PAGE No.
DATE 18/10/2023

Assignment -20

Q. Draw object layout of below code snippet & explain its internal working in detail.

Q1]

```
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;
```

PAGE No.
DATE / /

bobj

100		Base::i
104		Base::f
108		Base::d
116		

dobj

100	Base::i	(int)
104	Base::f	(int)
108	Base::d	(double)
116	Derived::i	(int)
120	/// ///	Padding
124	Derived::d	(double)
132		

1] ~~The Base class~~

1] In the code we use one paradigm of Object oriented programming ~~is~~ Encapsulation. ~~In order~~ Encapsulation is binding of characteristics & behaviours. In order to achieve encapsulation we create a class.

2] Another object oriented paradigm used here is Inheritance. Inheritance in simple terms is reusability. Single-level Inheritance is used in this code. The Derived class inherits characteristics & behaviours of its Base class.

3] ~~OS~~ OS calls the entry point function & create Objects of Derived & Base class

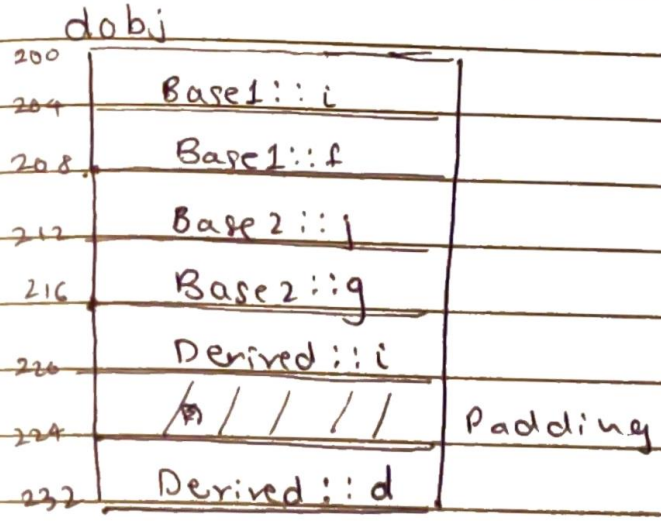
4] Object of Base class can access characteristics ~~of~~ & Behaviours of Base class only. Whereas, object of Derived class can be used to access characteristics & behaviours of ~~Ba~~ Derived class & of Base class also.

5] Padding gets added during memory allocation as memory gets allocated ~~to~~ in terms of highest largest data member. This is done by compiler to allocate memory faster. Padding is wastage of memory.

```
    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;
}
```



1] This code uses 2 object oriented paradigms

Encapsulation : Binding of characteristics & behaviours together.

Inorder to achieve encapsulation we create a class.

2] Inheritance : This code uses multiple level Inheritance

In multiple level Inheritance a single class can inherit from 2 or more classes

3] Inheritance in simple terms is reusability.

The derived class in our case inherits characteristics & behaviours from Base1 & Base 2 class

3] OS calls the entry point function & create object of derived class

4] Padding gets added during memory allocation of object. This is done due to compiler allocates memory in terms of largest data member of the object.

As double is the largest data member, memory gets allocated in terms of 8 bytes

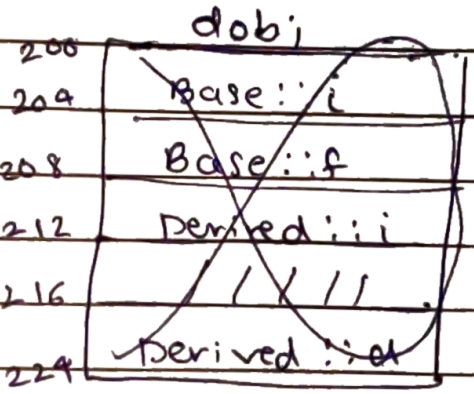
5] We return 0 to the operating system indicating success.

Q3]

