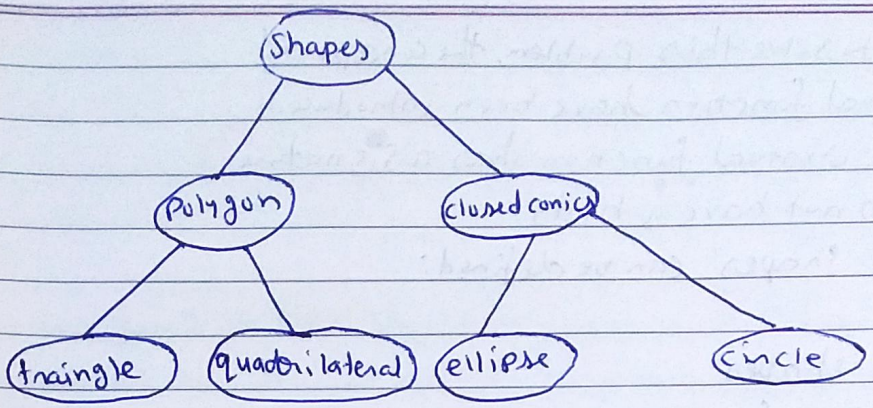

Consider the above hierarchy of classes.

```
main()
  {
     Shapes *an[] = {new traingle, new quadrilateral,
                     new ellipse, new circle};
  for(int i=0; i < sizeof(an); i++)
       an[i] -> draw();
  }
```

here, 'an' is an array of pointer to object of class 'Shapes'. However, each element of the array 'an' is pointing to the dynamically created objects of classes traingle, quadrilateral, ellipse and circle, respectively. So, it is the case of 'upcast', so no error.
If we define a draw() function in each of the bottom layer classes then depending on the properties of these classes, the code for draw() function will be different. If we want to invoke these draw() function through 'an' (as given in the code) then we need to define a draw() function in the 'Shapes' class also and should be declared as 'virtual'. However, draw() function in 'Shapes' class cannot have a particular algo.

In order to solve this problem, the concept of pure virtual function have been introduced.
A pure virtual function has a signature but does not have a body.
So, class 'Shapes' can be defined:

```
class shapes
    { public:
      virtual void draw() = 0;
    };
```

Abstract Base class

① If a class contains at least one pure virtual function then the class is called Abstract Base class (ABC).

② No instance can be created for an ABC.
If we create an object of ABC then from that object pure virtual func^n can be called but since, pure virtual func^n does not have a body, it should be prevented from calling.

③ ABC does not have a constructor or destructor.

④ An ABC may have multiple pure and non-pure virtual functions as well as data members.

⑤ Data members in an ABC should be 'protected'. Since, 'public' data members should be avoided whereas 'private' data members have no use

4th 5th

Page No.
Date

bcoz these can only be accessed from ~~base~~ member functions. But, Since member functions can only be invoked by objects ~~of the base cl~~ but there is no object for ABC, So, there is no use of private data members.

(6) member func's in ABC should be public so that they can be inherited to derived classes. However, private and protected member func's are also allowed.

(7) A derive class must override and ~~implement~~ define all pure virtual functions of the ABC otherwise pure virtual func" of the base class may become member func" of the derived class which can be called by the object of the derived class.

(8) A pure virtual function may have a body. However, the virtual func" will still remain pure. Body of the pure virtual func" is sometime needed bcoz if a common code is needed by each of the corresponding func" in derived classes. For example, in case of draw() function, let its body in base class initialites the brushes to draw something. This initialita" of brushes is actually needed in all derived classes. So, this course can be reused in derived classes.

```
class shapes { public: virtual void draw() = 0; };
    void shapes::draw()
        { cout << "Initialite brushes" << endl; }

class traingle : public polygon
    { public: void draw(); };
void traingle::draw()
    { shapes::draw();
      cout << "traingle drawing";
    }
```