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: Fundamentals of Programming	Course Code: CS-114

"ASSIGNMENT # 01"

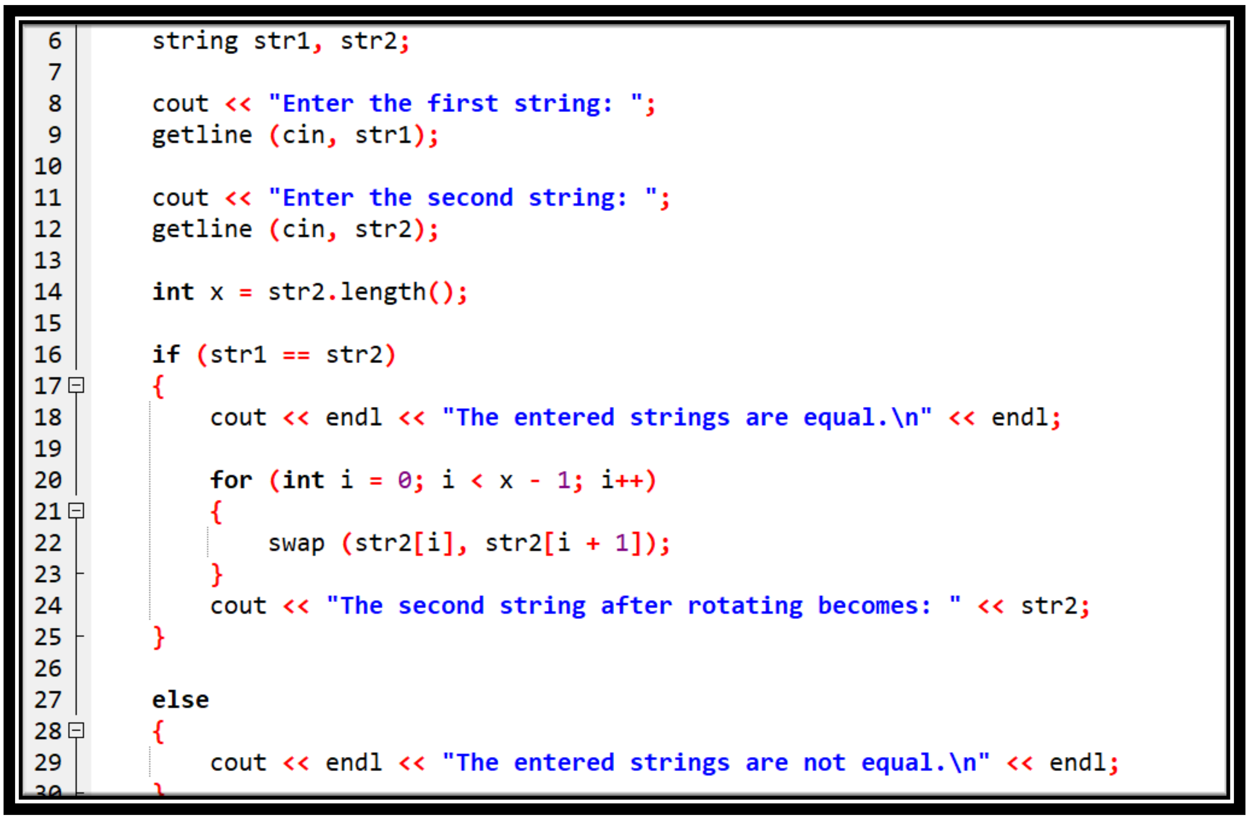
Name: M.MUSHAF KHAN

CMS: 459299

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.

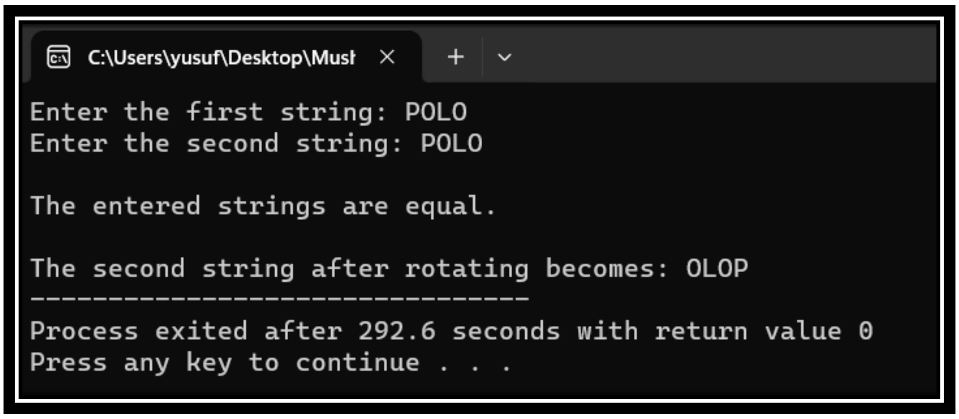
Answer:

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

int main ()
{
        string str1, str2;
        cout << "Enter the first string: ";
        getline (cin, str1);

        cout << "Enter the second string: ";
        getline (cin, str2);

        int x = str2.length();

        if (str1 == str2)
        {
                cout << endl << "The entered strings are equal.\n" << endl;

