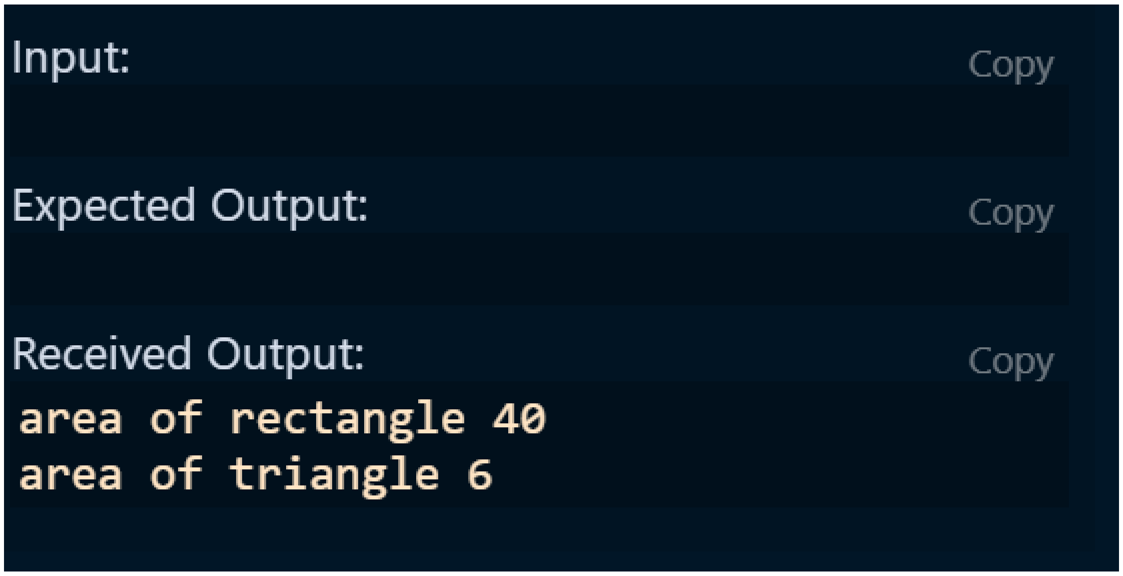
WAP to implement a C++ program to find out the area of the rectangle and triangle using hierarchical inheritance .

```
#include <iostream>
using namespace std;
class Shape{
    protected:
    float area;

    public:
    Shape(){
        area = 0.0;
    }
};

class Rectangle: public Shape{
    protected:
        int length;
        int breadth;

    public:
    Rectangle(int l, int b){
        length = l;
        breadth = b;
    }

    void calculateAreaofRect(){
        area = length * breadth;
    }

    void diplay(){
        cout << "area of rectangle " << area << endl;
    }
};

class Triangle: public Shape{
    protected:
        int base;
        int height;

    public:
    Triangle(int b, int h){
        base = b;
        height = h;
    }

    void calculateAreaofTri(){
        area = (base * height) / 2;
    }

    void display(){
        cout << "area of triangle " << area << endl;
    }
};

int main()
{
    Rectangle r(8, 5);
    r.calculateAreaofRect();
    r.diplay();
    Triangle t(3, 4);
    t.calculateAreaofTri();
    t.display();

    return 0;
}
```

