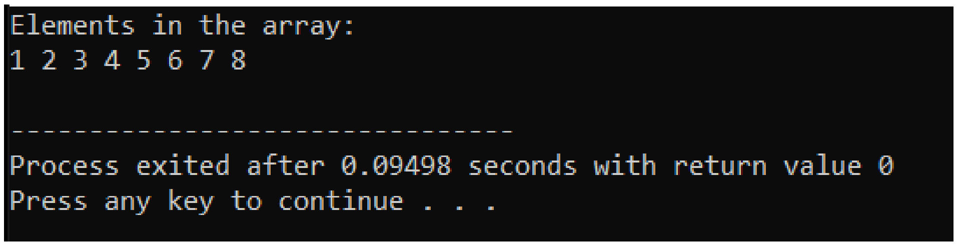
Question 3:

Suppose an integer array a[5] = {1,2,3,4,5}. Add more elements to it and display them in C++.



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

int main() {
    vector<int> a = {1, 2, 3, 4, 5};

    // Adding more elements to the vector
    a.push_back(6);
    a.push_back(7);
    a.push_back(8);

    // Displaying elements of the vector
    cout << "Elements in the array:" << endl;
    for (int element : a) {
        cout << element << " ";
    }
    cout << endl;

    return 0;
}
```

Output:



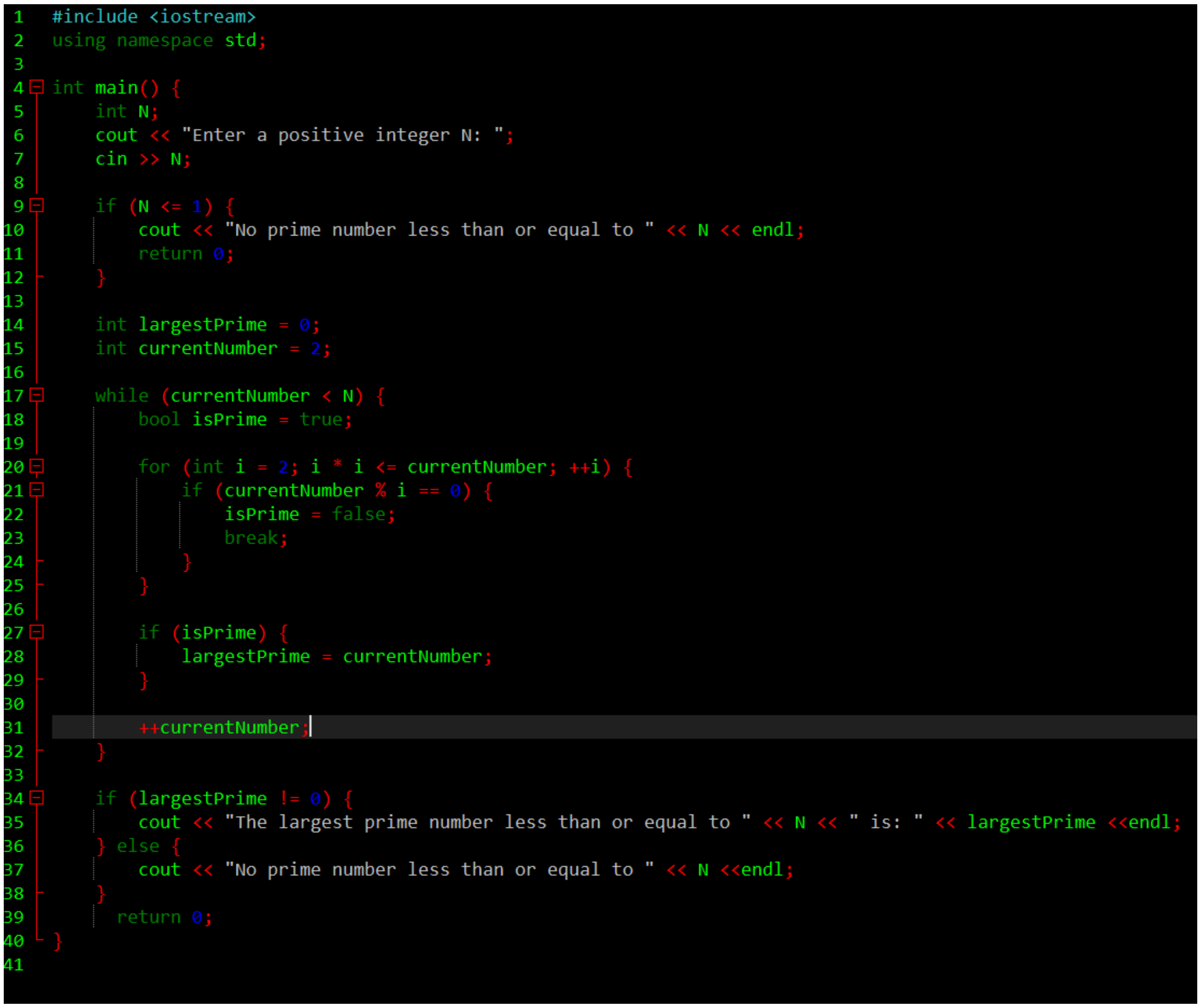
```
Elements in the array:
1 2 3 4 5 6 7 8