           for (int i = 0; i < x - 1; i++)
           {
                swap (str2[i], str2[i + 1]);
           }
                cout << "The second string after rotating becomes: " << str2;
        }
        else
        {
                cout << endl << "The entered strings are not equal.\n" << endl;
        }
        return 0;
}
```



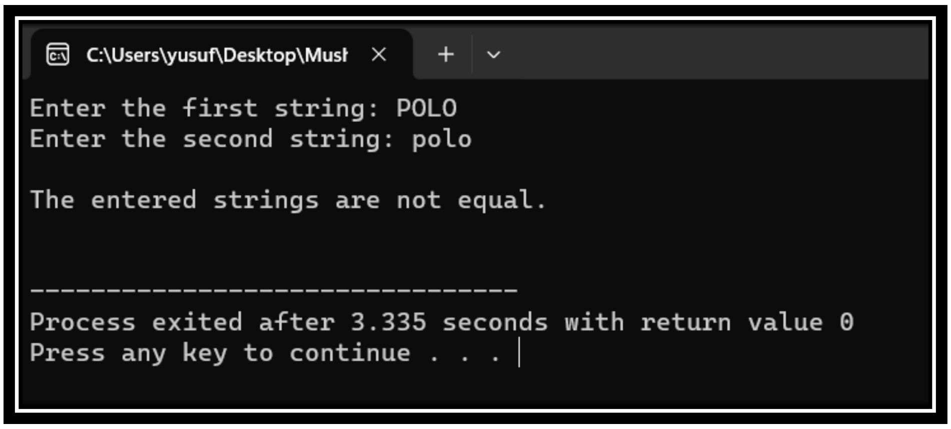
```
    string str1, str2;

    cout << "Enter the first string: ";
    getline (cin, str1);

    cout << "Enter the second string: ";
    getline (cin, str2);

    int x = str2.length();

    if (str1 == str2)
    {
        cout << endl << "The entered strings are equal.\n" << endl;

        for (int i = 0; i < x - 1; i++)
        {
            swap (str2[i], str2[i + 1]);
        }
        cout << "The second string after rotating becomes: " << str2;
    }

    else
    {
        cout << endl << "The entered strings are not equal.\n" << endl;
    }
```

<u>Body of the code</u>



```
Enter the first string: POLO
Enter the second string: POLO

The entered strings are equal.

The second string after rotating becomes: OLOP
--------------------------------
Process exited after 292.6 seconds with return value 0
Press any key to continue . . .
```

Output if the entered strings are equal



```
Enter the first string: POLO
Enter the second string: polo

The entered strings are not equal.


--------------------------------
Process exited after 3.335 seconds with return value 0
Press any key to continue . . .
```

Output if the entered strings are not equal

EXPLANATION

This C++ program first prompts the user to input two strings. It then compares the two strings to check if they are equal. If the strings are equal, it means they have the same characters.

In that case, the program performs a rotation operation on the second string by shifting each character one position to the left. This is achieved using a loop that iterates through each character in the second string, swapping it with the next character. The loop runs until the second-to-last character in the string to avoid going beyond the string's bounds.

Finally, the program prints the modified second string after the rotation. If the entered strings are not equal, it informs the user that the strings are not equal.

In simple terms, if you input two equal strings, the program shows you the second string after rotating its characters to the left, and if the strings are not equal, it lets you know that they are different.

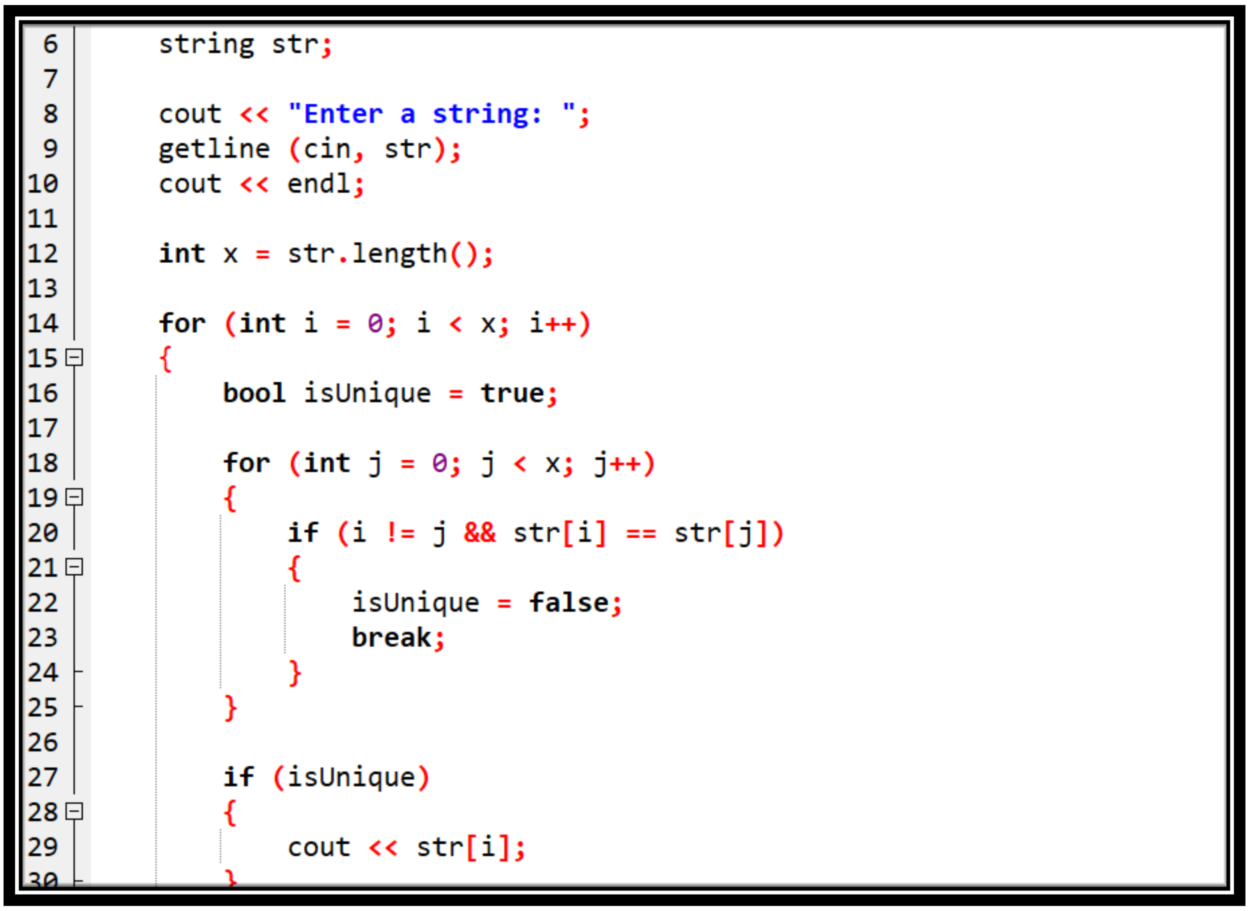
XXX-----------------------------------XXX

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.

<u>Answer</u>:

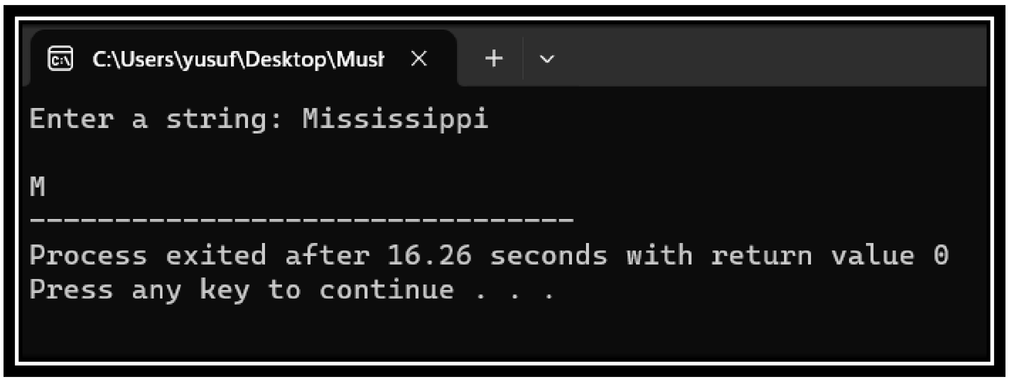
```
#include <iostream>
using namespace std;

