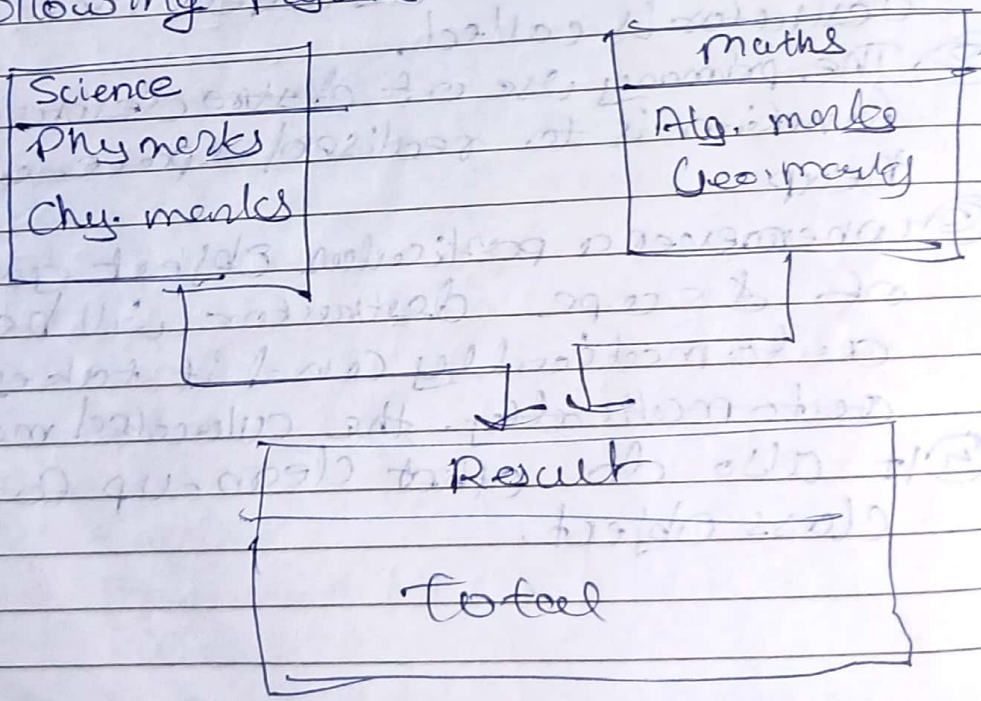
Page No.
Date

Unit III : Extending class using Inheritance
(weightage - 16 marks)

Q1 Write a C++ Program to declare class college with name and code. derived an new class as student with members name. Accept and display details of student along with college details. - (4m)

Q2 Describe visibility mode and their effects used in inheritance - 4m/6m

Q3 write a C++ program to Implement following figure.



- 6m

Scanned with CamScanner

Page No.
Date

03/10/2023

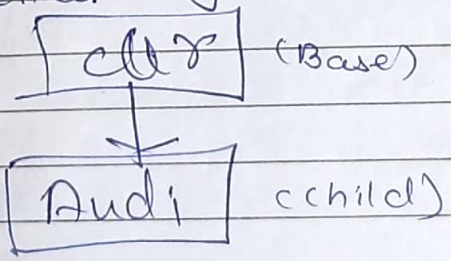
Inheritance and it's type

* In OOP, it is possible for a base class to have one or many derived classes and for one derived class to have one or many base classes.

Types of inheritance

- Single inheritance
  - A derived class with only one base class is called Single inheritance