--------------------------------
Process exited after 0.09498 seconds with return value 0
Press any key to continue . . .
```

1- vector<int> a = {1, 2, 3, 4, 5}; initializes a vector **a** with initial elements.
2- a.push_back(x); adds elements **x** to the end of the vector.
3- The loop **for (int element : a)** iterates through all elements in the vector and displays them.

This way, using vector, we can add elements dynamically to a collection and display them in C++.

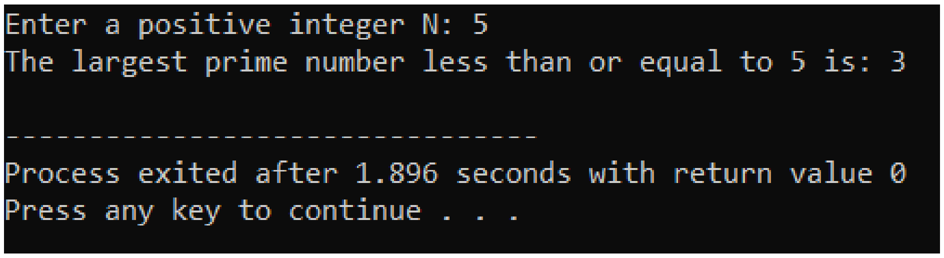
Question 4:

Write a C++ program that uses a while loop to find the largest prime number less than a given positive integer N. Your program should take the value of N as input from the user and then find the largest prime number less than or equal to N. You are not allowed to use any library or pre-existing functions to check for prime numbers.



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

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

    if (N <= 1) {
        cout << "No prime number less than or equal to " << N << endl;
        return 0;
    }

    int largestPrime = 0;
    int currentNumber = 2;

    while (currentNumber < N) {
        bool isPrime = true;

        for (int i = 2; i * i <= currentNumber; ++i) {
            if (currentNumber % i == 0) {
                isPrime = false;
                break;
            }
        }

        if (isPrime) {
            largestPrime = currentNumber;
        }

        ++currentNumber;
    }

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

Output:



```
Enter a positive integer N: 5
The largest prime number less than or equal to 5 is: 3