int main()
{
        string str;
        cout << "Enter a string: ";
        getline (cin, str);
        cout << endl;
        int x = str.length();

        for (int i = 0; i < x; i++)
        {
                bool isUnique = true;

                for (int j = 0; j < x; j++)
                {
                        if (i != j && str[i] == str[j])
                        {
```

```
                isUnique = false;
                break;
            }
        }
        if (isUnique)
        {
            cout << str[i];
        }
    }
    return 0;
}
```



```
    string str;

    cout << "Enter a string: ";
    getline (cin, str);
    cout << endl;

    int x = str.length();

    for (int i = 0; i < x; i++)
    {
        bool isUnique = true;

        for (int j = 0; j < x; j++)
        {
            if (i != j && str[i] == str[j])
            {
                isUnique = false;
                break;
            }
        }

        if (isUnique)
        {
            cout << str[i];
        }
```

Body of the code



```
Enter a string: Mississippi

M
--------------------------------
Process exited after 16.26 seconds with return value 0
Press any key to continue . . .
```

Output of the code

EXPLANATION

In this C++ program, the user is prompted to input a string, and the program then proceeds to analyze and showcase the unique characters within that string. The uniqueness check is carried out using a nested loop structure.

The outer loop navigates through each character in the string, while the inner loop compares the current character with every other character in the string. If a match is found at a different position, indicating a duplicate, the program sets a flag, marking the character as non-unique. However, if the inner loop completes without finding any duplicates, the flag remains true, signifying that the character is unique. The program then prints these unique characters, creating an output string that excludes any repeated occurrences, ensuring that each character is represented only once in the final result.

