1.Write C++ program to overload Area() function to calculate area of different shapes like square, circle, triangle and rectangle.

main.cpp

```cpp
#include <iostream>
using namespace std;
class Shape {
public:
    virtual double Area() = 0;
};
class Square:public Shape {
private:
    double side;
public:
    Square(double s) : side(s) {}
    double Area() {
        return side*side;
    }
};
class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
    double Area() {
        return 3.14*radius*radius;
    }
};
class Triangle : public Shape {
private:
    double base,height;
public:
    Triangle(double b,double h):base(b), height
        (h) {}
    double Area() {
        return 0.5*base*height;
    }
};
class Rectangle : public Shape {
private:
    double length,width;
public:
    Rectangle(double l,double w) : length(l),
        width(w) {}
    double Area() {
        return length*width;
    }
};
int main() {
    Square sq(5);
    Circle cir(6);
    Triangle tri(2,3);
    Rectangle rect(4,5);
    Shape*shapes[4] = {&sq,&cir,&tri,&rect};
    for (int i = 0; i < 4; i++) {
        cout<<"Area of shape "<< i+ 1<<": "
            <<shapes[i]->Area()<<endl;
    }
    return 0;
}
```

Output

```
/tmp/LjKmNdrOjx.o
Area of shape 1: 25
Area of shape 2: 113.04
Area of shape 3: 3
Area of shape 4: 20


=== Code Execution Successful ===
```

2.Create a class Student with data members regno, Name, mobile, dep[10], sem, year and cgpa. Create member function StudDetails() and friend function maxmark().
StudDetails()- Displays the student details.
Maxmark()- Finds and prints the student who scored maximum cgpa.
Create 2 Student objects, display their details and print the maximum mark scored student.

main.cpp

Run

```cpp
#include <iostream>
using namespace std;
class Student{
  char*regno;
  char*name;
  char*mobile;
  char*dep;
  int sem;
  int year,cgpa;
  public:
  Student(){}
  Student(char*r,char*n,char*m,char*d,int s,
      int y,int c){
    regno = r;
    name = n;
    mobile = m;
    dep = d;
    sem = s;
    year = y;
    cgpa = c;
  }
  void StudDetails(){
    cout <<"Regno : "<<regno<<"\nName : "<<
        name<< "\nMobile : "<< mobile
        <<"\nDepartment : "<<dep<<"\nSemester
        "<<sem<<"\nYear : "<<year<<"\nCGPA :"<
        cgpa<<endl;
  }
  friend void Maxmark(Student a, Student b);
};
void Maxmark(Student a, Student b){
  Student c;
  if(a.cgpa > b.cgpa){
    c.cgpa = a.cgpa;
    c.name = a.name;
  }
  else{
    c.cgpa = b.cgpa;
    c.name = b.name;
  }
  cout<<"The Max mark Scored Student Details :
      <<endl<<"Name   :"<<c.name<<endl<<"cgpa
       : "<<c.cgpa;
}
int main()
{
 Student s1,s2;
 s1 = Student("61772231031","Suresh",
     "9876543210","ECE", 5, 4, 9.15);
 s2 = Student("61772231015","Ramesh",
     "9123456780","ECE", 4, 5, 10);
 s1.StudDetails();
 cout<<"\n\n\n";
 s2.StudDetails();
 cout<<"\n\n\n";
 Maxmark(s1,s2);
}
```

```
/tmp/kv7myJiAAI.o
Regno : 61772231031
Name : Suresh
Mobile : 9876543210
Department : ECE
Semester : 5
Year : 4
CGPA :9



Regno : 61772231015
Name : Ramesh
Mobile : 9123456780
Department : ECE
Semester : 4
Year : 5
CGPA :10



The Max mark Scored Student Details :
Name   :Ramesh
cgpa   : 10

=== Code Execution Successful ===
```

3.Implement a C++ program to calculate the sum of two matrices using operator overloading.

