Aim

Implement a class string containing the following functions:

1. Overload + operator to carry out the concatenation of strings.
2. Overload = operator to carry out string copy.
3. Overload <= operator to carry out the comparison of strings.
4. Function to display the length of string.
5. Function tolower() to convert upper case to lower case.
6. Function toupper() to convert lower case letters to upper case.

Experiment - 22

Object Oriented Programming Lab

Syeda Reeha Quasar

14114802719

4C7

# **EXPERIMENT – 22**

## **Aim:**

Implement a class string containing the following functions:

1. Overload + operator to carry out the concatenation of strings.
2. Overload = operator to carry out string copy.
3. Overload <= operator to carry out the comparison of strings.
4. Function to display the length of string.
5. Function tolower() to convert upper case to lower case.
6. Function toupper() to convert lower case letters to upper case.

## **Overload + operator to carry out the concatenation of strings.**

## **Source Code:**

#include <iostream>

#include <cstring>

using namespace std;

// concatenating 2 strings

class concatString{

    public:

        char str[100]; // class object

        concatString(){} // no parameter constructor

        // initalising class variable

        concatString(char s[]){

            strcpy(this->str, s);

        }

        // overloading operator+ for concatenation

        concatString operator+(concatString& s2){

            concatString s3;

            strcat(this->str, s2.str);

            strcpy(s3.str, this->str);

            return s3;

        }

};

int main(){

    char s1[100], s2[100];

    cout << "Enter 2 strings u want to concatenate" << endl;

    cin >> s1 >> s2;

    concatString a1(s1);

    concatString a2(s2);

    concatString a3;

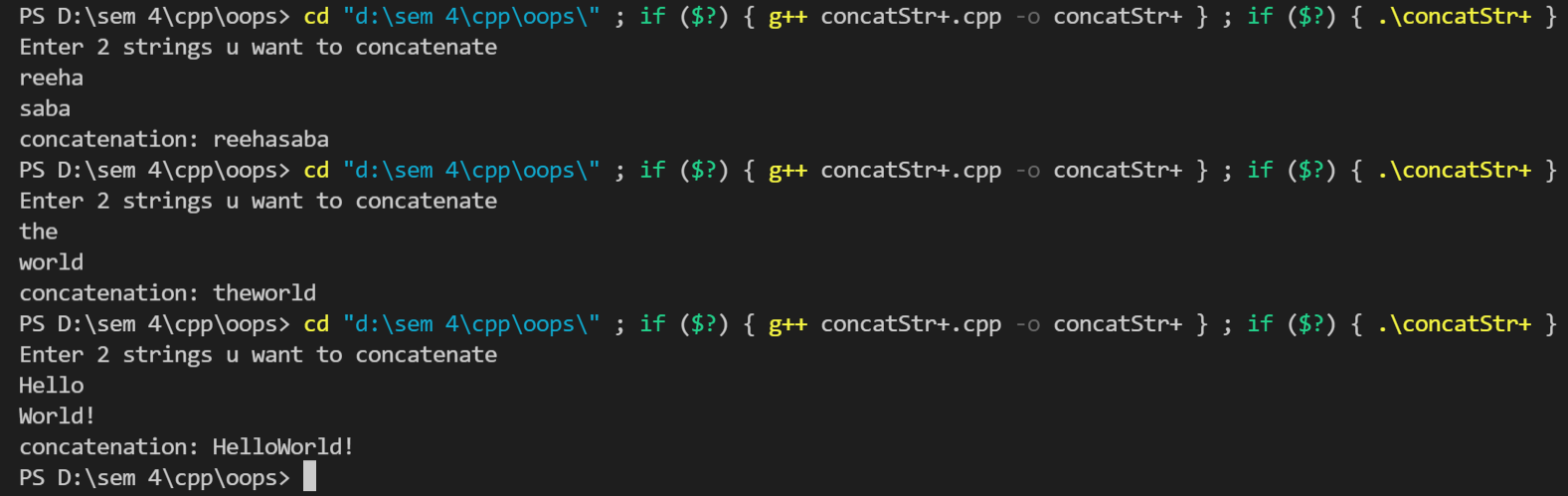
    a3 = a1 + a2;