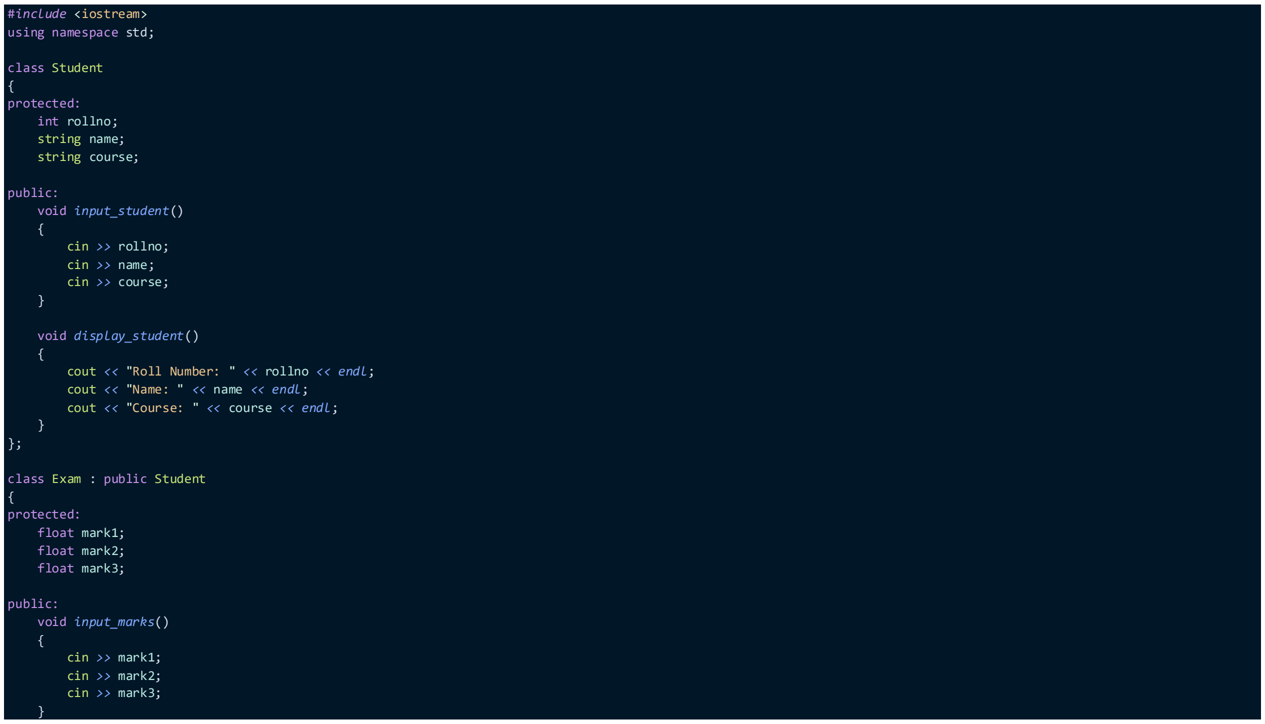

Input: Copy

Expected Output: Copy

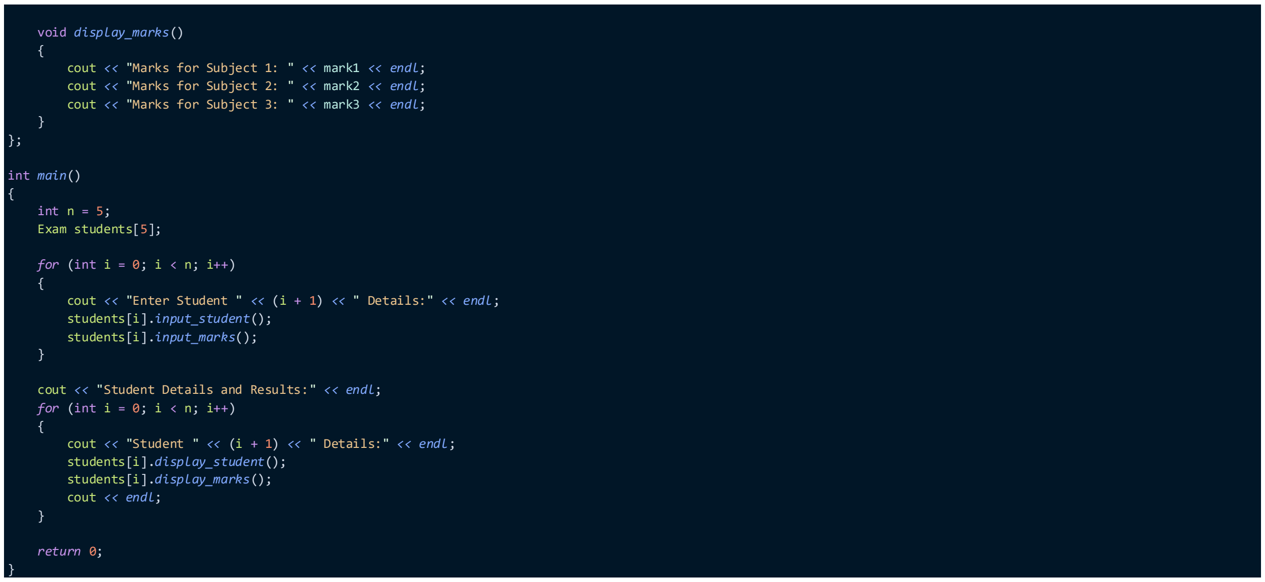
Received Output: Copy

```
area of rectangle 40
area of triangle 6
```

