SCS 1209 - Object Oriented Programming

Practical 09

1. Write a C++ program to find the area of a rectangle using inheritance.
2. Write a C++ program to make a class named Vegetable with a data member to calculate the number of vegetables in a basket. Create two other classes named Carrot and Potato to calculate the number of carrots and potatoes in the basket. Print the number of vegetables of each type and the total number of vegetables in the basket.
3. We want to calculate the total marks of each student of a class in Biology,Chemistry and Physics and the average marks of the class. The number of students in the class are entered by the user. Create a class named Marks with data members for roll number, name and marks. Create three other classes inheriting the Marks class, namely, Biology,Chemistry and Physics, which are used to define marks in individual subjects of each student. Roll number of each student will be generated automatically.
4. Check the output of the following code

```
#include<iostream>
using namespace std;
class base
{
        public:
                void fun_1() { cout << "base-1\n"; }
                virtual void fun_2() { cout << "base-2\n"; }
                virtual void fun_3() { cout << "base-3\n"; }
                virtual void fun_4() { cout << "base-4\n"; }
};

class derived : public base
{
        public:
                void fun_1() { cout << "derived-1\n"; }
                void fun_2() { cout << "derived-2\n"; }
                void fun_4(int x) { cout << "derived-4\n"; }
};
int main()
{
        base *p;
        derived obj1;
        p = &obj1;
        // Early binding because fun1() is non-virtual
        // in base
        p->fun_1();
        // Late binding (RTP)
        p->fun_2();
        // Late binding (RTP)
        p->fun_3();
        // Late binding (RTP)
        p->fun_4();
}
```

5. Check the output of the code. Indicate the reason if there is an error.

```
#include<iostream>
using namespace std;
class base
{
        public:
                virtual void print ()
                { cout<< "print base class" <<endl; }
                void show ()
                { cout<< "show base class" <<endl; }
};

class derived:public base
{
        public:
                void print ()
                { cout<< "print derived class" <<endl; }
                void show ()
                { cout<< "show derived class" <<endl; }
};
int main()
{
        base *bptr;
        derived d;
        bptr = &d;
        bptr->print();
        bptr->show();
}
```