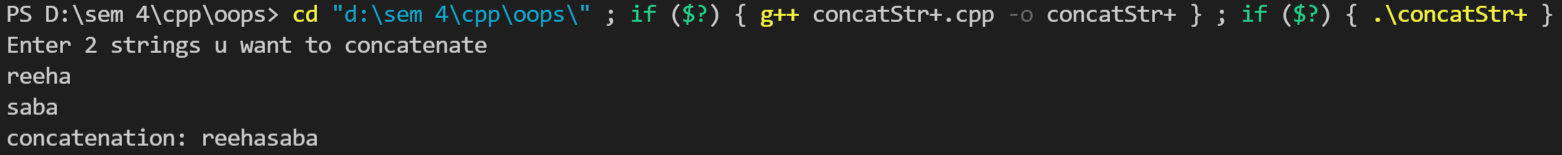
    cout << "concatenation: " << a3.str;

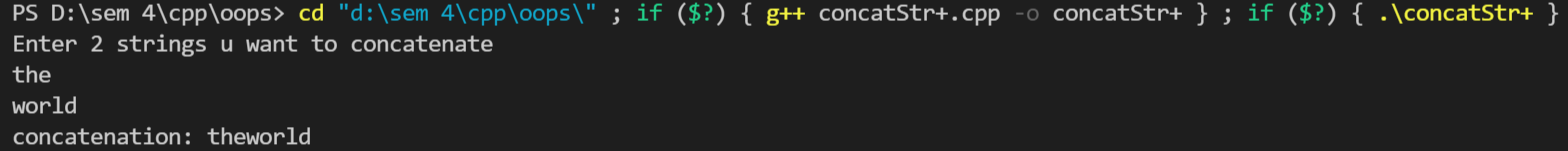
    return 0;

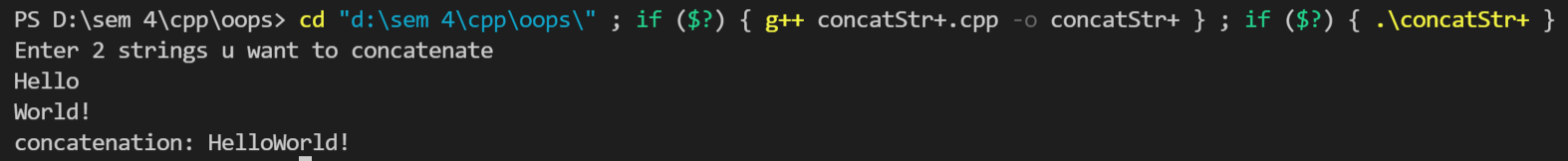
}

## **Output:**









## **Overload = operator to carry out string copy**

## **Source Code:**

#include <iostream>

#include <cstring>

using namespace std;

class equalString{

    public:

        char str[25]; // Classes object of string

        equalString(){} // no parameter constructor

        // Parametrized Constructor

        equalString(char s[]){

            strcpy(this->str, s); // Initialize the string to class object

        }

        bool operator==(equalString s2)

        {

            if (strcmp(str, s2.str) == 0)

                return true;

            else

                return false;

        }

};

int main(){

    char s1[100], s2[100];

    cout << "Enter 2 strings u want to compare and check equal or not" << endl;

    cin >> s1 >> s2;

    equalString a1(s1);

    equalString a2(s2);

    if (a1 == a2) {

        cout << "Strings are equal" << endl;