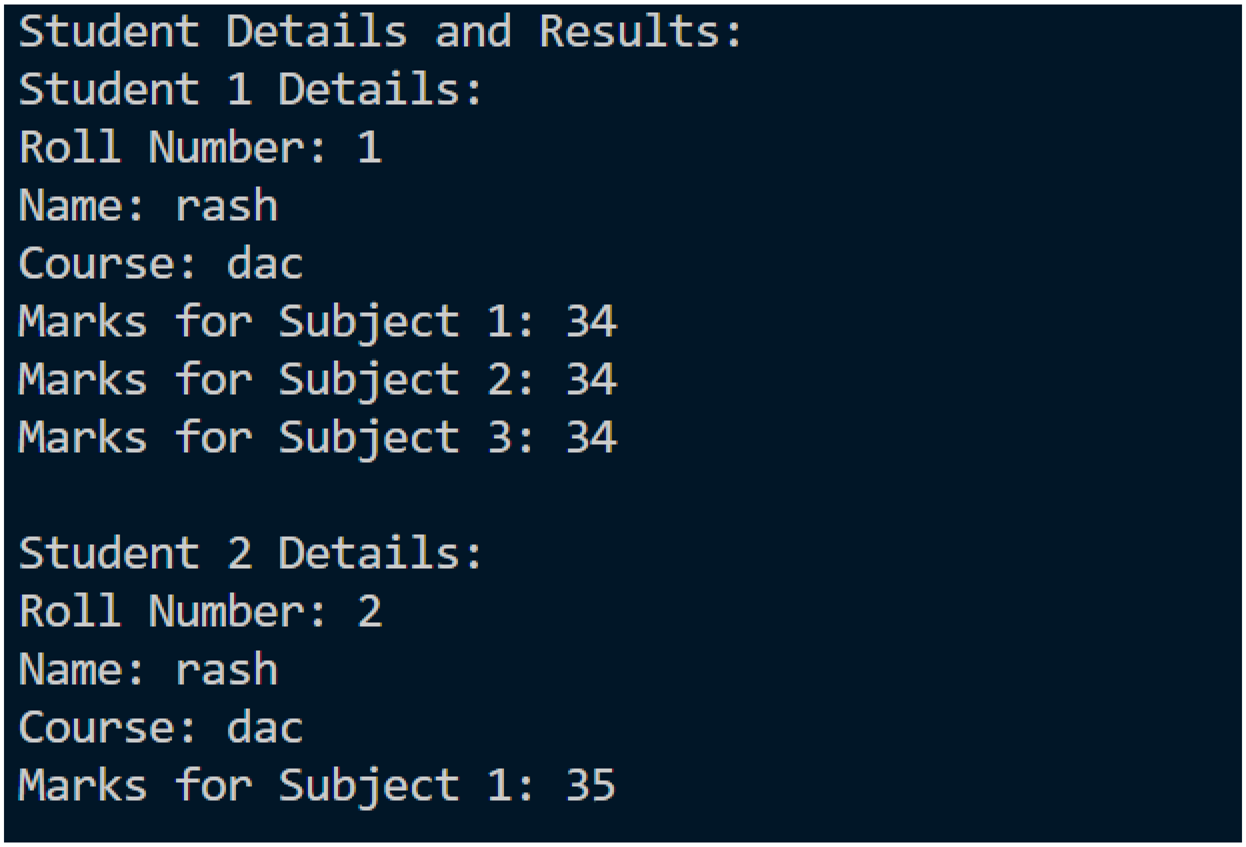
WAP to implement a C++ program to find out the student details using multilevel inheritance. Class student contains roll number, name and course as data member and Input_student and display_student as member function. A derived class exam is created from the class student with publicly inherited. The derived class contains mark1, mark2, mark3 as marks of three subjects and input_marks and display_result as member function. Create an array of object of the exam class and display the result of 5 students. Try the same program with privately inheritance.



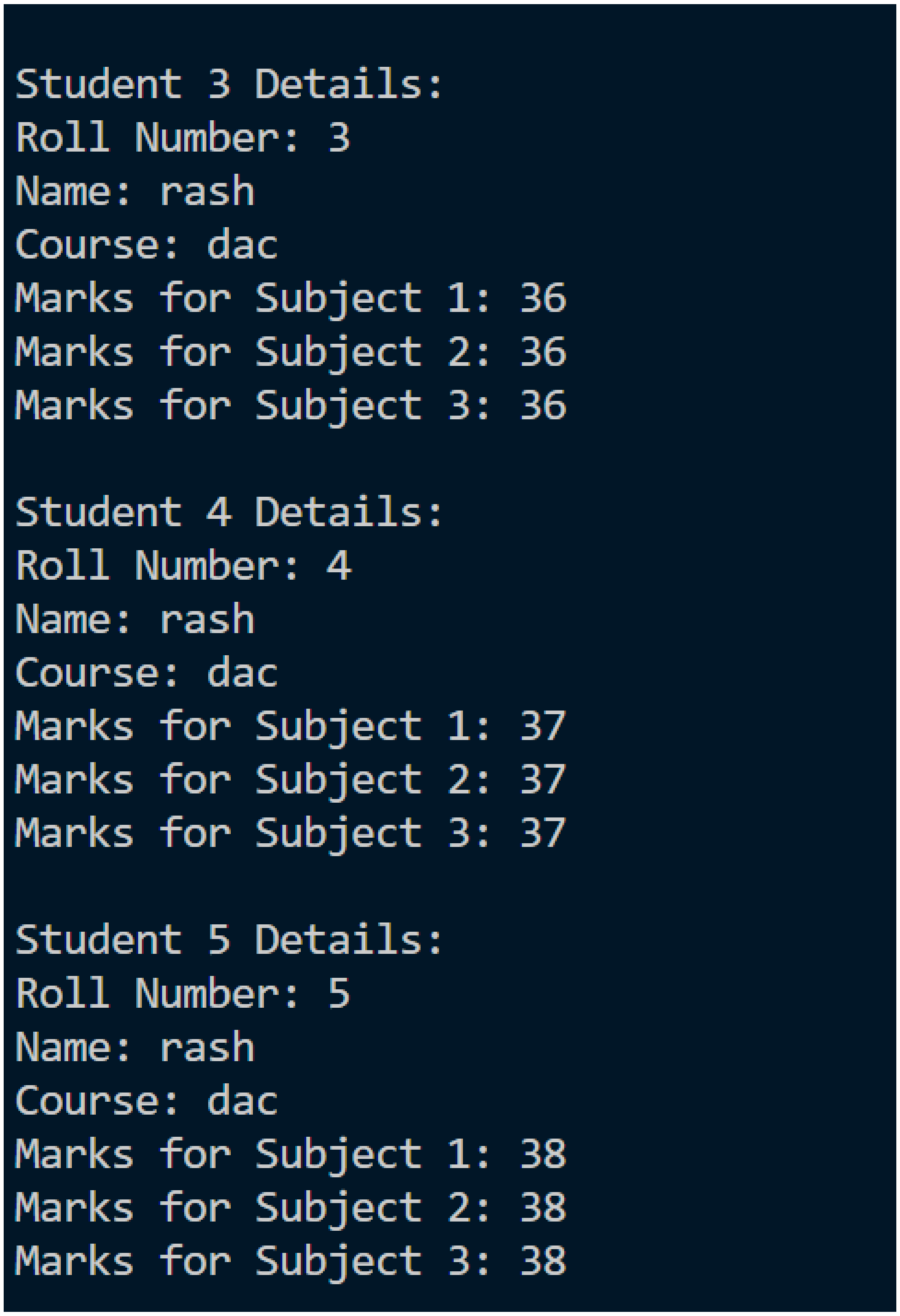
```cpp
#include <iostream>
using namespace std;

class Student
{
protected:
    int rollno;
    string name;
    string course;

public:
    void input_student()
    {
        cin >> rollno;
        cin >> name;
        cin >> course;
    }

    void display_student()
    {
        cout << "Roll Number: " << rollno << endl;
        cout << "Name: " << name << endl;
        cout << "Course: " << course << endl;
    }
};

class Exam : public Student
{
protected:
    float mark1;
    float mark2;
    float mark3;

public:
    void input_marks()
    {
        cin >> mark1;
        cin >> mark2;
        cin >> mark3;
    }
```



```cpp
    void display_marks()
    {
        cout << "Marks for Subject 1: " << mark1 << endl;
        cout << "Marks for Subject 2: " << mark2 << endl;
        cout << "Marks for Subject 3: " << mark3 << endl;
    }
};

int main()
{
    int n = 5;
    Exam students[5];

    for (int i = 0; i < n; i++)
    {
        cout << "Enter Student " << (i + 1) << " Details:" << endl;
        students[i].input_student();
        students[i].input_marks();
    }

    cout << "Student Details and Results:" << endl;
    for (int i = 0; i < n; i++)
    {
        cout << "Student " << (i + 1) << " Details:" << endl;
        students[i].display_student();
        students[i].display_marks();
        cout << endl;
    }

    return 0;
}
```

