C++

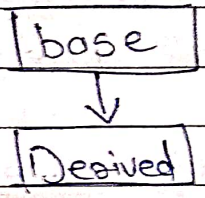
Assignment No- 20

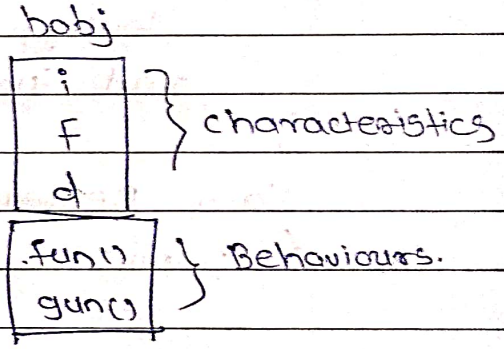
Q1) Draw object layout of below code and explain the internal working in detail.

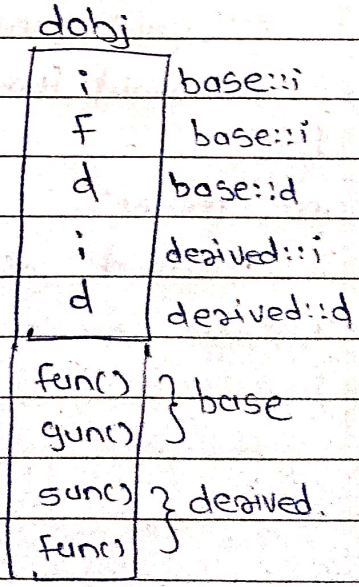
```
Class base
{ public:
    int i;
    float f;
    double d;

    Void fun() {}
    Void gun() {}
};
Class Derived : public base
{ public:
    int i;
    double d;

    Void sun() {}
    Void fun() {}
};
int main()
{ base bobj;
  Derived dobj;
  return 0;
}
```

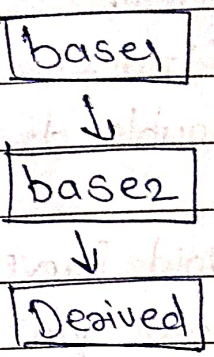






- Single level Inheritance
The derived class inherits all members of base

```
class base1
{ public:
    int i;
    float f;
    void gun() {}
};
class base2
{ public:
    int j;
    float g;
    void fun() {}
};
class Derived : public base1, base2
{ public:
    int i;
    double d;
    void sun() {}
    void fun() {}
};
int main()
{ Derived dObj;
return 0;
}
```



dobj

i	base1::i
f	base1::f
j	base2::j
g	base2::g
d	derived::d

gun()	base1
fun()	bas2
sun()	Derived

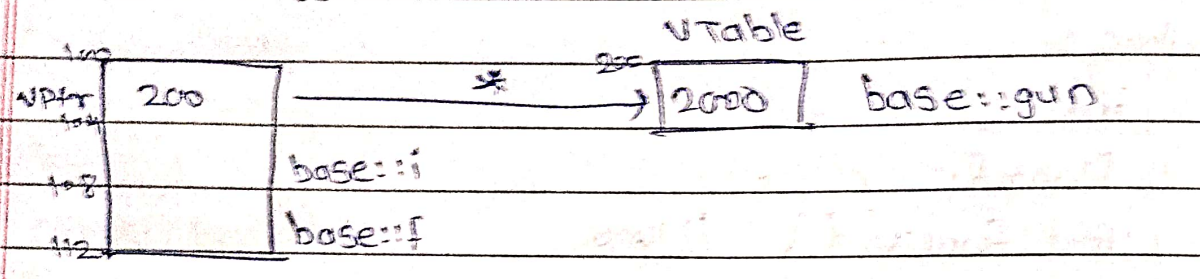
Multilevel Inheritance
Derived class inherits both base1 and base2.
The constructor of Derived will first call the constructors of base1 and base2.
Then the members of Derived are initialized.

3)
```
class base
{ public:
    int i;
    float f;
    void fun() {}  // 1000
   virtual void gun () {} // 2000
};
class Derived : public base
{ public:
    int i;
    double d;
   virtual void fun() {} // 3000
    void gun () {}  // 4000
   virtual void sun() {} // 5000
};
int main()
{ Derived dobj;
return 0;
}
```