    }

    else{

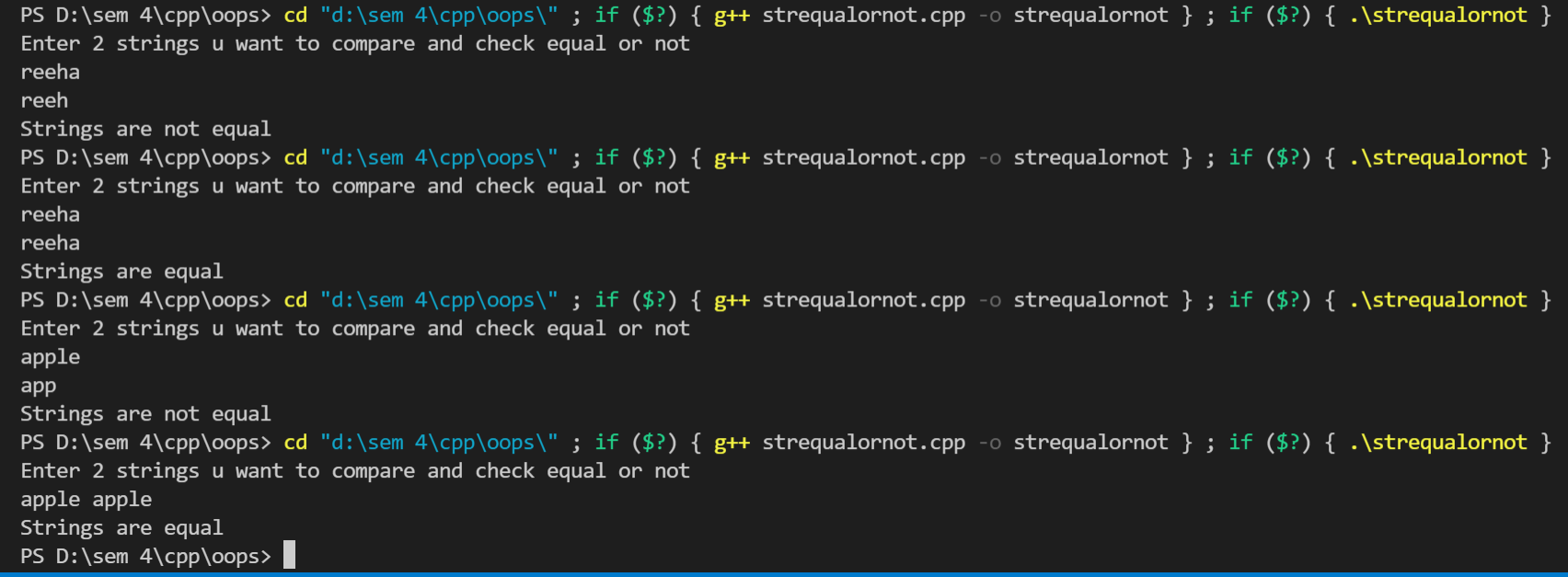
        cout << "Strings are not equal" << endl;