--------------------------------
Process exited after 1.896 seconds with return value 0
Press any key to continue . . .
```

1- Taking Input: Asks the user for a positive integer **N.**

2-Validity Check: Checks if the input is less than or equal to 1. If so, it informs the user that there's no prime number less than or equal to that value.

3- Finding Largest Prime: Initializes **largestPrime** to 0 and starts a while loop from **2 up to N - 1.**

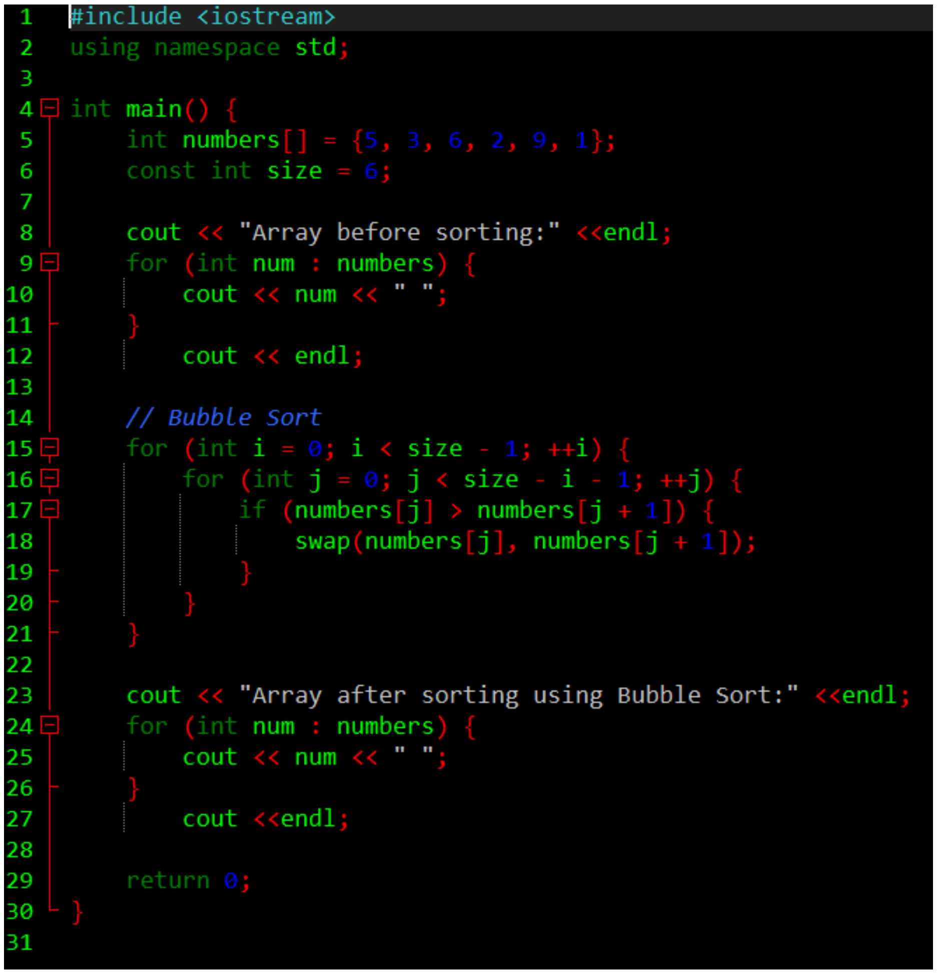
4- Prime Check: For each **currentNumber** within the loop, it checks if the number is prime by iterating from 2 to the square root of the number **(i * i <= currentNumber).** If the number is divisible by any i, it's not prime.

5- Storing the Largest Prime: If the **currentNumber** is prime, it updates the **largestPrime** variable with that number.

6- Display Result: After the loop, it displays the largest prime number less than or equal to N if found, else notifies that there's no such prime number.

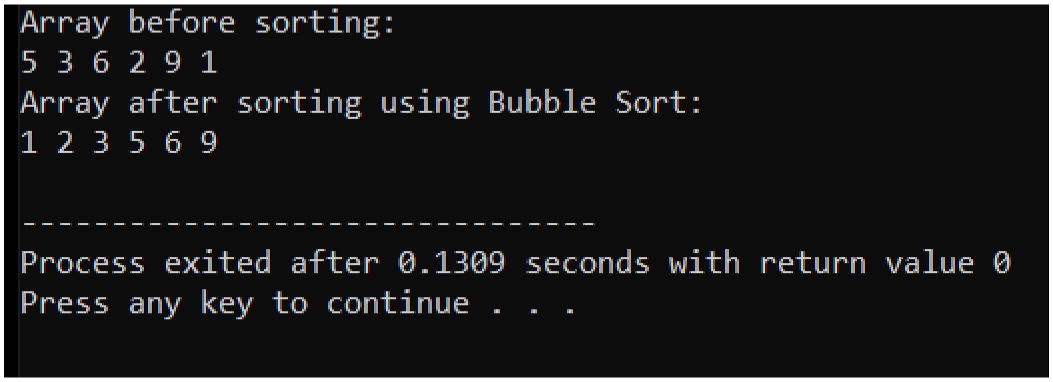
Question 5:

Implement Bubble Sort on an array of 6 integers.



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

int main() {
    int numbers[] = {5, 3, 6, 2, 9, 1};
    const int size = 6;

    cout << "Array before sorting:" <<endl;
    for (int num : numbers) {
        cout << num << " ";
    }
        cout << endl;

    // Bubble Sort
    for (int i = 0; i < size - 1; ++i) {
        for (int j = 0; j < size - i - 1; ++j) {
            if (numbers[j] > numbers[j + 1]) {
                swap(numbers[j], numbers[j + 1]);
            }
        }
    }

    cout << "Array after sorting using Bubble Sort:" <<endl;
    for (int num : numbers) {
        cout << num << " ";
    }
        cout <<endl;

    return 0;
}
```