main.cpp

```cpp
#include<iostream>
using namespace std;
class Matrix{
  int a[3][3];
  public:
    void initialize(){
      for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
          cout << "Enter M_"<<i+1<<j+1<< " : ";
          cin >> a[i][j];
        }
      }
    }
    Matrix operator+(Matrix b){
      Matrix c;
      for(int i =0;i<3;i++){
        for(int j =0;j<3;j++){
          c.a[i][j] = a[i][j] + b.a[i][j];
        }
      }
      return c;
    }
    void display(){
      for(int i=0;i<3;i++){
        for(int j = 0;j<3;j++){
          cout<<a[i][j]<<" ";
        }
        cout<<endl;
      }
    }
};
int main(){
  Matrix a,b,c;
  a.initialize();
  cout<<endl<<endl;
  b.initialize();
  c =a+b;
  cout<<"\n\n\nThe addition of the Matrices A and B : "<<endl;
  c.display();
}
```

Output

```
/tmp/L3EHmLPtEi.o
Enter M_11 : 1
Enter M_12 : 2
Enter M_13 : 3
Enter M_21 : 4
Enter M_22 : 5
Enter M_23 : 6
Enter M_31 : 7
Enter M_32 : 8
Enter M_33 : 9


Enter M_11 : 1
Enter M_12 : 2
Enter M_13 : 3
Enter M_21 : 4
Enter M_22 : 5
Enter M_23 : 6
Enter M_31 : 7
Enter M_32 : 8
Enter M_33 : 9


The addition of the Matrices A and B :
2 4 6
8 10 12
14 16 18


=== Code Execution Successful ===
```

4. Define a class String that could work as a user-defined string type. Include constructors that will enable us to create an uninitialized string String s1; and also to initialize an object with a string constant at the time of creation like String s2("Welcome");
Include a function that adds two strings to make a third string.
Write a complete program to test your class to see that it does the following tasks:
(a) Creates uninitialized string objects.
(b) Creates objects with string constants.
(c) Concatenates two strings properly.
(d) Displays a desired string object.

main.cpp

```cpp
#include<iostream>
using namespace std;
class String{
  char *ch;
  public:
  String(){
    ch = new char[50];
  }
  String(char *c){
    ch =c;
  }
  String add(String b){
    String c;
    int i;
    for(i = 0;ch[i]!='\0';i++){
      c.ch[i]=ch[i];
    }
    for(int j=0;b.ch[j]!='\0';j++){
      c.ch[i]=b.ch[j];
      i++;
    }
    return c;
  }
  void display(){
    for(int i=0;ch[i]!='\0';i++){
      cout<<ch[i];
    }
  }
};
int main()
{
  String s1,s2;
  s1 = String(" Good");
  s2 = String(" Morning");
  String s3=s1.add(s2);
  cout<<"String 1 : ";
  s1.display();
  cout<<endl<<"String 2 : ";
  s2.display();
  cout<<endl<<"String 1 + String 2 : ";
  s3.display();
}
```

```
/tmp/pvZNcPX8Ha.o
String 1 :  Good
String 2 :  Morning
String 1 + String 2 :  Good Morning

=== Code Execution Successful ===
```

5.Apply multiple inheritance concept using employee details and show their data.

main.cpp

Run

```cpp
#include <iostream>
using namespace std;
class Manager{
  protected:
  char* name;
  int age;
  int salary;
  public:
  Manager(){ }
  Manager(char* n,int a,int s){
    name = n;
    age = a;
    salary = s;
  }
};
class Worker{
  protected:
  char* name;
  int age;
  int salary;
  public:
  Worker(){ }
  Worker(char* n,int a,int s){
    name = n;
    age = a;
    salary = s;
  }
};
class Details : public Manager , public Worker{
  public:
  Details(char* n1, int a1, int s1, char* n2,
      int a2, int s2) : Manager(n1,a1,s1),
      Worker(n2,a2,s2){ }
  void display(){
    cout << "Details of Manager : "<<endl<<
        "Name : "<<Manager::name<<endl<< "Age :
        "<<Manager::age<<endl<<"Salary : "
        <<Manager::salary<<endl<<endl;
    cout << "Details of Worker : " <<endl
        <<"Name : "<<Worker::name<<endl<<"Age :
        "<<Worker::age<<endl<<"Salary : "
        <<Worker::salary;
  }
};
int main()
{
 Details d("Suresh",50,100000,"Ramesh",30,50000
     );
 d.display();
}
```

```
/tmp/16uiN4h4v7.o
Details of Manager :
Name : Suresh
Age : 50
Salary : 100000

Details of Worker :
Name : Ramesh
Age : 30
Salary : 50000

=== Code Execution Successful ===
```