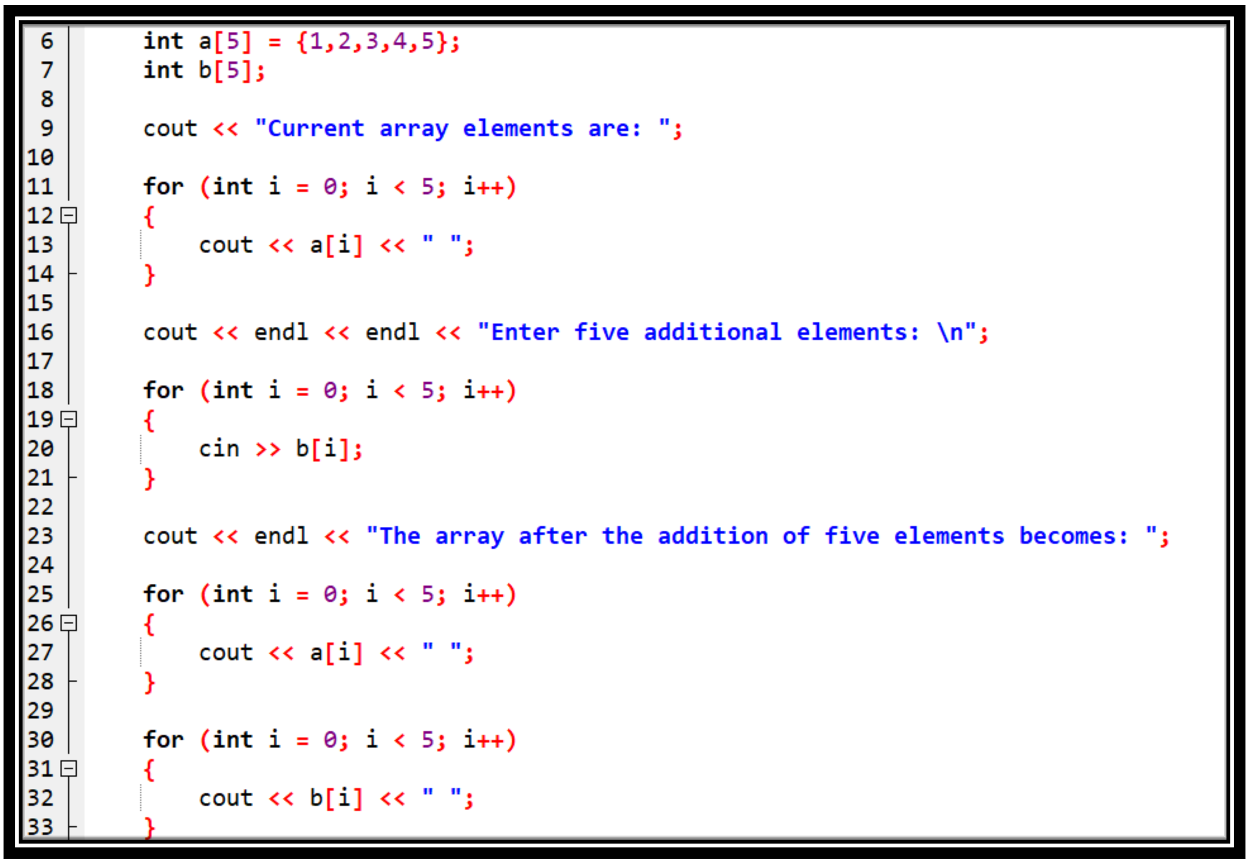
This approach provides a concise and effective way to extract and display the distinct elements from the user-provided string.

XXX----------------------------------XXX

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

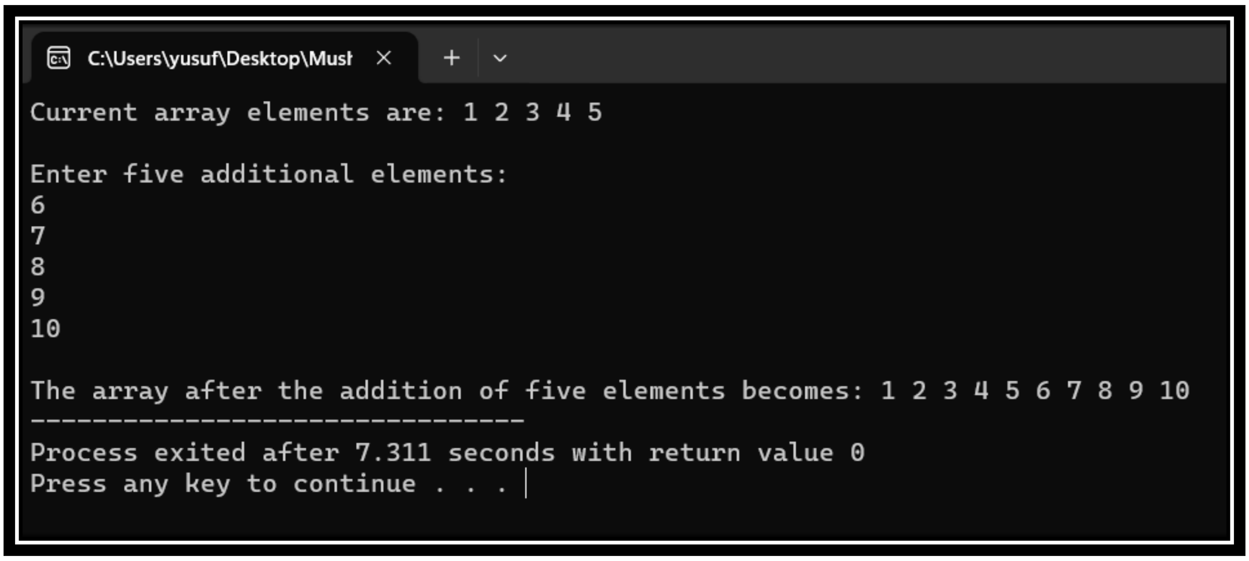
Answer:

```
#include <iostream>

using namespace std;


int main ()

{

        int a[5] = {1,2,3,4,5};

        int b[5];


        cout << "Current array elements are: ";


        for (int i = 0; i < 5; i++)

        {

                cout << a[i] << " ";

        }


        cout << endl << endl << "Enter five additional elements: \n";


        for (int i = 0; i < 5; i++)

        {

                cin >> b[i];

        }


        cout << "The array after the addition of five elements becomes: ";
        for (int i = 0; i < 5; i++)

        {

                cout << a[i] << " ";

        }


        for (int i = 0; i < 5; i++)

        {

                cout << b[i] << " ";

        }


        return 0;

}
```



```
    int a[5] = {1,2,3,4,5};
    int b[5];

    cout << "Current array elements are: ";

    for (int i = 0; i < 5; i++)
    {
        cout << a[i] << " ";
    }

    cout << endl << endl << "Enter five additional elements: \n";

    for (int i = 0; i < 5; i++)
    {
        cin >> b[i];
    }

    cout << endl << "The array after the addition of five elements becomes: ";

    for (int i = 0; i < 5; i++)
    {
        cout << a[i] << " ";
    }

    for (int i = 0; i < 5; i++)
    {
        cout << b[i] << " ";
    }
```

Body of the code



