Assignment - 21

Q.1 why we cannot create the object of such a class which contains pure virtual function in it?

-> • Because the function does not has the body i.e the VTABLE of that class is incomplete so, our object is also incomplete therefore, we cannot create the object of class which has pure virtual function in it.

Q.2 what is meant by pure virtual function.

-> • Pure virtual function is a type of virtual function in which the parent / Base class has function declaration & initialized with zero. / Abstract method.

• for eg. - return-type function_name ( ) = 0;

• pure virtual does not have the body only declaration is done is Base and the definition is in the derived class.

eg. -

```
class Demo{
    public :
        int i;
    virtual void fun() = 0;        // Abstract method
};

class derived : public Demo
```

```
{   public:
virtual void fun()
  {
      cout << "inside fun"<<endl;
  }
}; 
int  main ()
{
    Demo * D  =  new derived;
      D  -> fun();
    return 0;
}
```

→ // Inside Fun

Q. 3 What happens if base class has contains virtual function under private access specifier?

→ • As the declaration of virtual function will be in private access specifier so, no one will access virtual function outside the class.

Q.4 What is meant by abstract method & concrete method?

→ • The function without body, we can calls it as a abstract method. Abstract method does not have the function defination.

• eg. → return-Type Method_name () = 0;

- The function with body we can call it as a concrete function.

```
return_Type   function_name ()
   {     //
   }
```

Q.5

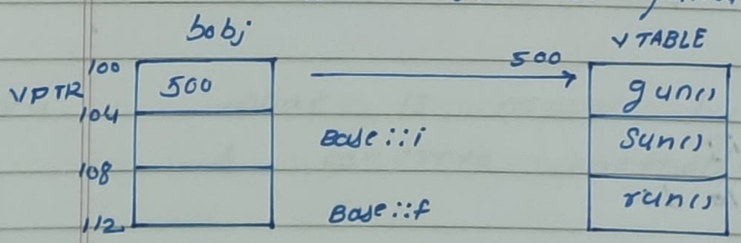
```
class   Base
{
    public:
       int i;
       int f;
     virtual void gun() = 0;
     virtual void sun() = 0;
     virtual void run() {
          cout << " Base Run";
                           }
};
class   derived : public Base
  {
       public:
          int i;
          double d;
        void  sun()
       {  cout << " Derived sun" ;
       }
        void  fun ()
     {     cout << " Derived fun" ;
     }
```

```
void gun()
{
    cout << " Derived gun";
}
```

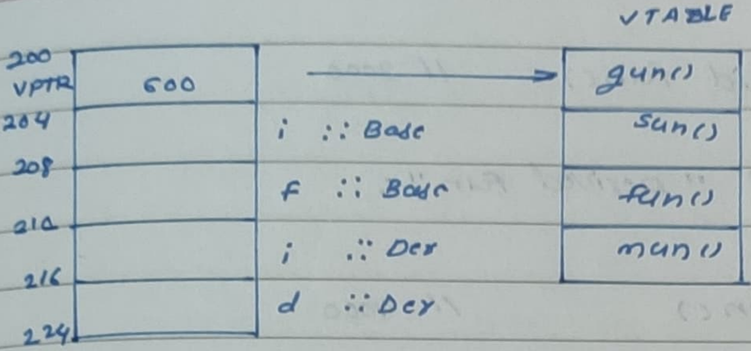
→ • In above code single level inheritance is seen.

- The Base class has 2 pure virtual functions named as sun, gun.
- one simple virtual function is also present in Base class. i.e run.
- Derived class derived all the resources from Base & this derived class has defined pure virtual functions in it.

• As base class has virtual keyword the VPTR of 4 bytes will be created when object of base will created & this VPTR points to VTABLE of base class which contain virtual function.