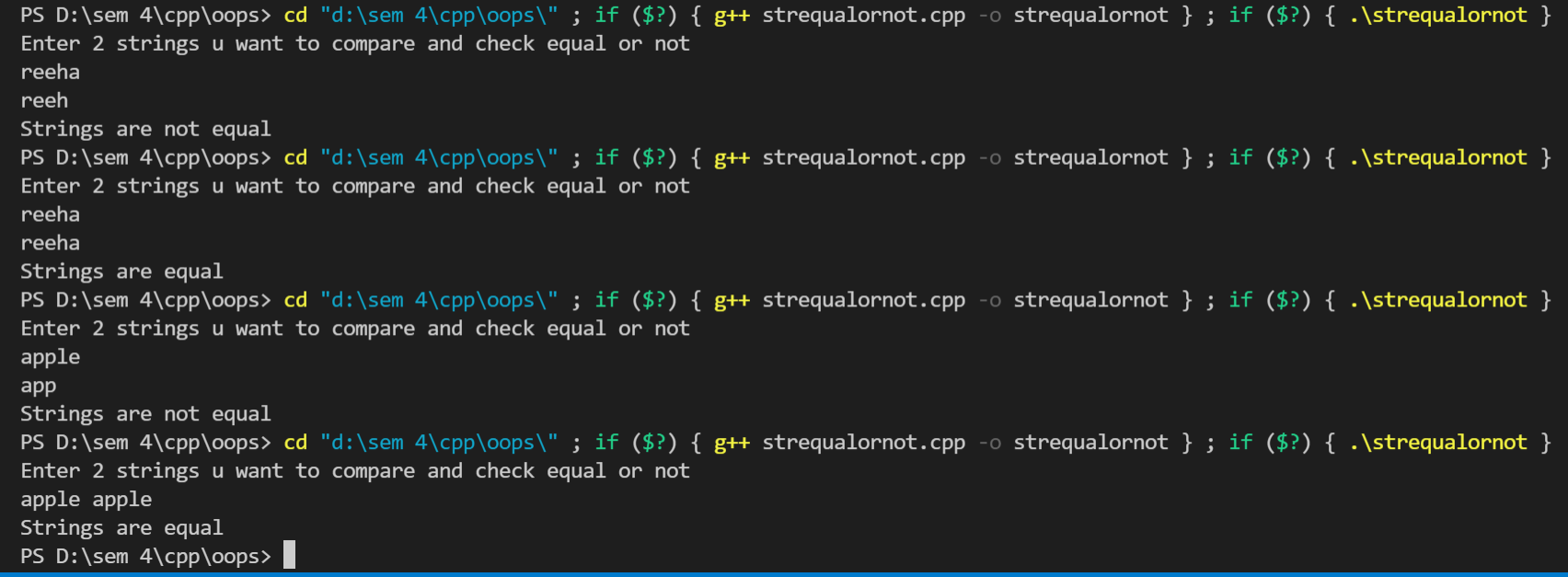
    }

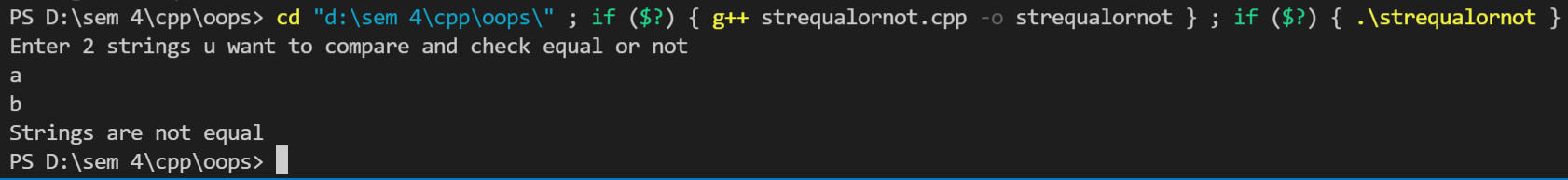
    return 0;

}

## **Output:**







## **Overload <= operator to carry out the comparison of strings.**

## **Source Code:**

#include <cstring>

#include <iostream>

#include <string.h>

using namespace std;

class equalString{

    public:

        char str[25]; // Classes object of string

        equalString(){} // no parameter constructor

        // Parametrized Constructor

        equalString(char s[]){

            strcpy(this->str, s); // Initialize the string to class object

        }

        bool operator<=(equalString s2)

        {

            if (strlen(str) <= strlen(s2.str))

                return true;

            else

                return false;

        }

};

int main(){

    char s1[100], s2[100];

    cout << "Enter 2 strings u want to compare and check equal or not" << endl;

    cin >> s1 >> s2;

    equalString a1(s1);

    equalString a2(s2);

    if (a1 <= a2) {

        cout << "First string is smaller than or equal to second " << endl;

    }

    else{

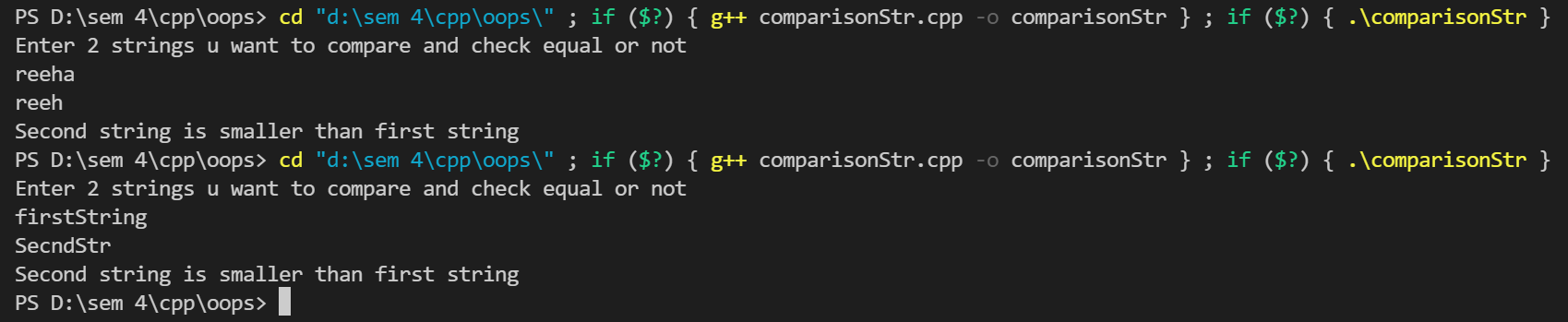
        cout << "Second string is smaller than first string" << endl;