```
Current array elements are: 1 2 3 4 5

Enter five additional elements:
6
7
8
9
10

The array after the addition of five elements becomes: 1 2 3 4 5 6 7 8 9 10
--------------------------------
Process exited after 7.311 seconds with return value 0
Press any key to continue . . .
```

Output of the code

EXPLANATION

This C++ program begins by initializing an array, **"a"**, with five elements **(1, 2, 3, 4, and 5)** and displays its content. Subsequently, the user is prompted to input five additional elements into another array, **"b"**. The program then intends to exhibit the combined array after the addition of these new elements.

The subsequent objective of the program is to illustrate the outcome of combining the two arrays, incorporating the original elements from **"a"** and the newly input elements from **"b"**.

It is important to ensure that the display accurately reflects the combined array with updated values from both **"a"** and **"b"**, offering the user a complete view of the modified array composition.

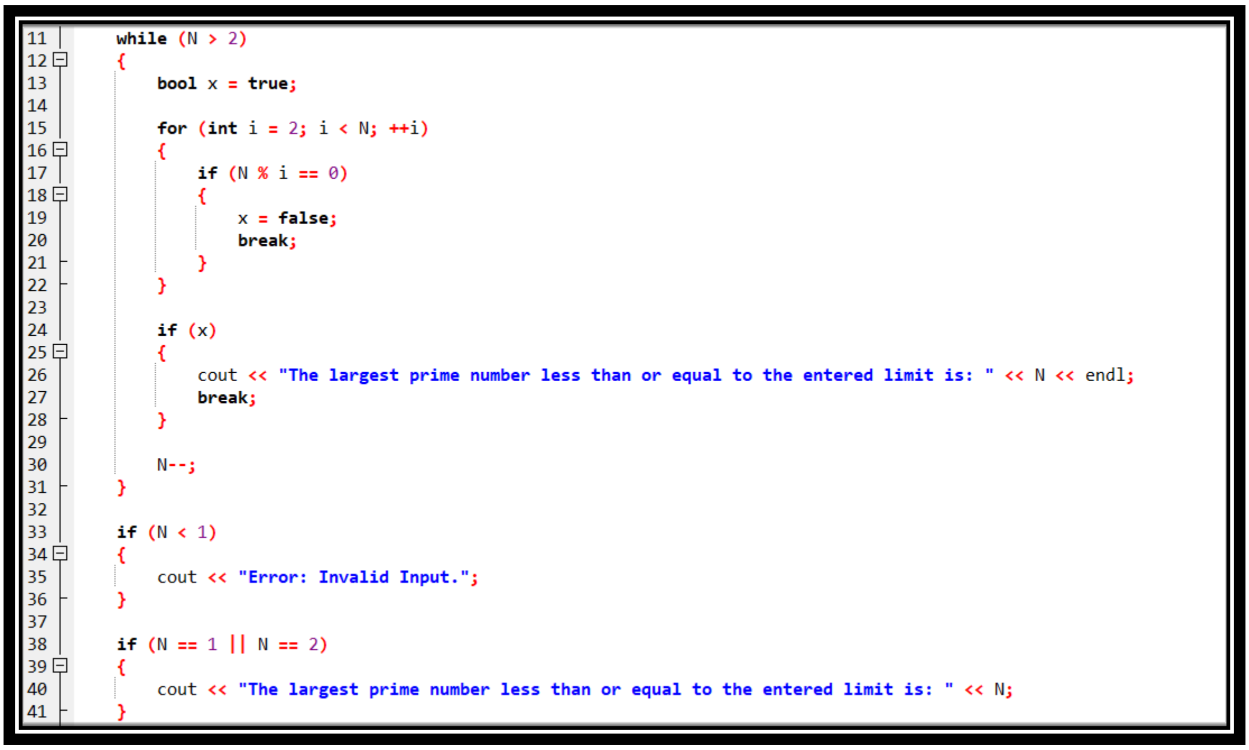
XXX-----------------------------------XXX

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.

Answer:

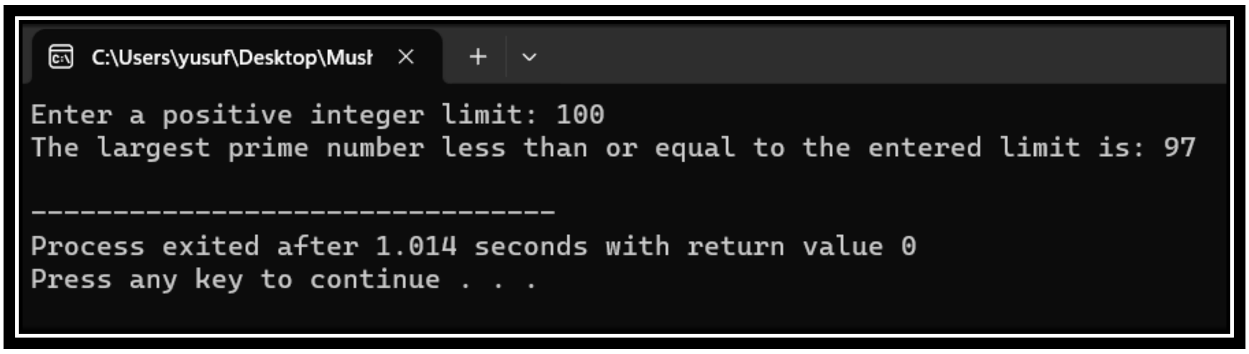
```
#include <iostream>

using namespace std;


int main ()

{

   int N;

   cout << "Enter a positive integer limit: ";

   cin >> N;


 while (N > 2)

   {

      bool x = true;


      for (int i = 2; i < N; ++i)

      {

         if (N % i == 0)

         {

            x = false;

            break;

