1. Add friend functions for the previous program. Write a C++ program to create a class called COMPLEX and implement the following overloading functions ADD that return a COMPLEX number.

        i. ADD (a, s2) – where a is an integer (real part) and s2 is a complex number.

        ii. ADD (s1, s2) – where s1 and s2 are complex numbers.

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

class Complex
{
    private:
        int x,y;
    public:
        Complex(int x_=0,int y_=0): x(x_), y(y_){}
        void get();
        friend Complex add(int a, Complex s2);
        friend Complex add(Complex s1, Complex s2);
        void print();
};

void Complex::get()
{
    cout<<"Enter the real part and imaginary part of a Complex number : ";
    cin>>x>>y;
```

```cpp
}

Complex add(int a, Complex s2)
{
        Complex c(s2.x + a, s2.y);
        return c;
}

Complex add(Complex s1, Complex s2)
{
        Complex c(s1.x + s2.x, s1.y + s2.y);
        return c;
}

void Complex::print()
{
        cout<<x<<" + i"<<y<<endl;
}

int main()
{
        Complex c1;
        c1.get();
        int a;
        cout<<"Enter a value to add it the real part : ";
        cin>>a;
```

```cpp
    Complex c2 = add(a, c1);
    cout<<"The Complex number after adding the real part is : ";
    c2.print();
    Complex c3 = add(c1, c2);
    cout<<"Addition of two Complex numbers : "<<endl;
    cout<<"  ";
    c1.print();
    cout<<" + ";
    c2.print();
    cout<<" = ";
    c3.print();
    return 0;
}
```

```
deven@deven-VirtualBox:~/C++Lab/asgn6$ g++ complexwithFriend.cpp
deven@deven-VirtualBox:~/C++Lab/asgn6$ ./a.out
Enter the real part and imaginary part of a Complex number : 5 10
Enter a value to add it the real part : 5
The Complex number after adding the real part is : 10 + i10
Addition of two Complex numbers :
   5 + i10
 + 10 + i10
 = 15 + i20
deven@deven-VirtualBox:~/C++Lab/asgn6$
```

2. Write a program to find the transpose of a matrix using friend function and with member functions to read and display the matrix.

#include <iostream>

using namespace std;

class matrix

```cpp
{
    private:
        int m,n;
        int a[3][3];
    public:
        matrix(): m(3),n(3){}
        void read();
        friend matrix findTranspose(matrix);
        void display();
};

void matrix::read()
{
    cout<<"Enter the elements of the matrix of order "<<m<<" x "<<n<<" : "<<endl;
    for(int i=0;i<m;++i)
    {
        for(int j=0;j<n;++j)
        {
            cin>>a[i][j];
        }
    }
}

matrix findTranspose(matrix A)
{
    matrix TA;
```

```cpp
        for(int i=0;i<A.m;++i)
        {
                for(int j=0;j<A.n;++j)
                {
                        TA.a[j][i] = A.a[i][j];
                }
        }
        return TA;
}

void matrix::display()
{
        cout<<"The matrix is : "<<endl;
        for(int i=0;i<m;++i)
        {
                for(int j=0;j<n;++j)
                {
                        cout<<a[i][j]<<" ";
                }
                cout<<endl;
        }
        cout<<endl;
}