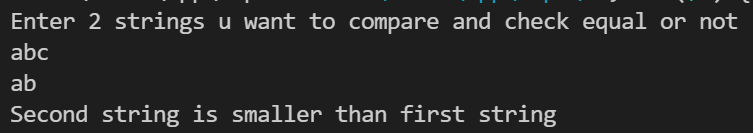
    }

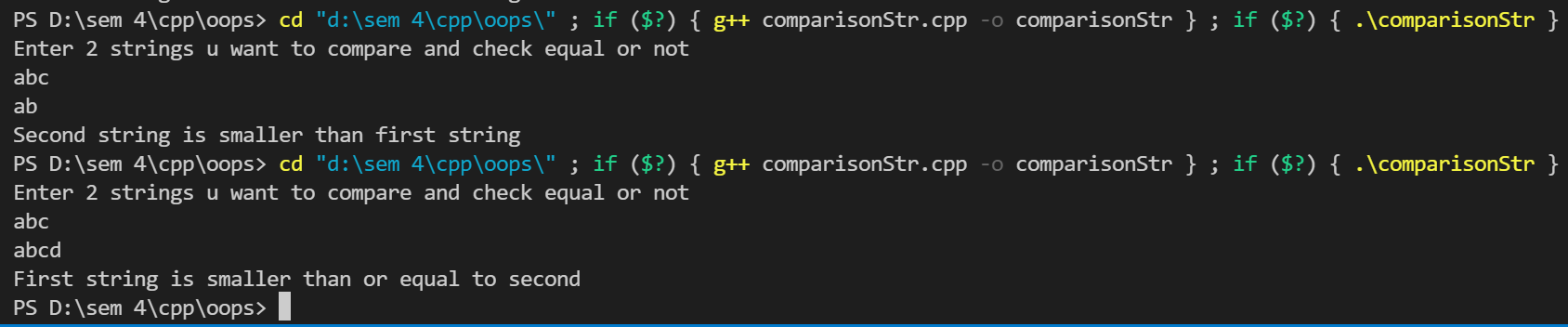
    return 0;

}

## **Output:**







## **Function to display the length of string.**

# **Source code**

#include <iostream>

#include <cstring>

using namespace std;

class strLength

{

public:

    char str[25]; // Classes object of string

    void strlen()

    {

        cout << "Enter strings u want to find length of:" << endl;

        cin >> str;

        int l = 0;

        for (int i = 0; str[i] != '\0'; i++)

        {

            l++;

        }

        cout << "\n The length of String is: " << l;

    }

};

int main()

{

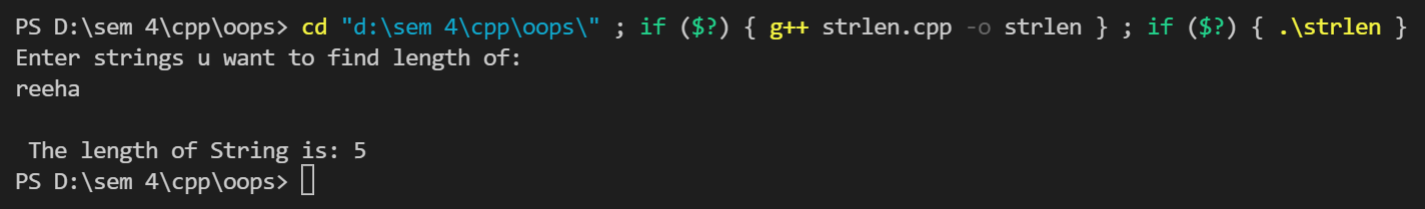
    strLength obj;