Example -

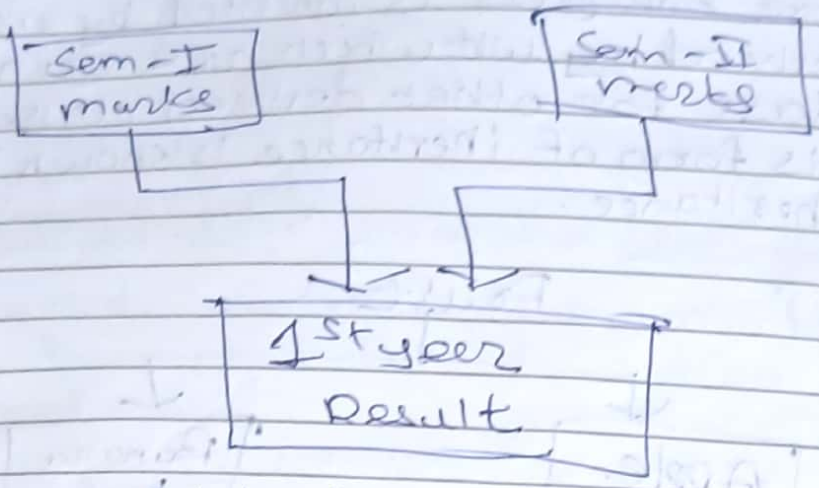


Scanned with CamScanner

22316

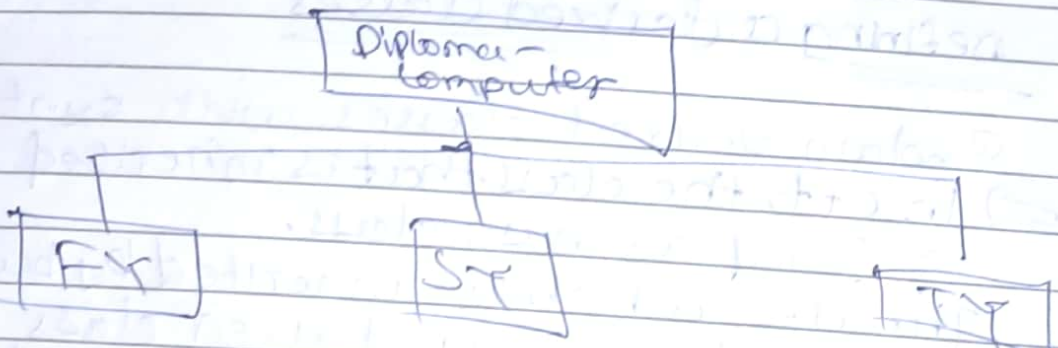
Multiple ~~level~~ inheritance

A derived class with more than one base class is called multiple ~~level~~ inheri inheritance



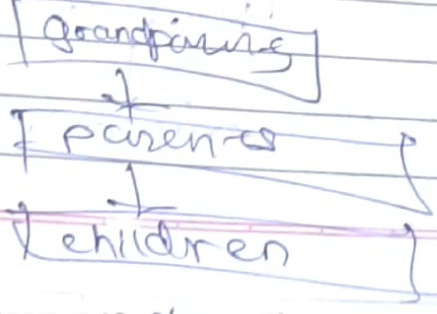
Hierarchical inheritance

If one base class is inherited by more than one derived classes is called Hierarchical inheritances.



Multilevel inheritance (2m)

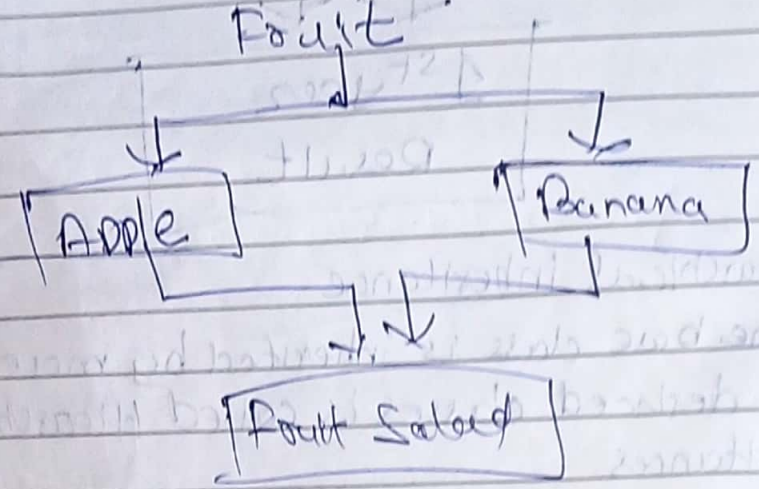
The mechanism of deriving a class from another derived class is known as multilevel inheritance.



Scanned with CamScanner

Hybrid inheritance

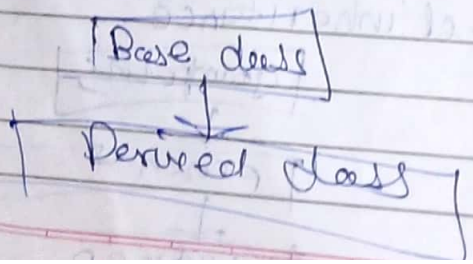
It is combination single, multiple, hierarchical multilevel inheritance.
Here one class is inherited by many derived ~~by~~ wh which are again base class for other derived classes.
This form of inheritance is known hybrid inheritance.



Defining a derived classes

Explain derived classes with syntax?
⇒ In c++, the class that is inherited is refered as base class.
The class which is inherited by base class is known as derived class.
Derived class is known as child class.
Base class is known as parent class.