int main()
{
```
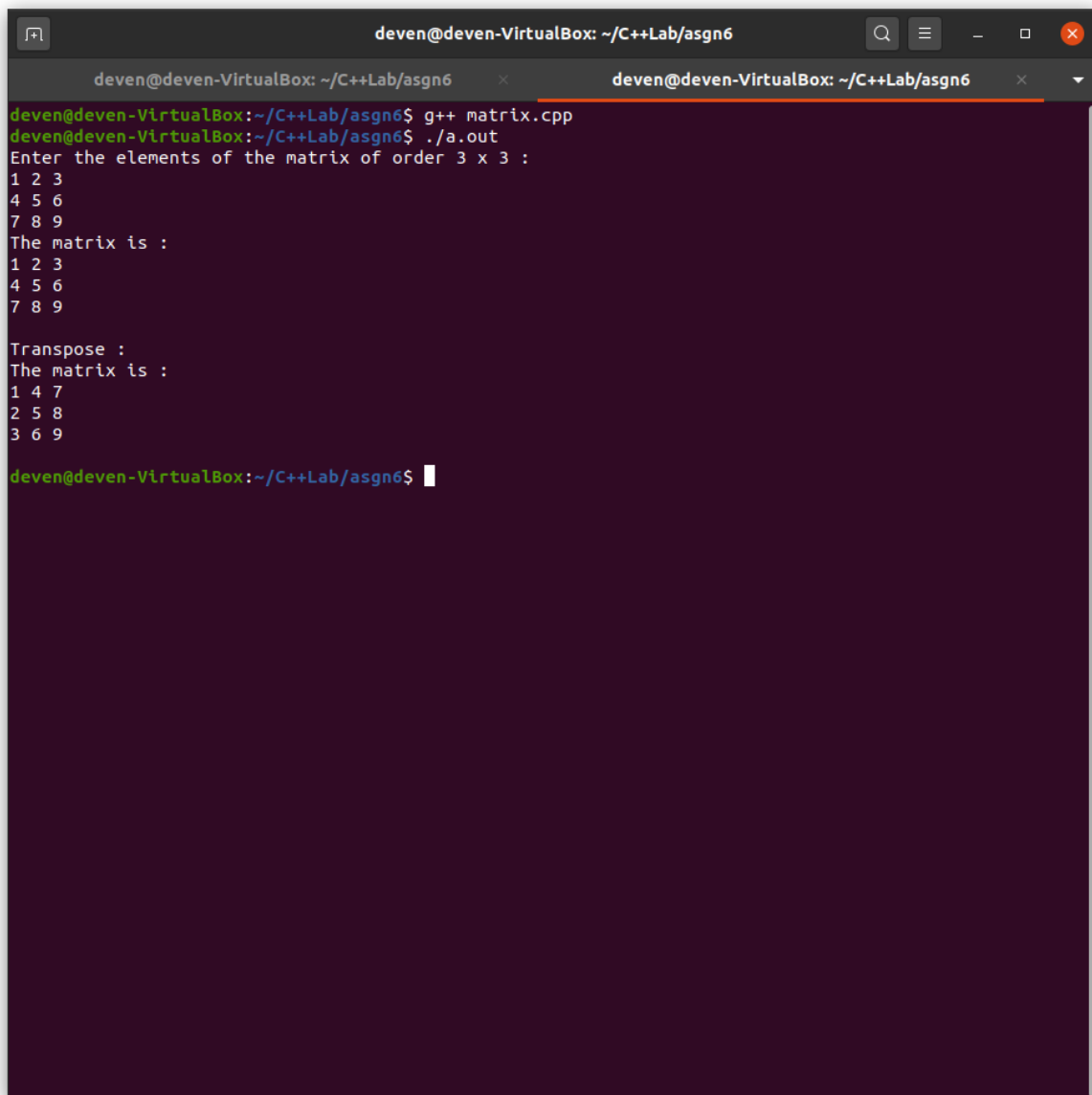
```cpp
    matrix A;

    A.read();

    A.display();

    matrix TA = findTranspose(A);

    cout<<"Transpose : "<<endl;

    TA.display();

    return 0;

}
```

```
deven@deven-VirtualBox:~/C++Lab/asgn6$ g++ matrix.cpp
deven@deven-VirtualBox:~/C++Lab/asgn6$ ./a.out
Enter the elements of the matrix of order 3 x 3 :
1 2 3
4 5 6
7 8 9
The matrix is :
1 2 3
4 5 6
7 8 9

Transpose :
The matrix is :
1 4 7
2 5 8
3 6 9

deven@deven-VirtualBox:~/C++Lab/asgn6$
```

3. Prepare a class Student with suitable attributes and member functions. Add a friend function Predict to print the grade for a course.

