**Question 1)** Write a program for multiplication of two matrices using OOP.

**Code ::**

*/\**

*Write a program for multiplication of two matrices using OOP.*

*\*/*

*#include*<iostream>

using namespace std;

class matrix

{

    int a[10][10];

    int m,n;

    public:

    void input();

    void output();

    void multiply(matrix,matrix);

};

void matrix::input(){

    cout<<"Enter the number of row : ";

    cin>>m;

    cout<<"Enter the number of column : ";

    cin>>n;

    cout<<"Matrix"<<"\n";

*for*(int i=0;i<m;i++){

*for*(int j=0;j<n;j++){

            cin>>a[i][j];

        }

    }

}

void matrix :: output()

{

*for*(int i=0;i<m;i++){

        cout<<"\n";

*for*(int j=0;j<n;j++){

            cout<<a[i][j]<<"\t";

        }

    }

}

void matrix :: multiply(matrix m1, matrix m2){

*if*(m1.n!=m2.m) cout<<"matrix multiplication is not possible";

*else*{

*for*(int i=0;i<m1.m;i++){

*for*(int j=0;j<m2.n;j++){

                a[i][j]=0;

*for*(int k=0;k<m1.n;k++){

                    a[i][j]=a[i][j] +( m1.a[i][k]\*m2.a[k][j]);

                    m=m1.m;

                    n=m2.n;

                }

            }

        }

    }

}

int main()

{

    cout << "asdf";

    matrix m1,m2,m3;

    m1.input();

    m2.input();

    m3.multiply(m1,m2);

    m3.output();

    cout << "\n\n";

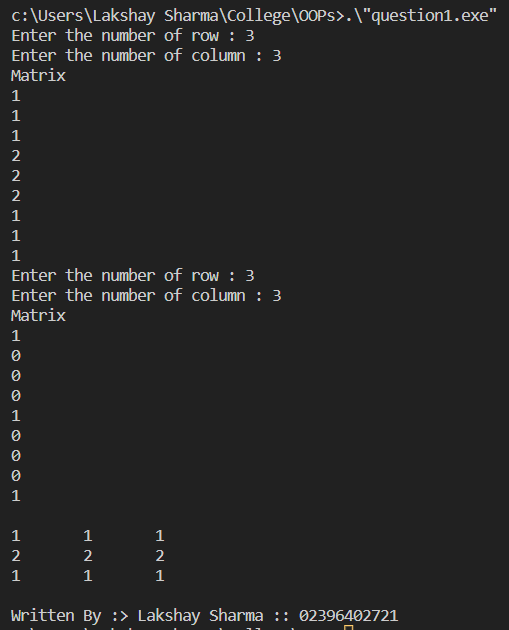
    cout << "Written By :> Lakshay Sharma :: 02396402721";

*return* 0;

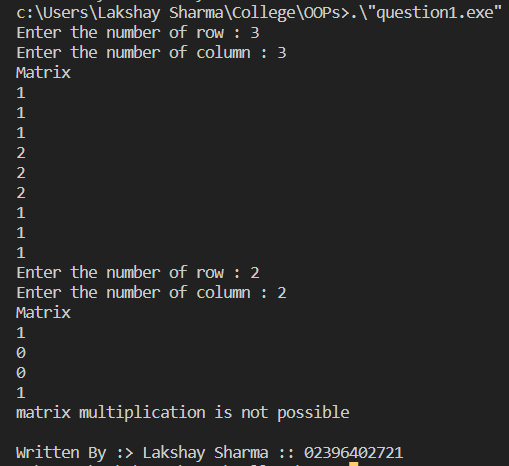
}

**Output ::**

When size of matrices matches



When size of matrices do not match



**Question 2)** Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two argument is used to initialized real and imag to two different values.

**Code ::**

*/\**

*Write a program to perform addition of two complex numbers using constructor overloading. The* *first constructor which takes no argument is used to create objects which are not initialized, second* *which takes one argument is used to initialize real and imag parts to equal values and third which* *takes two argument is used to initialized real and imag to two different values.*

*\*/*

*#include* <iostream>

using namespace std;

class Complex {

    public:

        int real;

        int imaginary;

        Complex(){ *//DefaultConstructor*

        real = 0;

        imaginary = 0;

        }

        Complex(int a){ *// Single Parametrized Constructor*

        real = a;

        imaginary = a;

        }

        Complex(int r, int i){ *// Double Parametrized Constructor*

        real = r;

        imaginary = i;

        }

        Complex addComplexNumber(Complex C1, Complex C2){

*/\* Member Function to perform addition \*/*

            Complex res;

            res.real = C1.real + C2.real;

            res.imaginary = C1.imaginary + C2.imaginary;

*return* res;

        }

    };

int main(){

    Complex C1(3);

    cout << "Complex number 1 : " << C1.real<< " + i" << C1.imaginary << endl;

    Complex C2(11,13);

    cout << "Complex number 2 : " << C2.real << " + i" << C2.imaginary << endl;

    Complex C3;

    C3 = C3.addComplexNumber(C1, C2);

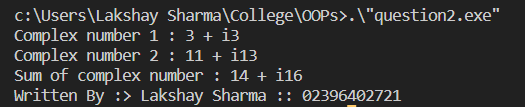
    cout << "Sum of complex number : "<< C3.real << " + i"<< C3.imaginary << endl;

    cout << "Lakshay Sharma :: 02396402721";

*return* 0;

}

**Output ::**

****

**Question 3)** Write a program to find the greatest of two given numbers in two different classes using friend function.