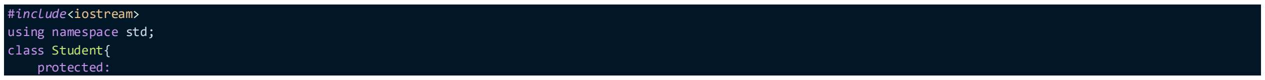


```
Student Details and Results:
Student 1 Details:
Roll Number: 1
Name: rash
Course: dac
Marks for Subject 1: 34
Marks for Subject 2: 34
Marks for Subject 3: 34

Student 2 Details:
Roll Number: 2
Name: rash
Course: dac
Marks for Subject 1: 35
```

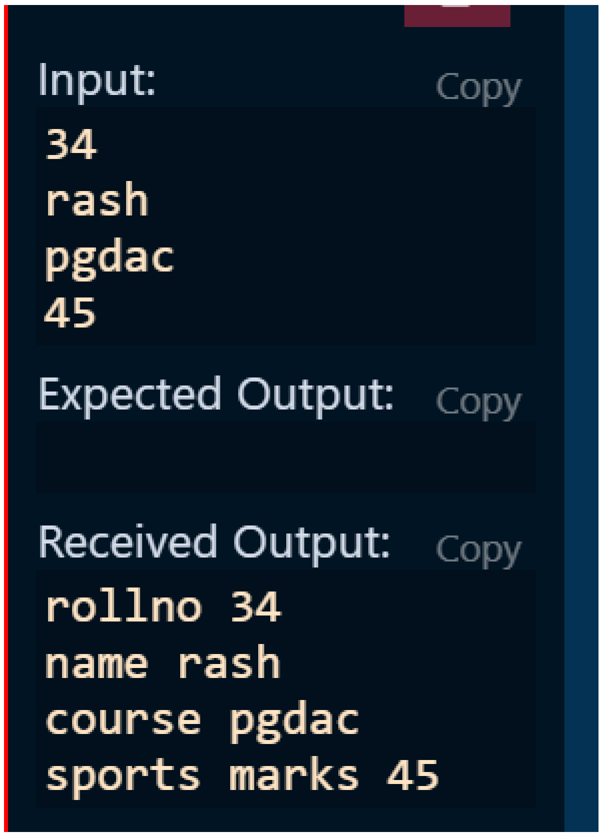


```
Student 3 Details:
Roll Number: 3
Name: rash
Course: dac
Marks for Subject 1: 36
Marks for Subject 2: 36
Marks for Subject 3: 36

Student 4 Details:
Roll Number: 4
Name: rash
Course: dac
Marks for Subject 1: 37
Marks for Subject 2: 37
Marks for Subject 3: 37

Student 5 Details:
Roll Number: 5
Name: rash
Course: dac
Marks for Subject 1: 38
Marks for Subject 2: 38
Marks for Subject 3: 38
```

WAP to implement a C++ program to find out the student details and sport score using hybrid inheritance.