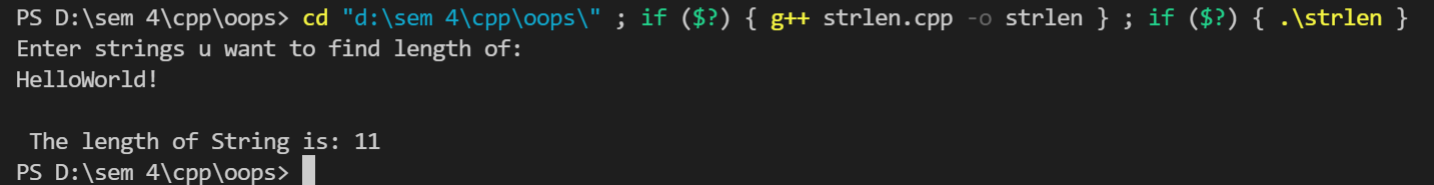
    obj.strlen();

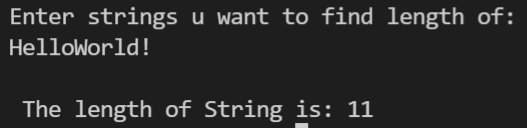
    return 0;

}

## **Output:**







## **Function tolower() to convert upper case to lower case.**

# **Source code**

#include <iostream>

#include <cstring>

using namespace std;

class caseChange

{

public:

    char str[25]; // Classes object of string

    void tolower()

    {

        cout << "Enter strings u want to lower:" << endl;

        cin >> str;

        for (int i = 0; str[i] != '\0'; i++)

        {

            if ((str[i] >= 65) && (str[i] <= 90))

            {

                str[i] = str[i] + 32;

            }

        }

        cout << "Lowered final string is " << str;

    }

};

int main()

{

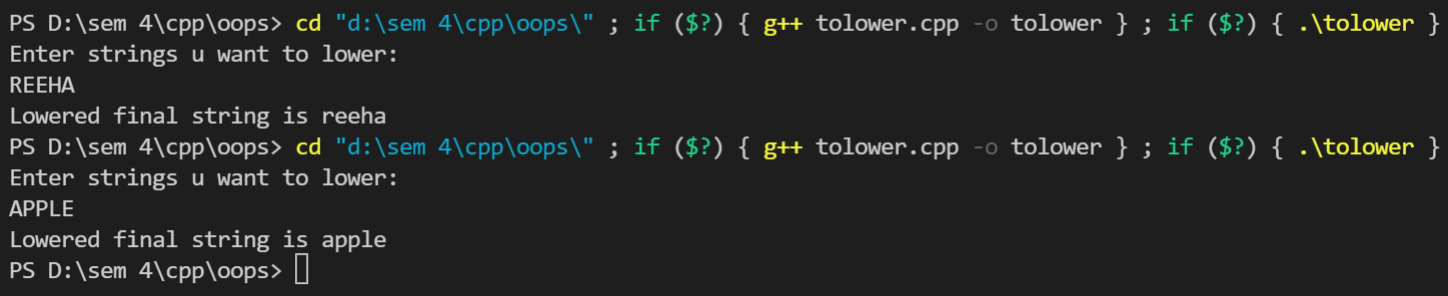
    caseChange obj;

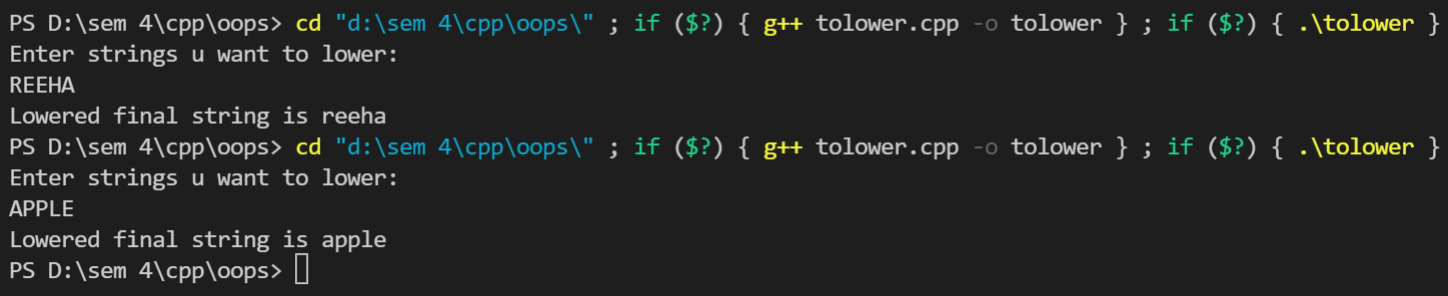
    obj.tolower();