Output:



```
Array before sorting:
5 3 6 2 9 1
Array after sorting using Bubble Sort:
1 2 3 5 6 9

--------------------------------
Process exited after 0.1309 seconds with return value 0
Press any key to continue . . .
```

1- Array Initialization: Initializes an array **numbers** of size 6 with integer values.

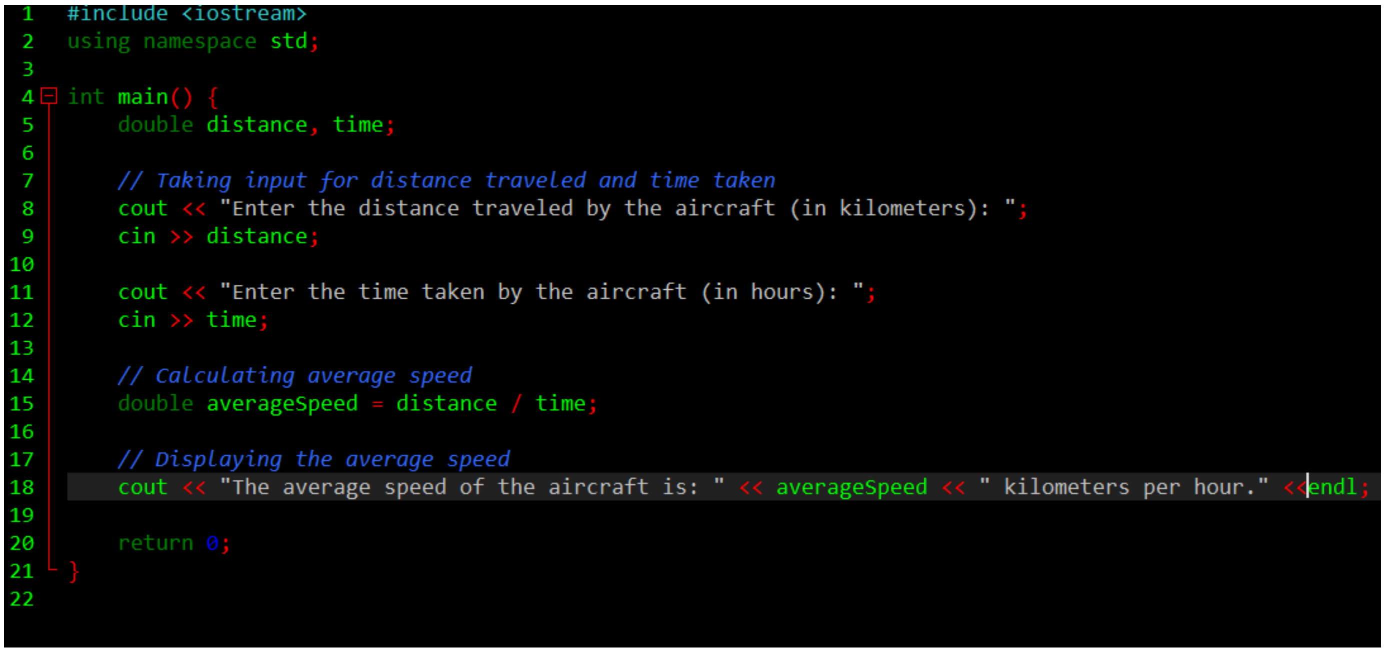
2- Displaying the Original Array:
Outputs the elements of the array before sorting.