```
#include <iostream>

using namespace std;


class student
{
        private:
                int id;
                string name;
                int marks[4];
        public:
                void read();
                void display();
                friend void predictGrade(student);
};


void student::read()
{
        cout<<"Enter the student's id and name : "<<endl;
        cin>>id>>name;
        cout<<"Enter the 4 courses marks : "<<endl;
        for(int i=0;i<4;++i)
                cin>>marks[i];
}


void student::display()
```

```cpp
{
    cout<<"Student details : "<<endl;

    cout<<"Id : "<<id<<endl;

    cout<<"Name : "<<name<<endl;

    cout<<"Marks in the 4 courses : "<<endl;

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

        cout<<marks[i]<<" ";

    cout<<endl;
}


void predictGrade(student s)
{
    for(int i=0;i<4;++i)
    {
        cout<<"Grade of course "<<i+1<<" : ";
        if(s.marks[i]>=90)
            cout<<"S"<<endl;
        else if(s.marks[i]>=80)
            cout<<"A"<<endl;
        else if(s.marks[i]>=70)
            cout<<"B"<<endl;
        else if(s.marks[i]>=60)
            cout<<"C"<<endl;
        else if(s.marks[i]>=50)
            cout<<"D"<<endl;
        else if(s.marks[i]>=40)
```
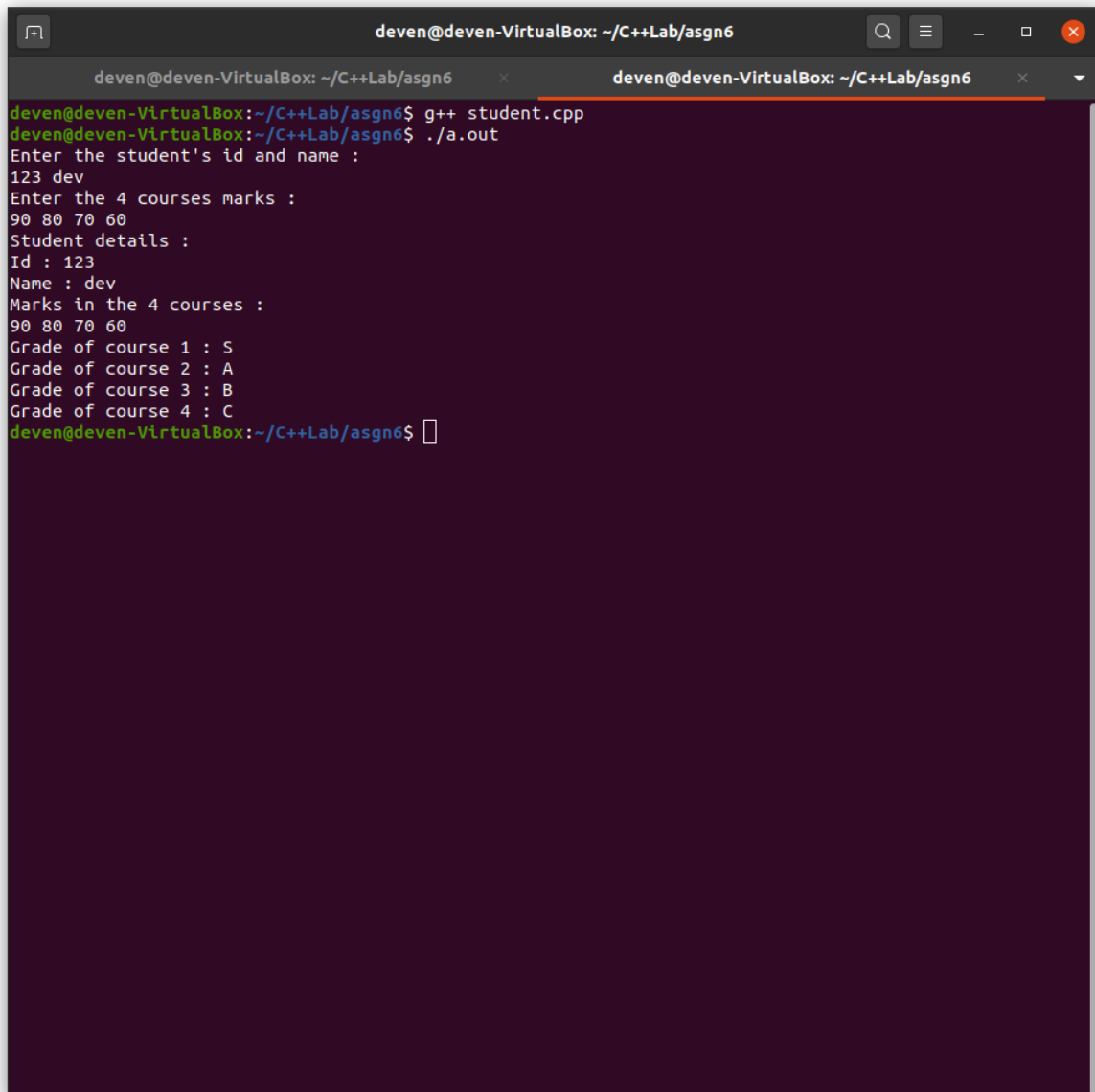
```cpp
                cout<<"E"<<endl;
        else
                cout<<"F"<<endl;
    }
}