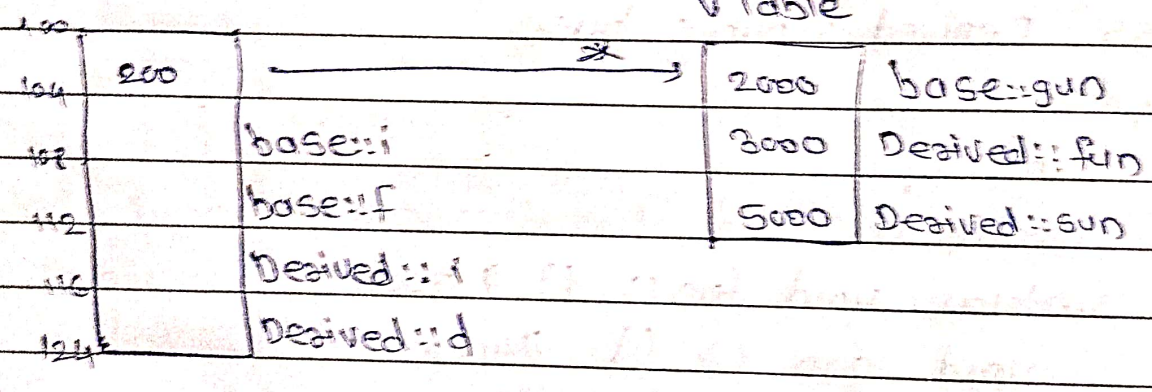
→ base has a vTable containing only gun().
Derived create a new vTable updating the function pointers.
· when derived is instantiated -
Memory is allocated for both base and derived parts.

- VTable pointer (vptr) in Derived is set to point to Derived's VTable.
- Base Constructor runs first.
- Derived Constructor runs afterward.

Base class



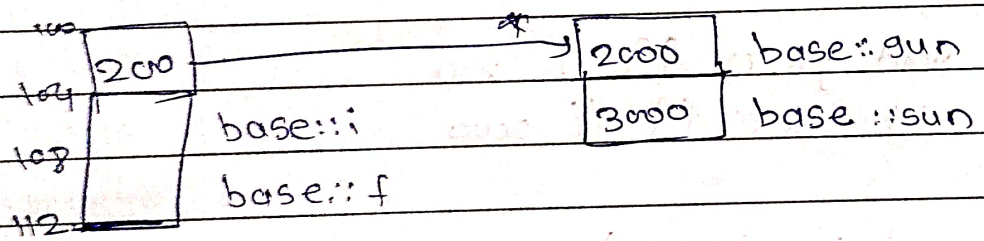
Derived class



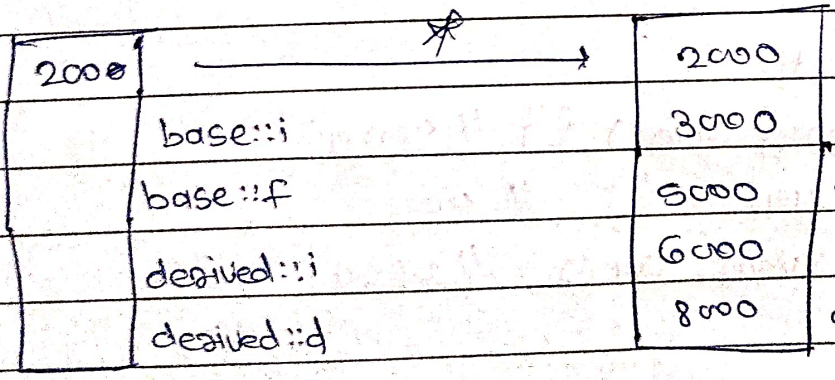
4)
```
class base
{ public:
    int i;
    float f;
    void fun() {} 1000
  virtual void gun() {} //2000
   Virtual void sun() {} //3000
    void run() {} // 4000
};
```

```
class Derived : public base
{ public:
    int i;
    double d;
   virtual void fun (){}// 5000
   virtual void gun () {} // 6000
    void sun (){} // 7000
   virtual void run () {} // 8000
};
int main()
{ Derived dobj;
return 0;
}
```

Base class



Derived class

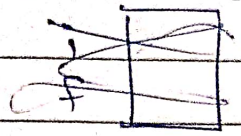


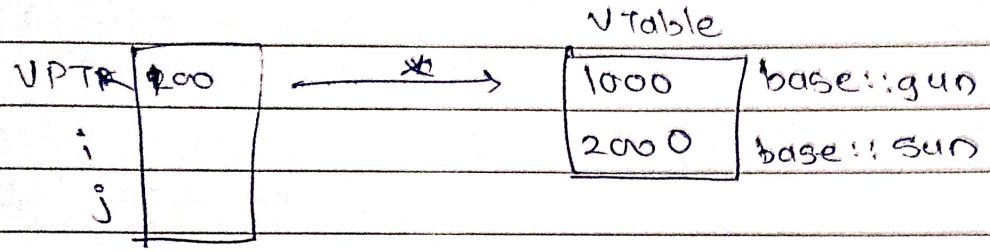
working -
The constructor of base runs first.
Then, the derived constructor executes.
The VTable pointer (vptr) is updated to point
Derived's VTable.

5] Class base
{ public:
    int i;
    float f;
  virtual void gun() {} // 1000
  virtual void sun() {} // 2000
};
class Derived : public base
{ public:
    int i;
    double d;
  virtual void fun() {} // 3000
    void gun() {} // 4000
    virtual void sun() // 5000
};
int main()
{ Derived dobj;
return 0;
}

Working -
The constructor of base runs first.
Then the derived constructor execute.
The VTable pointer is updated to point to derived's VTable.

Base class





Derived class