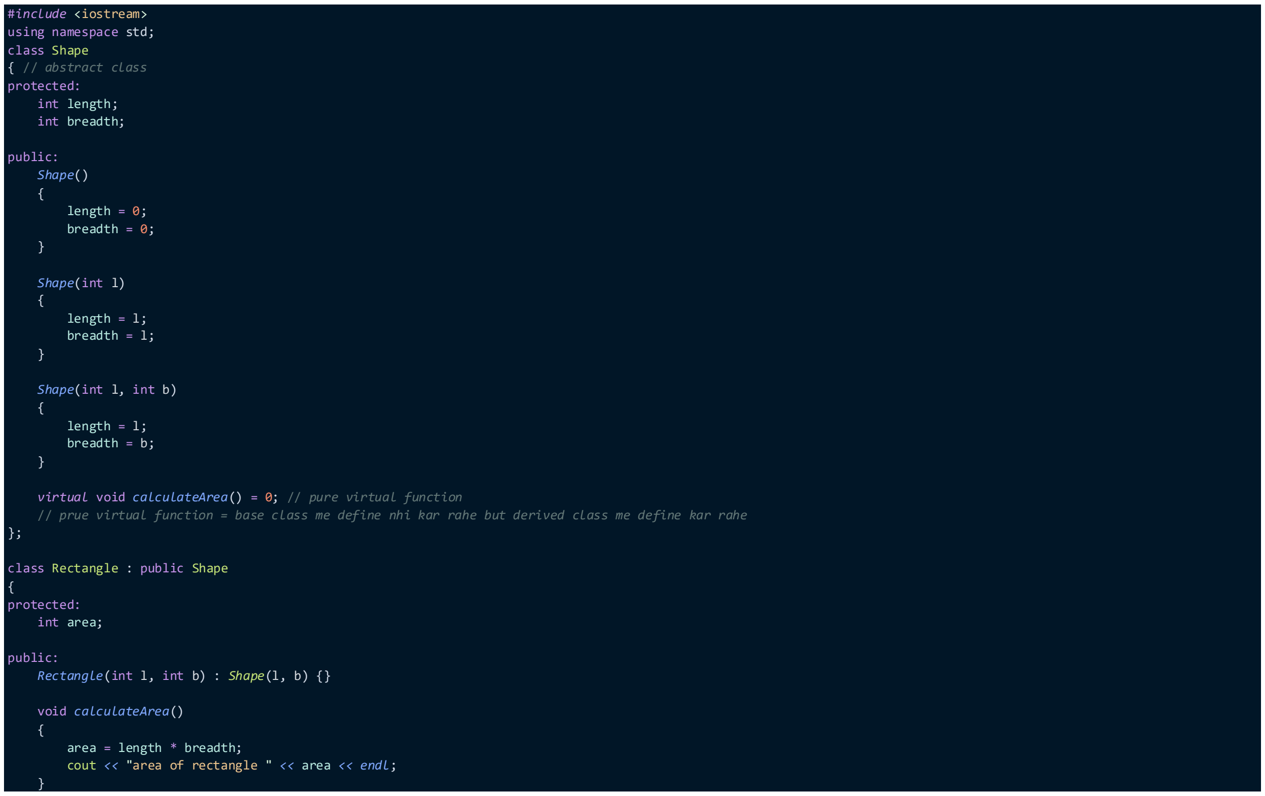
```cpp
#include<iostream>
using namespace std;
class Student{
    protected:
        int rollno;
        string name;
        string course;

    public:
    void input_student(){
        cin >> rollno;
        cin >> name;
        cin >> course;
    }

    void display_student(){
        cout << "rollno " << rollno << endl;
        cout << "name " << name << endl;
        cout << "course " << course << endl;
    }
};

class Sports{
    protected:
        int sports_marks;

    public:
    void input_sports_marks(){
        cin >> sports_marks;
    }

    void display_sports_marks(){
        cout << "sports marks " << sports_marks << endl;
    }
};

class StudentwithSports: public Student, public Sports{
    public:
    void input_studentwithscore(){
        input_student();
        input_sports_marks();
    }

    void display_studentwithscore(){
        display_student();
        display_sports_marks();
    }
};

int main()
{
    StudentwithSports s;
    s.input_studentwithscore();
    s.display_studentwithscore();

    return 0;
}
```