**Code ::**

*/\**

*Write a program to find the greatest of two given numbers in two different classes using friend* *function.*

*\*/*

*#include* <iostream>

using namespace std;

    class a;

    class b{

        private:

            int number;

        public:

            b(int x){

                number=x;

            }

        void friend greatest(a a1,  b b1);

    };

    class a{

        private:

            int number;

        public:

            a(int x){

                number=x;

            }

        void friend greatest(a a1,  b b1);

    };

    void greatest(a a1,b b1) {

        cout <<  "Greatest number is:" + to\_string((a1.number>b1.number)? a1.number : b1.number) << endl;

    }

int main() {

    int num;

    cout << "A:";   cin >> num;a a1(num);

    cout << "B:";   cin >> num;b b1(num);

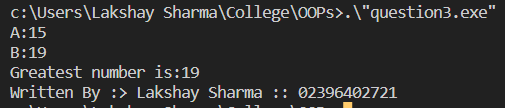
    greatest(a1,b1);

    cout << "Written By :> Lakshay Sharma :: 02396402721";

*return* 0;

}

**Output ::**

****

**Question 4)** Create a class Student, which have data members as name, branch, roll no., age, sex, five subjects. Display the name of student & his percentage who has more than 70%. Use array of objects. [Array of objects]

**Code ::**

*/\**

*Create a class Student, which have data members as name, branch, roll no., age, sex, five subjects. Display the name of student & his percentage who has more than 70%. Use array of objects. [Array of objects]*

*\*/*

*#include* <iostream>

*#include* <array>

*#include* <string>

class Student {

    public:

*// Data members*

        std::string name\_;

        std::string branch\_;

        int roll\_no\_;

        int age\_;

        char sex\_;

        std::array<int, 5> marks\_;

*// Constructor with initializer list to initialize data members*

        Student(std::string name, std::string branch, int roll\_no, int age, char sex, std::array<int, 5> marks){

            name\_ = name;

            branch\_ = branch;

            roll\_no\_ = roll\_no;

            sex\_ = sex;

            marks\_ = marks;

        }

*// Method to calculate the percentage of marks obtained by the student*

        double percentage(){

            int sum = 0;

*for* (auto mark : marks\_) {

                sum += mark;

            }

*return* sum / 5.0;

        }

};

int main() {

*// Create an array of objects of the Student class*

    std::array<Student, 5> students {

        Student("John", "Computer Science", 1, 18, 'M', {85, 92, 75, 91, 78}),

        Student("Jane", "Electrical Engineering", 2, 19, 'F', {70, 65, 80, 75, 78}),

        Student("Bob", "Mechanical Engineering", 3, 20, 'M', {65, 72, 68, 62, 60}),

        Student("Alice", "Electronics Engineering", 4, 21, 'F', {95, 98, 12, 94, 10}),

        Student("David", "Civil Engineering", 5, 22, 'M', {80, 75, 70, 85, 80})

    };

*// Loop through all the students and display the name of the student*

*// who has obtained more than 70% marks*

*for* (auto& student : students) {

*if* (student.percentage() > 70) {

            std::cout << student.name\_ << " has obtained more than 70% marks.\n";

        }

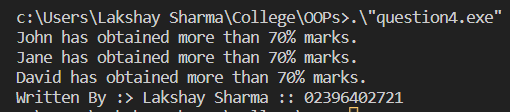
    }

    std::cout << "Written By :> Lakshay Sharma :: 02396402721";

*return* 0;

}

**Output ::**

****

**Question 5)** Implement a class string containing the following functions:

a. Overload + operator to carry out the concatenation of strings.

b. Overload = operator to carry out string copy.

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

d. Function to display the length of a string.

e. Function tolower( ) to convert upper case letters to lower case.

f. Function toupper( ) to convert lower case letters to upper case.