• The derived class will have all the resources of the base.



```
—>      Derived fun
        Derived gun
          Dervied sun
          Base run
            Derived mun
```

Q.6

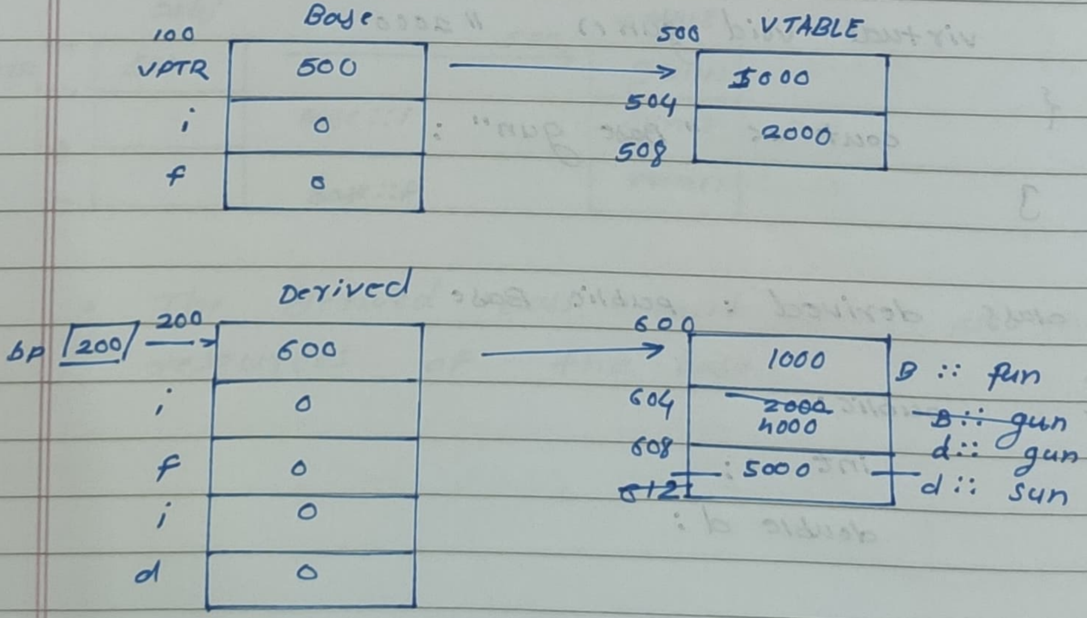
```
    class   Base
   {
        public :
           int i ;
           float f :

         void   fun()                  // 1000
      {
              cout   <<  " Base Fun" ;
      }
           virtual   void   gun()          // 2000
       {
                 cout  <<   " Base gun" ;
        }
 };

      class   derived  :   public   Base
    {
         public :
                int   i ;
                double  d :
```

```
    virtual void fun()      // 3000
    {
        cout << " Derived fun";
    }
    void gun()              // 4000
    {   cout << " Derived gun" ;
    }
    virtual void sun ()     // 5000
    {
        cout << " Derived sun" ;
    }
};
    int main ()
{
    Base * bp = new derived ;
        bp -> ~~bp~~ gun ();
        return 0 ;
}
```