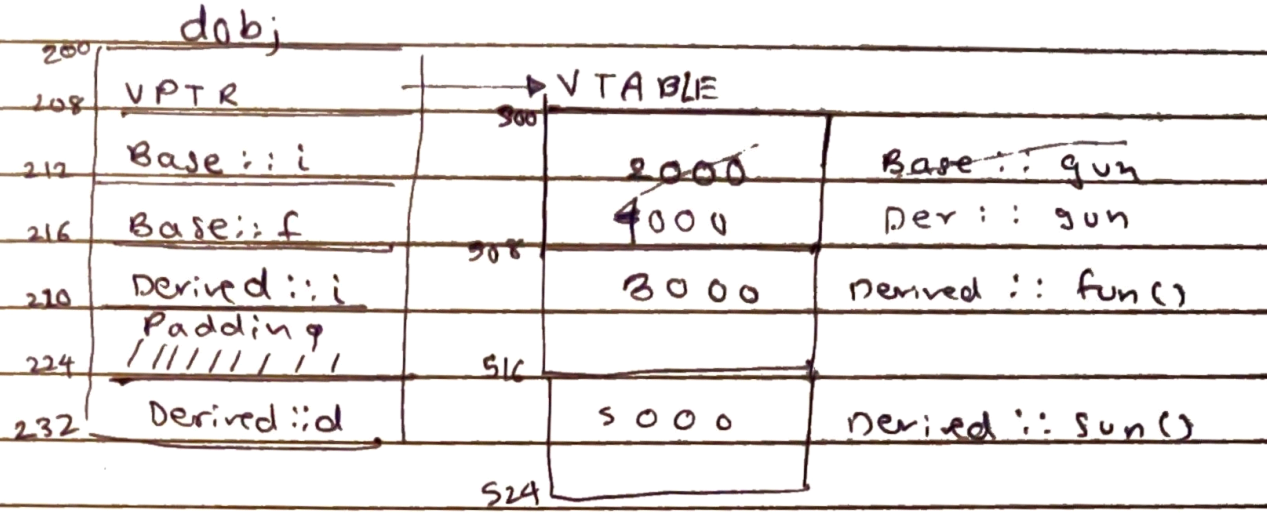
```
class base
{
    public:
        int i;
        float f;
        void fun()                  // 1000
        {}
        virtual void gun()          // 2000
        {}
};
class derived : public base
{
    public:
        int i;
        double d;

        virtual void fun()          // 3000
        {}
        void gun()                  // 4000
        {}
        Virtual void sun()          // 5000
        {}
};
int main()
{
    derived dobj;
    return 0;
}
```

Q3]
```
class base
{
public:
    int i;
    float f;
    void fun()
    {}
virtual void
```



Consider size of pointer as 8 bytes



1] In the following code we use ~~three~~ two object oriented paradigms encapsulation & Inheritance.

Encapsulation : the binding of characteristics & behaviours together. In order to achieve encapsulation we create a class

2] Inheritance : Inheritance in simple terms is called as reusability.

In our code derived class inherits characteristics & behaviours from base class

3] Whenever a class contains a virtual function or if a class is derived from such a class which contains virtual function, then first ~~four~~ 4/8 bytes of the object are reserved for the VPTR.

PAGE No.
DATE / /

4] that vptr points to a table which is called as VTABLE

5] the VTABLE holds address of all virtual functions in order in which they are defined in the class. Redefined methods override the base class functions in VTABLE

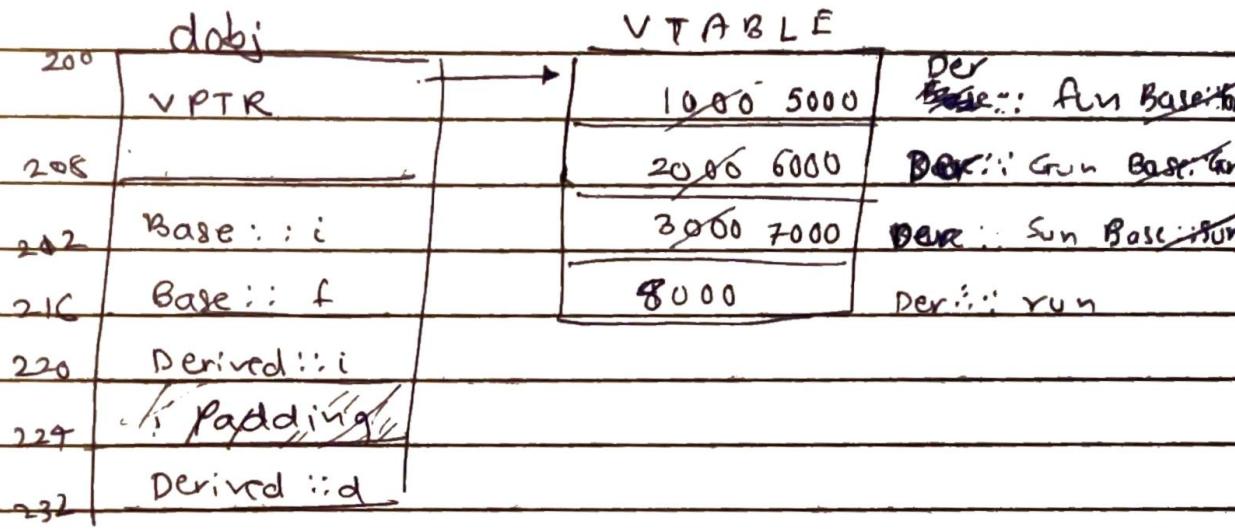
6] We return 0 to the OS indicating success

a4]

```
class base
{
  pubic :
      int i ;
      float f ;
  virtual 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 fun ()                    // 7000
{}
virtual void run ()            // 8000
{}
};

int main()
{
    derived dobj;
    return 0;
}
```



1) In this code we use two object oriented paradigms encapsulation & & Inheritance
Encapsulation : Binding of characteristics & behaviours together
Inheritance : In simple terms, Inheritance is reusability.