         }

      }
```

```
        if (x)
        {
            cout << "The largest prime number less than or equal to the entered limit is: " << N
<< endl;
            break;
        }
        N--;
    }

    if (N < 1)
    {
            cout << "Error: Invalid Input.";
    }

    if (N == 1 || N == 2)
    {
            cout << "The largest prime number less than or equal to the entered limit is: " << N;
    }

    return 0;
    }
```



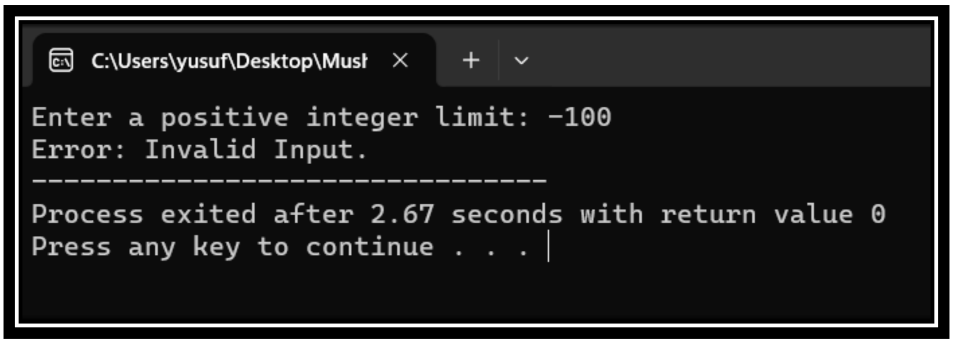
```cpp
    while (N > 2)
    {
        bool x = true;

        for (int i = 2; i < N; ++i)
        {
            if (N % i == 0)
            {
                x = false;
                break;
            }
        }

        if (x)
        {
            cout << "The largest prime number less than or equal to the entered limit is: " << N << endl;
            break;
        }

        N--;
    }

    if (N < 1)
    {
        cout << "Error: Invalid Input.";
    }

    if (N == 1 || N == 2)
    {
        cout << "The largest prime number less than or equal to the entered limit is: " << N;
    }
```

Body of the code



```
Enter a positive integer limit: 100
The largest prime number less than or equal to the entered limit is: 97

--------------------------------
Process exited after 1.014 seconds with return value 0
Press any key to continue . . .
```

Output of the code if the input is a greater than "1"



```
Enter a positive integer limit: -100
Error: Invalid Input.
--------------------------------
Process exited after 2.67 seconds with return value 0
Press any key to continue . . .
```

Output of the code if the input is a less than "1"

EXPLANATION

In this C++ program, the user is prompted to provide a positive integer limit, denoted as **“N”**. The program then employs a while loop to iteratively search for the largest prime number less than or equal to the entered limit.

Within the loop, a nested for loop checks the primality of each candidate by dividing **“N”** by numbers from 2 to **“N – 1”**. If a divisor is found, a **boolean** variable **“x”** is set to false, indicating that **“N”**` is not a prime number. The program continues this process, decrementing **“N”** in each iteration, until it identifies the largest prime number or until the limit becomes 2.

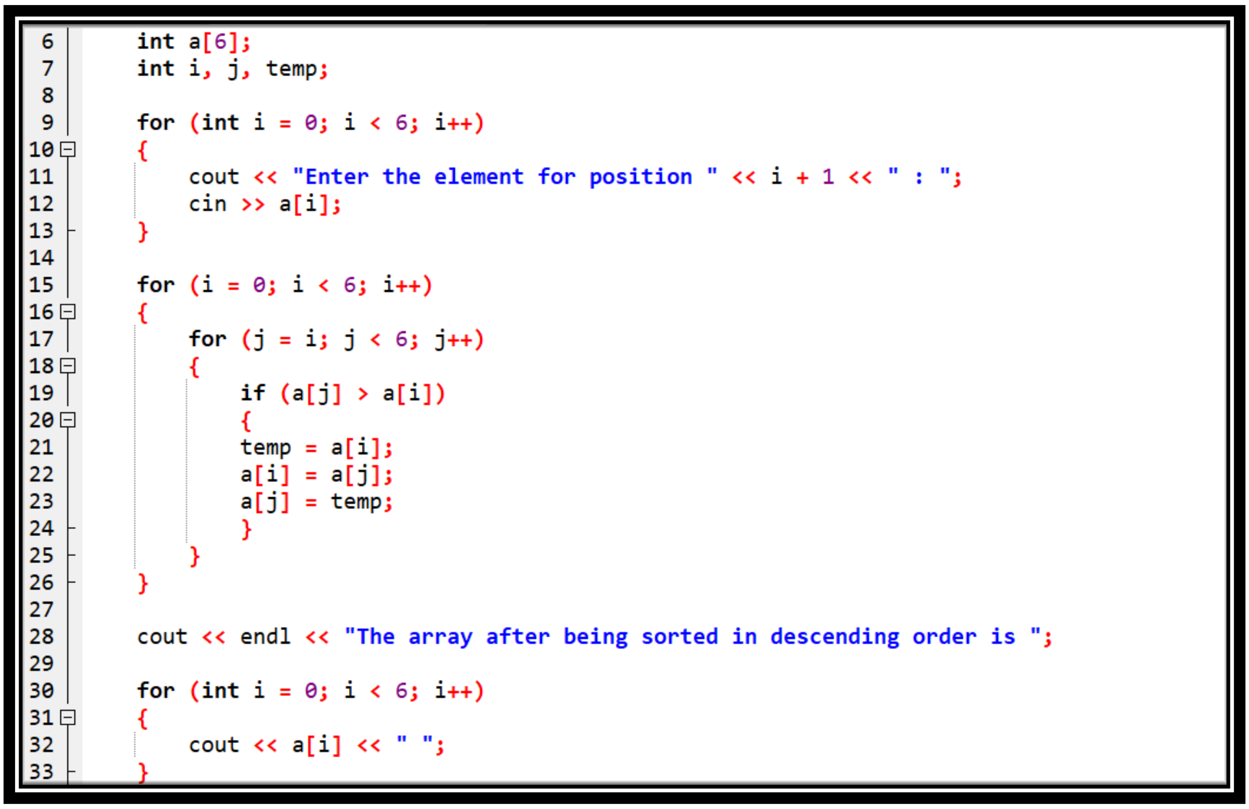
Special conditions handle scenarios where **“N”** is less than **1**, prompting an error message for invalid input, or when **“N”** is equal to **1** or **2**, directly printing the result. This program thus provides a systematic approach to finding the largest prime number within a user-specified range while incorporating error handling for a more robust user experience.

XXX-----------------------------------XXX

5. Implement Bubble Sort on an array of 6 integers.

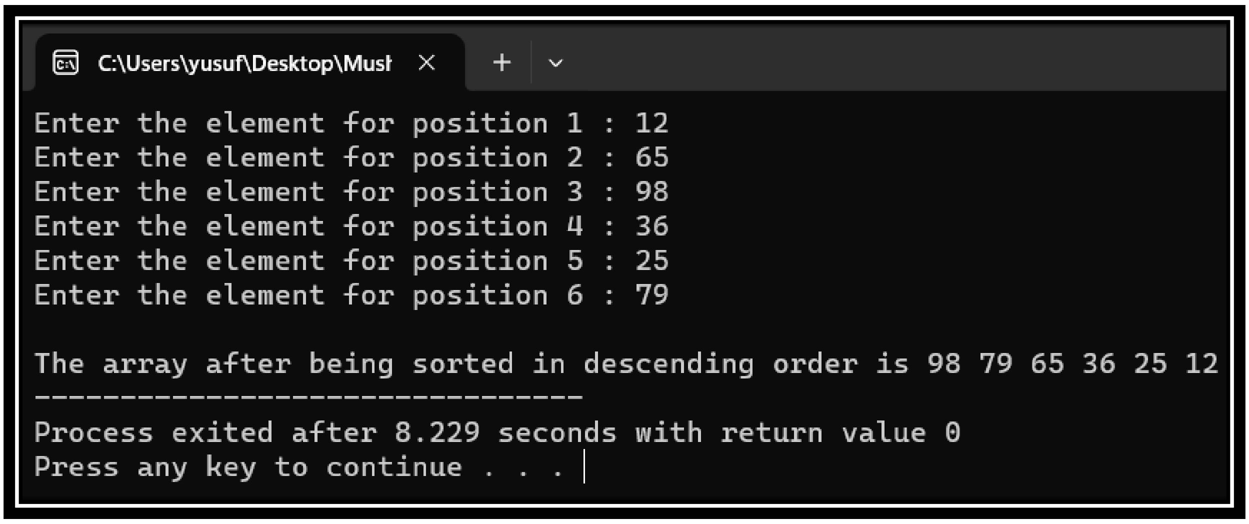
Answer:

```
#include <iostream>

using namespace std;


int main()

{

        int a[6];

        int i, j, temp;


        for (int i = 0; i < 6; i++)

        {
                cout << "Enter the element for position " << i + 1 << " : ";
                cin >> a[i];
        }
        for (i = 0; i < 6; i++)
        {
                for (j = i; j < 6; j++)
                {
                        if (a[j] > a[i])
                        {
                        temp = a[i];
                        a[i] = a[j];
                        a[j] = temp;
                        }
                }
        }

        cout << endl << "The array after being sorted in descending order is ";

        for (int i = 0; i < 6; i++)
        {
                cout << a[i] << " ";
        }

        return 0;
}
```



```cpp
    int a[6];
    int i, j, temp;

    for (int i = 0; i < 6; i++)
    {
        cout << "Enter the element for position " << i + 1 << " : ";
        cin >> a[i];
    }

    for (i = 0; i < 6; i++)
    {
        for (j = i; j < 6; j++)
        {
            if (a[j] > a[i])
            {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
            }
        }
    }

    cout << endl << "The array after being sorted in descending order is ";

    for (int i = 0; i < 6; i++)
    {
        cout << a[i] << " ";
    }
```

Body of the code



```
Enter the element for position 1 : 12
Enter the element for position 2 : 65
Enter the element for position 3 : 98
Enter the element for position 4 : 36
Enter the element for position 5 : 25
Enter the element for position 6 : 79