—>



Q.7

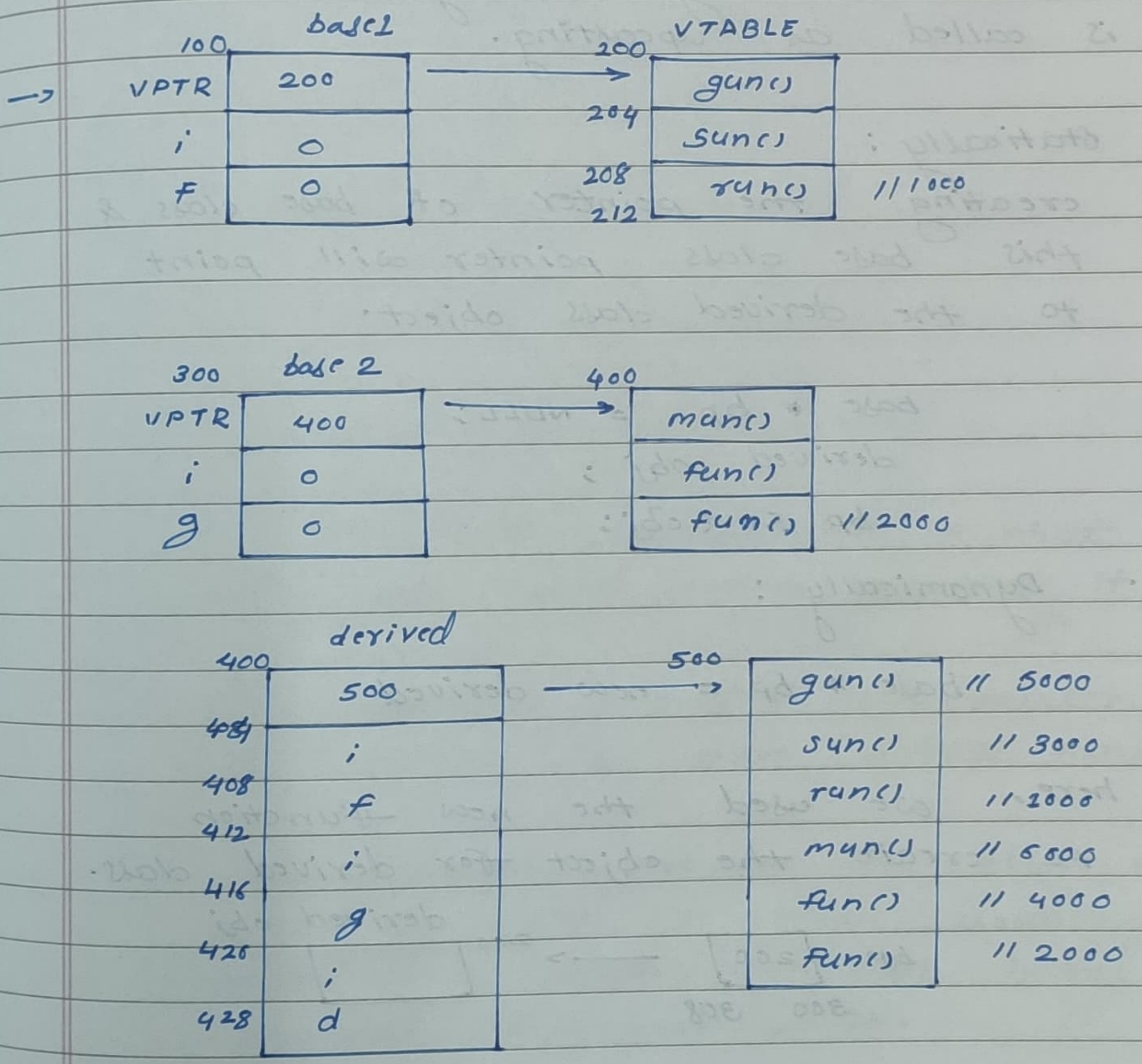
```
        class   base1
    {       public :
                int  i ;
                float f ;

                virtual  void gun () = 0 ;
                virtual  void sun () = 0 ;
                virtual  void run ()          // 1000
                    { }
    };
    class   base 2
    {
            public:
                int j ;
                float g :

            virtual  void mun()  = 0 ;
            virtual  void fun () = 0 ;
                void  fun ()          // 2000
                { }
    };
    class   derived  :  public base1 , base2
    {
            public :
                int i ;
                double d :

            void sun () { }      // 3000
            void fun() { }     // 4000
            void gun () { }      // 5000
            void  mun () { }     // 6000
    };
```

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



Q.8 What are ways in which we can achieve upcasting in object oriented language?

→ ◦ Upcasting : if a pointer having the less capacity that it points to a large capacity data then it is called as upcasting.

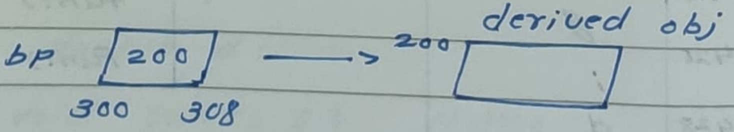
1. Statically :
- creating the pointer of base class & this base class pointer will point to the derived class object.

```
base * bp = NULL;
derived obj;
bp = &obj;
```

2. Dynamically :

```
base * bp = new derived
```

here, we used the new function to create the object for derived class.



Q.9 What is the purpose of pure virtual function.

→

- The use of pure virtual function is to declare the function in base class and defining it in the derived class.

Q.10 Can we create the pointer of that class which has pure virtual function in it.,

→

- Yes, we can create the pointer of the class which has pure virtual function in it.
- We cannot create the object of a such class but to c̶r̶eate upcasting we create the pointer of class having pure virtual function in it.