2) In order to achieve encapsulation we create class.
In code we create single-level Inheritance in which we derive characteristics & behaviours from Base class into the derived class.

DATE / /

3] Whenever a class contains virtual functions in it, first 4/8 bytes of its object are reserved for the VPTR. The VPTR contains address of the VTABLE.

4] The VTABLE contains all virtual functions in the sequence in which they are defined. The VTABLE will contain address of Base class virtual functions.

5] The Derived class contains redefinitions of some of the functions of Base class due to which the redefined functions will override the Base class functions.

6] Padding gets added in object as compiler allocates memory in terms of the ~~highest~~ largest data member. Since double is the largest data member memory gets allocated in terms of 8 bytes.

7] Since object dobj is of derived type the VTABLE will contain addresses of all derived class virtual functions.

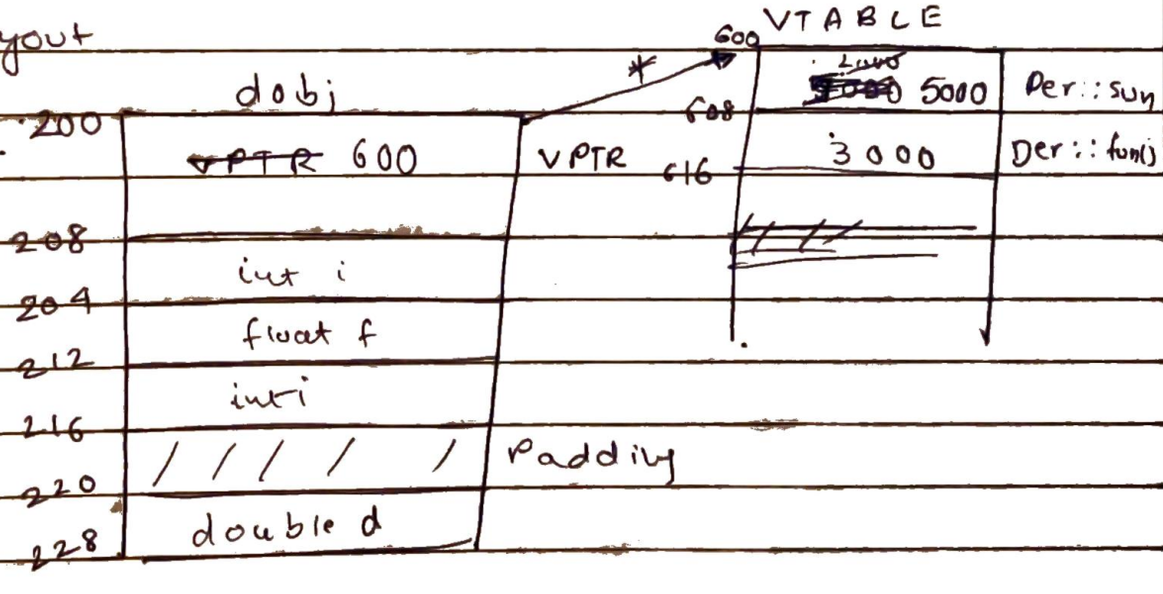
Qs]
```
class base {
    public:
        int i;
        float f;
        void fun ()          // 1000
        {}
        virtual void fun ()  // 2000
        {}
};
```

```
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;
}
```

Object layout



PAGE No.
DATE / /

1] This code contains two object oriented Paradigms
Encapsulation & Inheritance

Encapsulation : Binding of characteristics & behaviours together

Inheritance : This code uses single level Inheritance in which derived class inherits characteristics & behaviours of Base class.

2] Whenever a class contains a virtual function in it or if a class is derived from a class which contains virtual functions in it then the first 4/8 bytes are reserved for the VPTR.

3] The VPTR points to the VTABLE which contains all virtual function addresses according the sequence in which they're defined.

4] In our case
we create a derive object dobj
s

Sim: Due to inheritance everything of BAR class gets inherited along with VTABLE in the derived obj.

since it gets inherited by the derived class, the Base sun function in the VTABLE gets overridden by the derived sun due to its redefinition in the derived class

6] Therefore 2000 will be replaced by 9000 i.e address of the redefined sun function in the derived class

7] The next member of VTABLE is fun() which will be overriden only if another class would've derived from the derived class & would've contained redefinition of fun().