**Code ::**

*/\**

*Implement a class string containing the following functions:*

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

*b. Overload = operator to carry out string copy.*

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

*d. Function to display the length of a string.*

*e. Function tolower( ) to convert upper case letters to lower case.*

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

*\*/*

*#include* <iostream>

*#include* <algorithm>

*#include* <string>

class String {

    public:

*// Data member to store the string*

        std::string str\_;

*// Parametrized Constructor*

*// str\_ is the actual default string type*

        String(std::string str) : str\_(str) {}

*// Overload + operator to concatenate two strings*

        String operator+(String& other){

*return* String(str\_ + other.str\_);

        }

*// Overload = operator to copy a string*

        String& operator=(String& other){

            str\_ = other.str\_;

*return* \**this*;

        }

*// Overload <= operator to compare two strings*

        bool operator<=(String& other){

*return* str\_ <= other.str\_;

        }

*// Method to get the length of the string*

        int length() const {

*return* str\_.length();

        }

*// Method to convert the string to lower case (Using Algorithm header)*

        void tolower() {

            std::transform(str\_.begin(), str\_.end(), str\_.begin(), ::tolower);

        }

*// Method to convert the string to upper case (Using Algorithm header)*

        void toupper() {

            std::transform(str\_.begin(), str\_.end(), str\_.begin(), ::toupper);

        }

*// Overloaded << operator to output the string*

        friend std::ostream& operator<<(std::ostream& os, String& Str) {

*return* os << Str.str\_;

        }

};

int main() {

*// Create two string objects*

    String str1("Hello ");

    String str2("World");

    std::cout << "str1 = " << str1 << '\n';

    std::cout << "str2 = " << str2 << '\n';

*// Use the + operator to concatenate the strings*

    String str3 = str1 + str2;

    std::cout << "str3 (str1+str2) = " << str3 << '\n';

*// Use the = operator to copy the value of one string to another*

    str1 = str3;

    std::cout << "str1 (=str3) = " << str1 << '\n';

*// Use the <= operator to compare two strings*

    bool is\_smaller = (str1 <= str2);

    std::cout << "str1 is " << (is\_smaller ? "smaller" : "larger") << "than str2\n";

*// Use the length function to get number of characters in string*

    std::cout << "length of str1 = " << str1.length() << '\n';

*// Use of to\_lower() function in string*

    str1.tolower();

    std::cout << "to\_lower(str1) = " << str1 << '\n';

*// Use of to\_upper() function in string*

    str1.toupper();

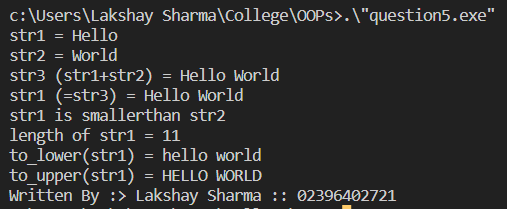
    std::cout << "to\_upper(str1) = " << str1 << '\n';

    std::cout << "Written By :> Lakshay Sharma :: 02396402721";

*return* 0;

}

**Output ::**

****

**Question 6)** Write a C++ program to define function power to raise a number m to a power n the function takes a double value for m. And integer value for n and return the result correctly use a default value of 2 for n to make the function calculate squares when this argument is omitted.

**Code ::**

*/\**

*Write a C++ program to define function power to raise a number m to a power n the function takes a double value for m. And integer value for n and return the result correctly use a default value of 2 for n to make the function calculate squares when this argument is omitted.*

*\*/*

*#include* <iostream>

*#include* <cmath>

*// Function to raise m to the power of n*

double power(double m, int n = 2) {

*return* std::pow(m, n);

}

int main() {

    std::cout << "power(3.0) = " << power(3.0) << '\n'; *// power(3.0) = 9*

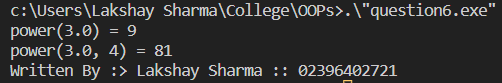
    std::cout << "power(3.0, 4) = " << power(3.0, 4) << '\n'; *// power(3.0, 4) = 81*

    std::cout << "Written By :> Lakshay Sharma :: 02396402721";

*return* 0;

}

**Output ::**

****

**Question 7)** Write a program to overload unary operator ++

**Code ::**

*/\**

*Write a program to overload unary operator ++.*

*\*/*

*#include* <iostream>

class Integer {

public:

*// Constructor to initialize the integer value*

  Integer(int value) : value\_(value) {}

*// Overloaded ++ operator to increment the integer value*

  Integer operator++() {

    ++value\_;

*return* \**this*;

  }

*// Overloaded << operator to output the integer value*

  friend std::ostream& operator<<(std::ostream& os, const Integer& obj) {

*return* os << obj.value\_;

  }

private:

*// The integer value*

  int value\_;

};

int main() {

*// Create an Integer object*

  Integer n(10);

*// Use the ++ operator to increment the integer value*

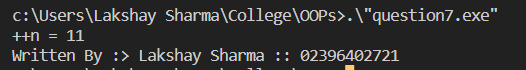
  std::cout << "++n = " << ++n << '\n'; *// ++n = 11*

  std::cout << "Written By :> Lakshay Sharma :: 02396402721";

*return* 0;

}

**Output ::**

****

**Question 8)**

**Code ::**