The array after being sorted in descending order is 98 79 65 36 25 12
--------------------------------
Process exited after 8.229 seconds with return value 0
Press any key to continue . . .
```

Output of the code

EXPLANATION

In this C++ program, the user engages in the interactive process of providing six numerical elements, creating an array named **“a”**. The program then employs the widely-known bubble sort algorithm to organize these elements in descending order.

The algorithm operates through multiple passes over the array, systematically comparing adjacent elements and swapping them if they are out of order. This iterative process repeats until the entire array is sorted, with the largest elements gradually **“bubbling up”** to their correct positions. The program facilitates this sorting mechanism using nested loops, where the outer loop governs the overall passes, and the inner loop manages the pairwise comparisons and swaps.

Once the sorting is complete, the program showcases the rearranged array, now sorted in descending order, offering users a visual representation of the step-by-step evolution of the bubble sort algorithm. Notably, the simplicity and straightforward nature of the bubble sort make it an accessible choice for educational purposes and smaller datasets.

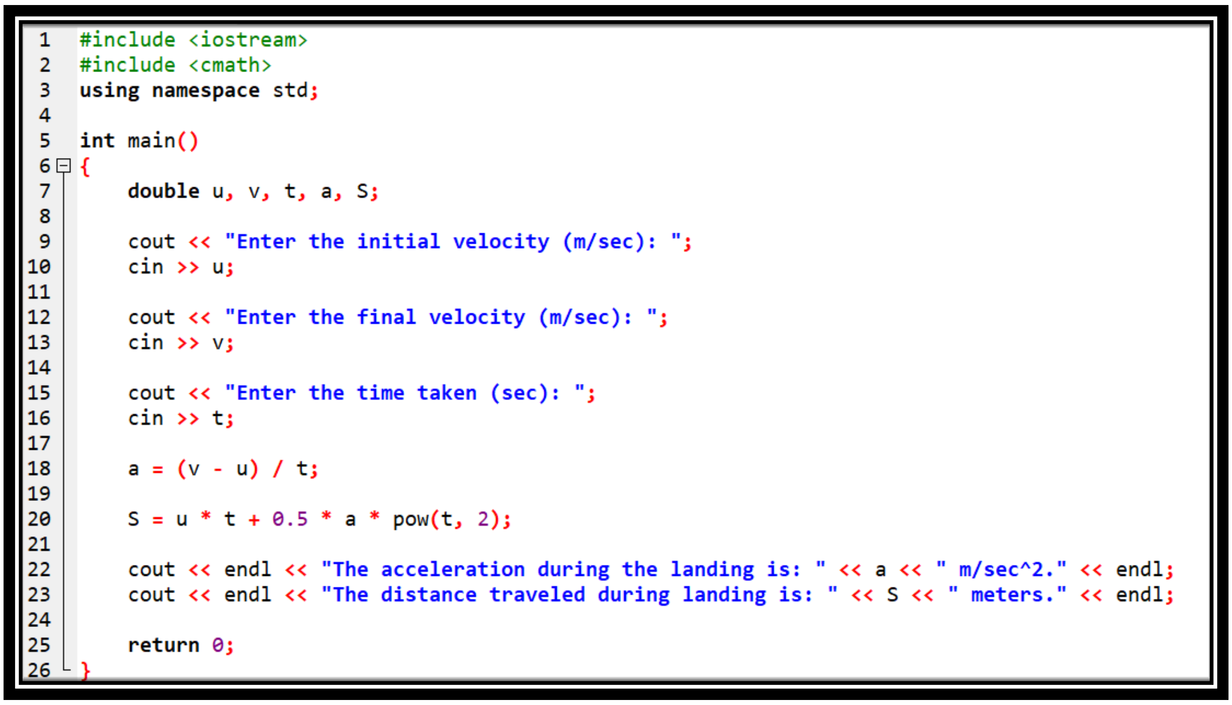
XXX------------------------------------XXX

6. Solve any Aerospace/Real Life Problem using C++ Programming.

Answer:

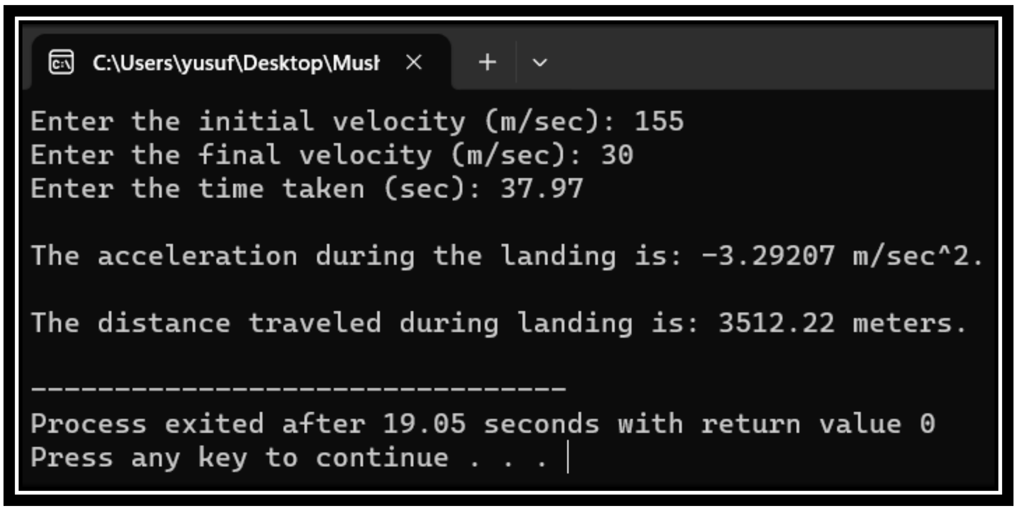
```
#include <iostream>

#include <cmath>

using namespace std;


int main()

{

   double u, v, t, a, S;

   cout << "Enter the initial velocity (m/sec): ";

   cin >> u;
```

```
    cout << "Enter the final velocity (m/sec): ";

    cin >> v;


    cout << "Enter the time taken (sec): ";

    cin >> t;


    a = (v - u) / t;

    S = u * t + 0.5 * a * pow(t, 2);


    cout << endl << "The acceleration during the landing is: " << a << " m/sec^2." << endl;

    cout << endl << "The distance traveled during landing is: " << S << " meters." << endl;


    return 0;

}
```



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

int main()
{
    double u, v, t, a, S;

    cout << "Enter the initial velocity (m/sec): ";
    cin >> u;

    cout << "Enter the final velocity (m/sec): ";
    cin >> v;

    cout << "Enter the time taken (sec): ";
    cin >> t;

    a = (v - u) / t;

    S = u * t + 0.5 * a * pow(t, 2);

    cout << endl << "The acceleration during the landing is: " << a << " m/sec^2." << endl;
    cout << endl << "The distance traveled during landing is: " << S << " meters." << endl;

    return 0;
}
```

Body of the code



C:\Users\yusuf\Desktop\Musł

```
Enter the initial velocity (m/sec): 155
Enter the final velocity (m/sec): 30
Enter the time taken (sec): 37.97

The acceleration during the landing is: -3.29207 m/sec^2.

The distance traveled during landing is: 3512.22 meters.

--------------------------------
Process exited after 19.05 seconds with return value 0
Press any key to continue . . .
```

Output of the code

EXPLANATION

In this C++ program, the user is engaged in a dynamic interaction where they input specific parameters related to the landing of an object, presumably an aircraft.

The program prompts the user to provide the initial velocity, final velocity, and time taken during the landing process. Utilizing these inputs, the program employs fundamental kinematic equations to calculate both the deceleration (negative acceleration) **"a"** experienced during the landing and the distance **"S"** covered by the object.

The calculated values are then presented to the user, offering a tangible representation of the physics involved in a simplified landing scenario. This program serves as an accessible introduction to the application of basic physics principles, making it a valuable learning tool for those exploring kinematics in the context of aerospace or other related fields.

XXX-----------------------------------XXX