3-Bubble Sort:
Implements the Bubble Sort algorithm:
Nested loops are used to iterate through the array.
Compares adjacent elements and swaps them if they are in the wrong order.
Moves the largest elements to the end of the array by iterating through and swapping pairs.

4-Displaying the Sorted Array:
Outputs the elements of the array after applying Bubble Sort to demonstrate the sorted sequence.
In summary, this code showcases the Bubble Sort algorithm applied to an array of 6 integers, displaying the array before and after sorting to illustrate the sorting process.

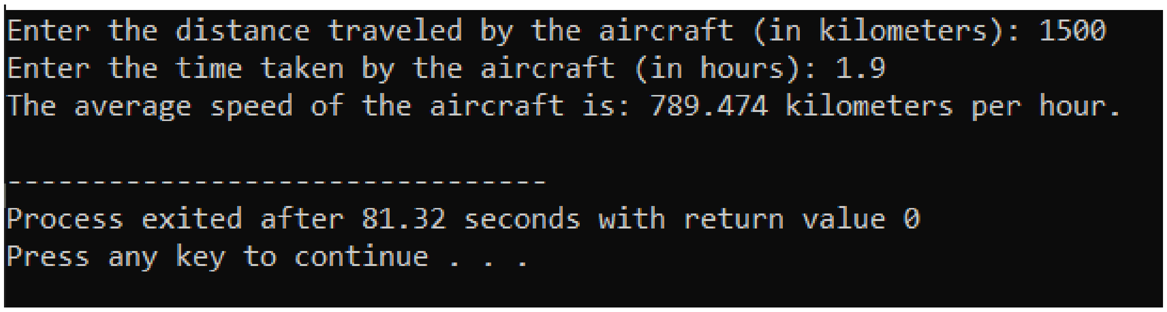
Question 6:

In this question, I chose a simple Aerospace Engineering related problem. Let's consider a simple real-life problem related to calculating the average speed of an aircraft during a flight. Suppose an aircraft travels a certain distance in a given time. We can calculate its average speed using C++. Here's a basic program that computes the average speed of an aircraft:



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

int main() {
    double distance, time;

    // Taking input for distance traveled and time taken
    cout << "Enter the distance traveled by the aircraft (in kilometers): ";
    cin >> distance;

    cout << "Enter the time taken by the aircraft (in hours): ";
    cin >> time;

    // Calculating average speed
    double averageSpeed = distance / time;

    // Displaying the average speed
    cout << "The average speed of the aircraft is: " << averageSpeed << " kilometers per hour." <<endl;

    return 0;
}
```

Output:



```
Enter the distance traveled by the aircraft (in kilometers): 1500
Enter the time taken by the aircraft (in hours): 1.9
The average speed of the aircraft is: 789.474 kilometers per hour.

--------------------------------
Process exited after 81.32 seconds with return value 0
Press any key to continue . . .
```

The program takes input for the distance traveled by the aircraft (in kilometers) and the time taken for the journey (in hours).

It calculates the average speed by dividing the distance traveled by the time taken.

Displays the calculated average speed in kilometers per hour.

This program provides a simple way to calculate the average speed of an aircraft based on the distance traveled and the time taken, which is a fundamental aspect of aerospace and real-life transportation problems.