int main()
{
    student s;
    s.read();
    s.display();
    predictGrade(s);
    return 0;
}
```

4. Prepare a class Hexa and a class Oct with suitable attributes and member functions. Add a friend function Convert to convert the numbers (both the classes) into Decimal format.

#include <iostream>

using namespace std;

class Hexa

```cpp
{
	private :
		string number;
	public :
		Hexa(string n = "0") : number(n) {}
		void read()
		{
			cout << "Enter a Hexadecimal number : ";
			cin >> number;
		}
		void display(){ cout << "The Hexadecimal number is : " << number << endl; }
		friend int toDecimal(Hexa);
};

class Oct
{
	private :
		int number;
	public :
		Oct(int n = 0) : number(n) {}
		void read()
		{
			cout << "Enter a Octal number : ";
			cin >> number;
		}
}
```

```cpp
            void display(){ cout << "The Octal number is : " << number << endl;
}

            friend int toDecimal(Oct);

};


int toDecimal(Hexa h)

{

      int len = h.number.size();

      int base = 1; // 16^0 = 1*********************************

      int decimalNum = 0;

      for(int i = len-1; i >= 0; --i)

      {

            if(h.number[i] >= '0' && h.number[i] <= '9')

                  decimalNum += (h.number[i] - '0') * base;

            else//if(h.numer[i]>='A' && h.number[i]<='F')

                  decimalNum += (h.number[i] - 'A' + 10) * base;// or use
pow();

            base *= 16;//base is 16, so 16^0, 16^1, 16^2 etc

      }

      return decimalNum;

}


int toDecimal(Oct o)

{

      int base = 1;

      int decimalNum = 0;

      int octalNum = o.number, rem;
```

```cpp
        while(octalNum != 0)

        {

                rem = octalNum % 10;

                decimalNum += (rem * base);

                base *= 8;

                octalNum /= 10;

        }

        return decimalNum;

}


int main()

{

        Hexa h;

        h.read();

        h.display();

        cout << "The decimal form of this number is : " << toDecimal(h) << endl;

        Oct o;

        o.read();

        o.display();

        cout << "The decimal form of this number is : " << toDecimal(o) << endl;

        return 0;

}
```

```
deven@deven-VirtualBox:~/C++Lab/asgn6$ g++ toDecimal.cpp
deven@deven-VirtualBox:~/C++Lab/asgn6$ ./a.out
Enter a Hexadecimal number : 1A
The Hexadecimal number is : 1A
The decimal form of this number is : 26
Enter a Octal number : 108
The Octal number is : 108
The decimal form of this number is : 72
deven@deven-VirtualBox:~/C++Lab/asgn6$
```