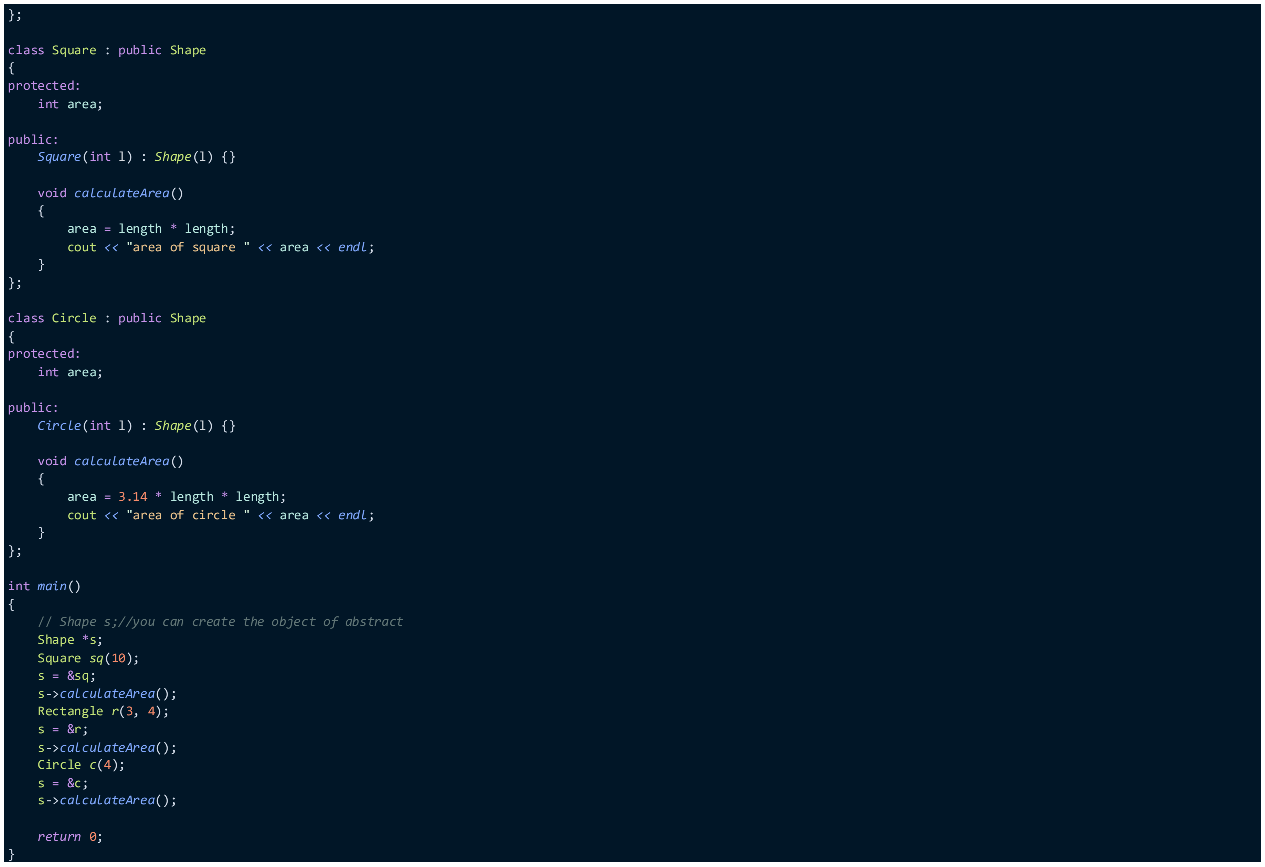
```
34
rash
pgdac
45
```

```
rollno 34
name rash
course pgdac
sports marks 45
```

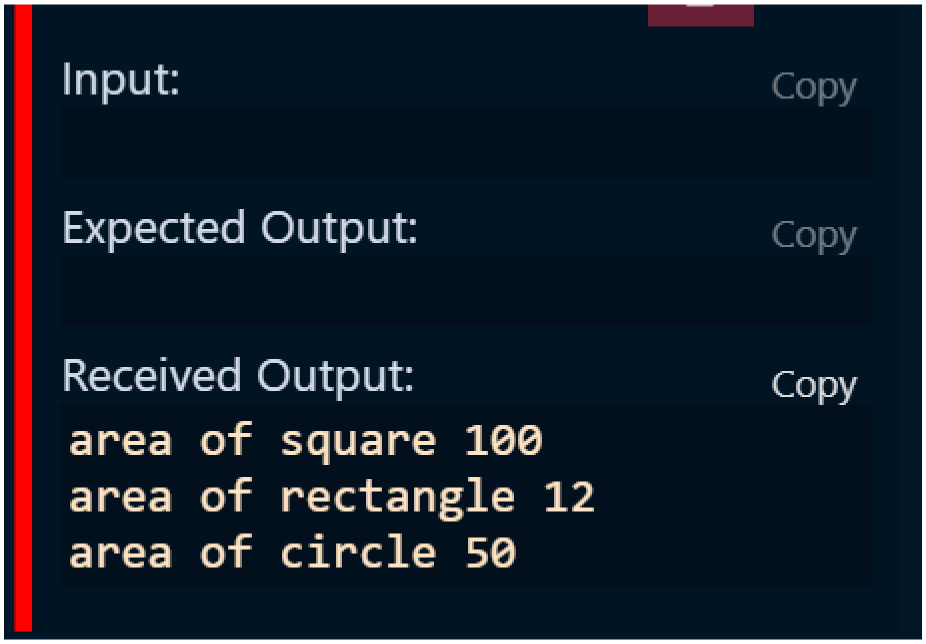
Implement function overriding by creating class shape through which area of figures are calculated.



```cpp
#include <iostream>
using namespace std;
class Shape
{ // abstract class
protected:
    int length;
    int breadth;

public:
    Shape()
    {
        length = 0;
        breadth = 0;
    }

    Shape(int l)
    {
        length = l;
        breadth = l;
    }

    Shape(int l, int b)
    {
        length = l;
        breadth = b;
    }

    virtual void calculateArea() = 0; // pure virtual function
    // prue virtual function = base class me define nhi kar rahe but derived class me define kar rahe
};

class Rectangle : public Shape
{
protected:
    int area;

public:
    Rectangle(int l, int b) : Shape(l, b) {}

    void calculateArea()
    {
        area = length * breadth;
        cout << "area of rectangle " << area << endl;
    }
```