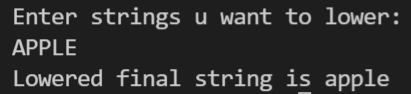
    return 0;

}

## **Output:**







## **Function toupper() to convert lower case letters to upper case.**

# **Source code**

#include <iostream>

#include <cstring>

using namespace std;

class caseChange

{

public:

    char str[25]; // Classes object of string

    void toupper()

    {

        cout << "Enter strings u want to uppercase:" << endl;

        cin >> str;

        for (int i = 0; str[i] != '\0'; i++)

        {

            if ((str[i] >= 97) && (str[i] <= 122))

            {

                str[i] = str[i] - 32;

            }

        }

        cout << "Uppered/capitalized final string is " << str;

    }

};

int main()

{

    caseChange obj;

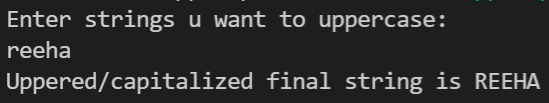
    obj.toupper();

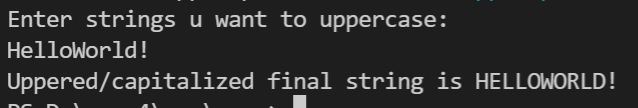
    return 0;

}

## **Output:**







# **Source code**

#include <cstring>

#include <iostream>

#include <string.h>

using namespace std;

class operate

{

    char str[25]; // Classes object of string

public:

    void operator+(char s[25])

    {

        strcat(str, " ");

        strcat(str, s);

    }

    void operator=(char s[25])

    {

        strcpy(str, s);

    }

    void operator<=(char s[25])

    {

        if (strcmp(str, s) > 0)

        {

            cout << str << " is larger than " << s;

        }

        else if (strcmp(str, s) < 0)

        {

            cout << s << " is larger than " << str;

        }

        else

        {

            cout << str << " is equal to " << s;

        }

    }

    void strlen()

    {

        int l = 0;

        for (int i = 0; str[i] != '\0'; i++)

        {

            l++;

        }

        cout << "\n The length of String is: " << l;

    }

    void toupper()

    {

        for (int i = 0; str[i] != '\0'; i++)

        {

            if ((str[i] >= 97) && (str[i] <= 122))

            {

                str[i] = str[i] - 32;

            }

        }

    }

    void tolower()

    {

        for (int i = 0; str[i] != '\0'; i++)

        {

            if ((str[i] >= 65) && (str[i] <= 90))

            {

                str[i] = str[i] + 32;

            }

        }

    }

    void display()

    {

        cout << "Final string -----------> " << str << endl;

    }

};

int main()

{

    char s[100], st[100], ch;

    cout << "Enter 2 strings u want to compare and check equal or not" << endl;

    cin >> s >> st;

    operate r;

    cout << "\n\nChoose whether u want to \n1)string length \nconcat'+' \n2)to Upper \ncompare'<' \n3) to lower \ncopy'=' \nexit(x)\n\n";

    //Invoking different operators

    cin >> ch;

    while (ch != 'x')

    {

        switch (ch)

        {

        case '+':

            r + st;

            r.display();

            break;

        case '=':

            r = st;

            r.display();

            break;

        case '<':

            r <= st;

            break;

        case 1:

            r.strlen();

            break;

        case 2:

            r.toupper();

            r.display();

            break;

        case 3:

            r.tolower();

            r.display();

            break;

        case 'x':

            cout << " Exit";

            exit(0);

        }

        cout << "\n\nChoose whether u want to \n1)string length \nconcat'+' \n2)to Upper \ncompare'<' \n3) to lower \ncopy'=' \nexit(x)\n\n";

        //Invoking different operators

        cin >> ch;

    }

    return 0;

}





# **Viva Questions**

#### Q1) What is overloading?

Ans.

#### C++ allows you to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.

#### An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

#### When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.

Q2) What are causes of function overloading?

Ans.

When the compiler is unable to decide which function is to be invoked among the overloaded function, this situation is known as **function overloading**.

When the compiler shows the ambiguity error, the compiler does not run the program.

**Causes of Function Overloading:**

* Type Conversion.
* Function with default arguments.
* Function with pass by reference.

