C++ Programming Assignment - 21

1. What is mean by Function overloading?

2. Why function overloading is considered as compile time polymorphism?

3. What is the use of function overloading?

4. What are the scenarios in which we can overload the function?

5. What is mean by name mangling / naming decoration?

6. Why return value is not considered as function overloading criteria?

7. What are the scenarios in which we can not perform function overloading?

8. Predict the output of below program.

```
class Demo
{
        public:
        void fun(int i)
        {
                cout<<" First defination";
        }

        void fun(int i, int j)
        {
                cout<<" Second defination";
        }
};

int main()
{
        Demo obj();

        obj.fun(10);
        obj.fun(10,20);

        return 0;
}
```

9. Predict the output of below program.

```
class Demo
{
        public:
        void fun(int *p)
        {
                cout<<" First defination";
        }

        void fun(float *p)
        {
                cout<<" Second defination";
        }

        void fun(int no)
        {
                cout<<" Third defination";
        }
};

int main()
{
        int no =11;
        float f = 3.14;

        Demo obj();

        obj.fun(no);
        obj.fun(&no);
        obj.fun(&f);

        return 0;
}
```

10. Draw object layout & class diagram of below code snippets and explain its internal working in detail. Explain the type of inheritance in the below code snippet.

```
class Base
{
        public:
                int i,j;
                static int k;

        Base()
        {
                i = 10;
                j = 20;
        }

        void fun()
        {
                cout<<“Base Fun";
        }
};

int Base::k = 11;

class Derived : public Base
{
        public:
                int x,y;

        Derived()
        {
                x = 50;
                y = 60;
        }
        void gun()
        {
                cout<<“Derived Gun";
        }
};

int main()
{
        Base bobj();
        Derived dobj();

        cout<<sizeof(bobj);
        cout<<sizeof(dobj);

        cout<<bobj.i;
        cout<<bobj.j;
        cout<<dobj.i;
        cout<<dobj.j;
        cout<<bobj.k;
        cout<<bobj.x;

        bobj.fun();
        dobj.fun();
        dobj.gun();

        return 0;
}
```