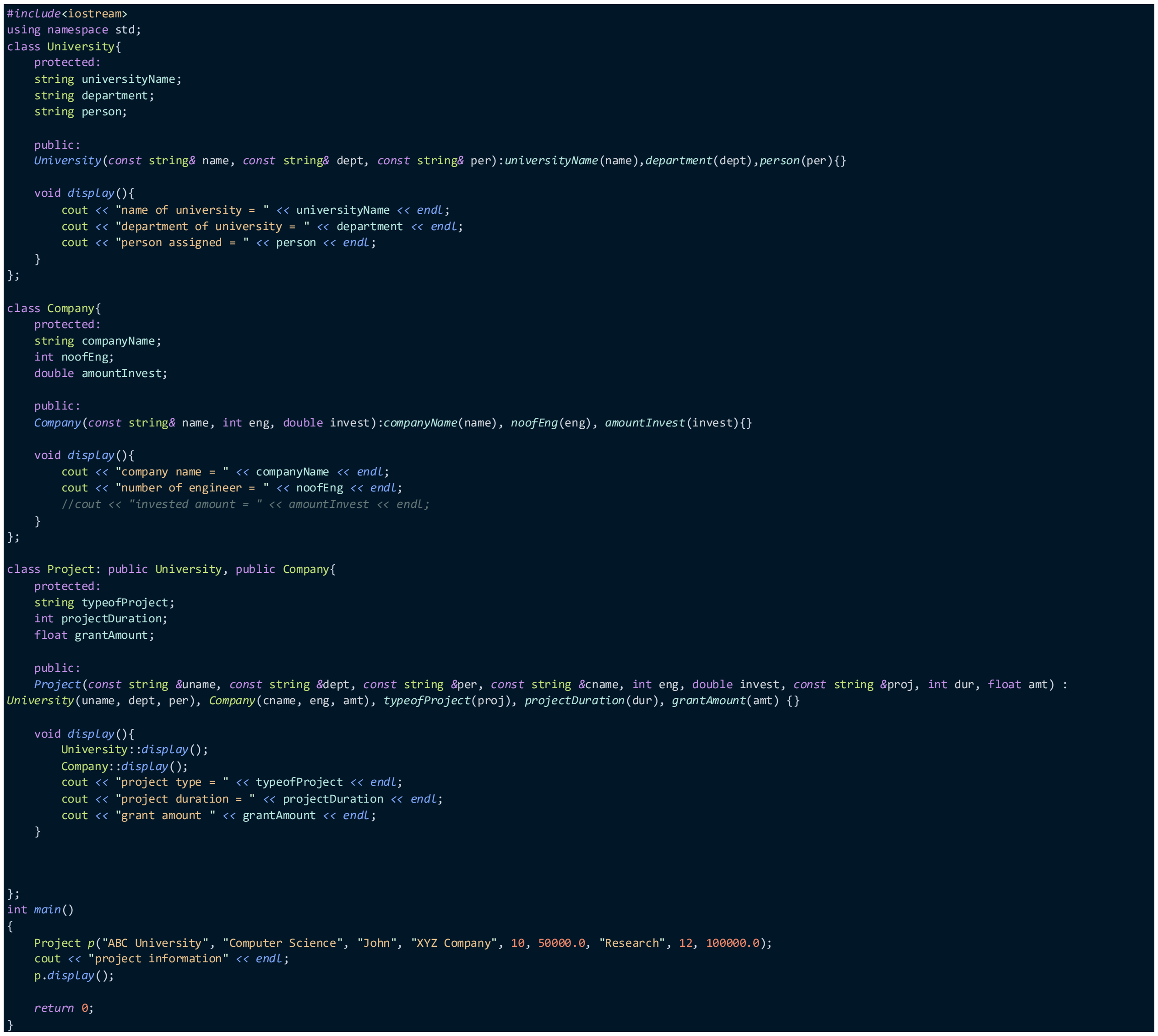


```cpp
};

class Square : public Shape
{
protected:
    int area;

public:
    Square(int l) : Shape(l) {}

    void calculateArea()
    {
        area = length * length;
        cout << "area of square " << area << endl;
    }
};

class Circle : public Shape
{
protected:
    int area;

public:
    Circle(int l) : Shape(l) {}

    void calculateArea()
    {
        area = 3.14 * length * length;
        cout << "area of circle " << area << endl;
    }
};

int main()
{
    // Shape s;//you can create the object of abstract
    Shape *s;
    Square sq(10);
    s = &sq;
    s->calculateArea();
    Rectangle r(3, 4);
    s = &r;
    s->calculateArea();
    Circle c(4);
    s = &c;
    s->calculateArea();

    return 0;
}
```

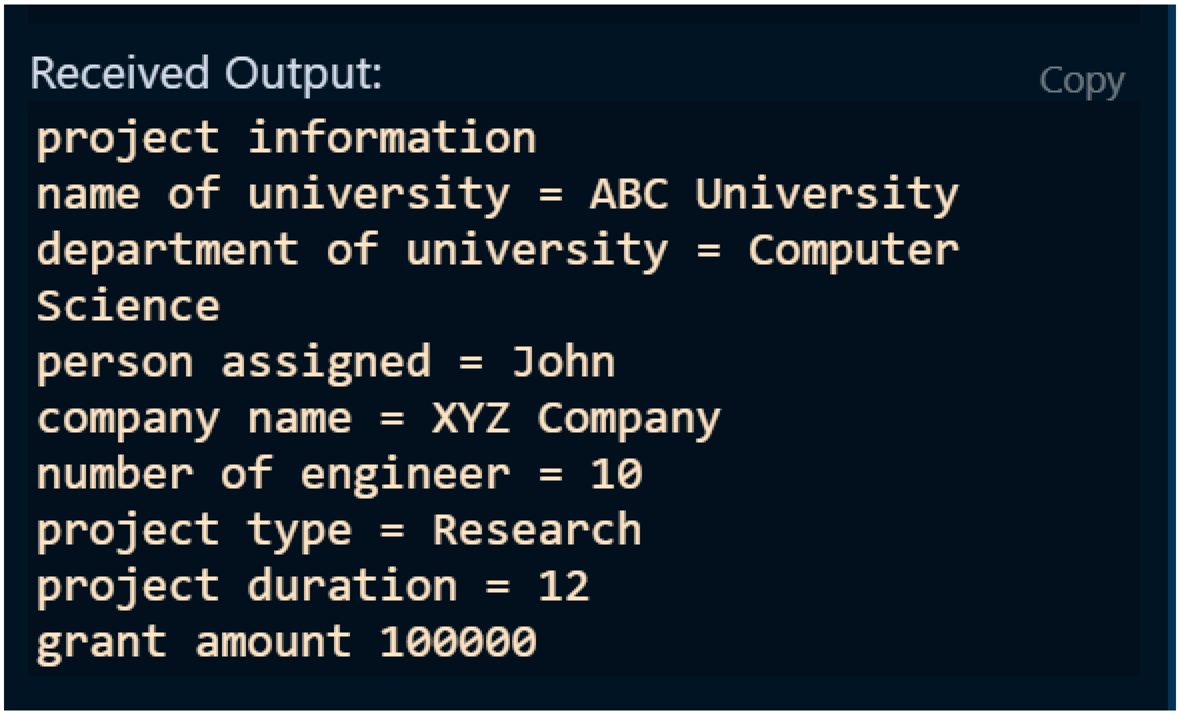


```
area of square 100
area of rectangle 12
area of circle 50
```

A University and a Company have jointly taken a project. Class University contains name of the university, department to which the project is assigned, person to whom the project is assigned. A function display is there to display the information. Class Company contains name of the company, Number of Engineers assigned, amount invested to do the project. A function display is there to display the information. Class Project is inherited from University and Company. It contains type of project, duration of project, amount granted to complete the project. A function display displays the related information. Write a C++ program to implement this and display all information except amount invested by company from Project class



```cpp
#include<iostream>
using namespace std;
class University{
    protected:
    string universityName;
    string department;
    string person;

    public:
    University(const string& name, const string& dept, const string& per):universityName(name),department(dept),person(per){}

    void display(){
        cout << "name of university = " << universityName << endl;
        cout << "department of university = " << department << endl;
        cout << "person assigned = " << person << endl;
    }
};

class Company{
    protected:
    string companyName;
    int noofEng;
    double amountInvest;

    public:
    Company(const string& name, int eng, double invest):companyName(name), noofEng(eng), amountInvest(invest){}

    void display(){
        cout << "company name = " << companyName << endl;
        cout << "number of engineer = " << noofEng << endl;
        //cout << "invested amount = " << amountInvest << endl;
    }
};

class Project: public University, public Company{
    protected:
    string typeofProject;
    int projectDuration;
    float grantAmount;

    public:
    Project(const string &uname, const string &dept, const string &per, const string &cname, int eng, double invest, const string &proj, int dur, float amt) :
University(uname, dept, per), Company(cname, eng, amt), typeofProject(proj), projectDuration(dur), grantAmount(amt) {}

    void display(){
        University::display();
        Company::display();
        cout << "project type = " << typeofProject << endl;
        cout << "project duration = " << projectDuration << endl;
        cout << "grant amount " << grantAmount << endl;
    }



};
int main()
{
    Project p("ABC University", "Computer Science", "John", "XYZ Company", 10, 50000.0, "Research", 12, 100000.0);
    cout << "project information" << endl;
    p.display();

    return 0;
}
```


Received Output:

Copy

```
project information
name of university = ABC University
department of university = Computer
Science
person assigned = John
company name = XYZ Company
number of engineer = 10
project type = Research
project duration = 12
grant amount 100000
```