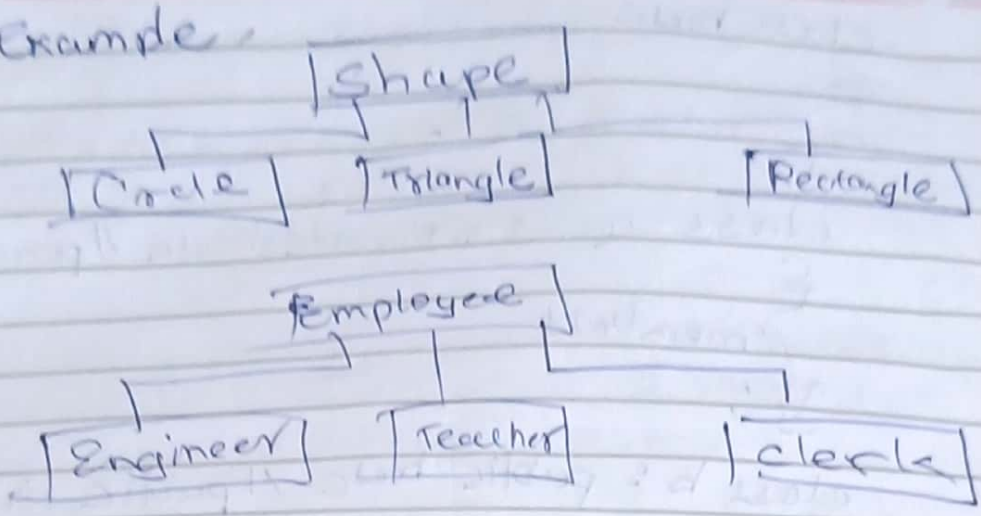
Representation



Scanned with CamScanner

22316

Example:



When a class inherits another member of the base class becomes members of the derived class

Syntax:

```
class derived class : visibility_mode base_class
{
   :
   :
//member of derived class
   :
   :
};
```

example

```
class engineer : : Public Employee
{
};
```

* above syntax contain
class class_name (visibility mode is option (default - Private) & name of base class.

* example

ab

Scanned with CamScanner

Page No.
Date

```
class base
{
};
class a : private base // private derivation
{
// members
};
class b : public base // public derivation
{
// members
};
class c : protected base // protected derivation
{
// members
};
class d : base // by default private
{
// members
};
```

3/10/23

Scanned with CamScanner

22316

Visibility modes and effects

Q// Explain visibilities modes in inheritance.
Q// Explain visibilities mode and its effect.
Q// List visibilities mode.

- Visibility modes used in inheritance are ① Private ② Public ③ Protected.
- The visibility mode decide whether the inherited features of the class will be public, private & protected members of derived class.

Base class visibilities	Derived class visibility		
	Private	Protected	Public
Private	not inherited	not inherited	not inherited
Protected	inherited	inherited	inherited
Public	inherit	inherit	inherit

Private: Where a base privately inherited by derived class, public and protected members of base class becomes private members of derived class.
Therefore, public and protected members of the base class can only be accessed by member function of derived class but cannot be accessed by objects of derived class.

Scanned with CamScanner

Syntax

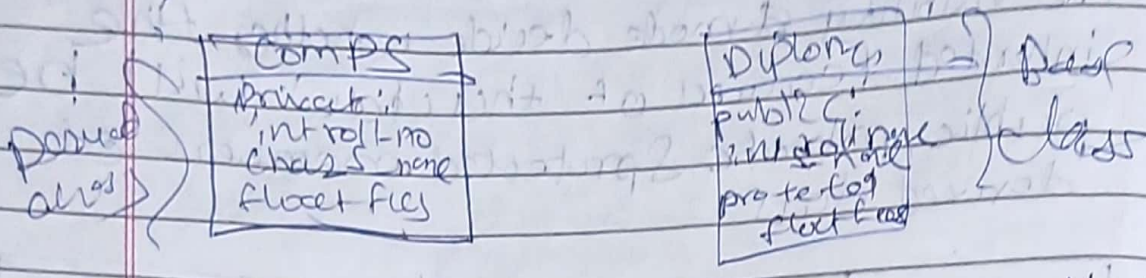
class comps: public Diploma

Class derived : Private Base

{

// members of derived class

};



public : When a base class is publicly inherited by derived class then protected members of base class becomes protected member of derived class. public members of base class becomes public members of derived class. Therefore, public members of base can be accessed by both the member function and object of Derived class

Class derived : public base

{

// members of derived class

};

class comp : public Diploma.

22316

protected : when a base class protected by inherited by derived class public and protected members of base class become protected member of derived class. Therefore, the public and protected members of the base class can be accessed by member function of derived class. But they cannot be accessed by object of derived class.

: casting

5/10/2023

Page No.
Date

Virtual Base class

- Consider a student situation where multilevel, multiple and hierarchical all three kind of inheritance are involved

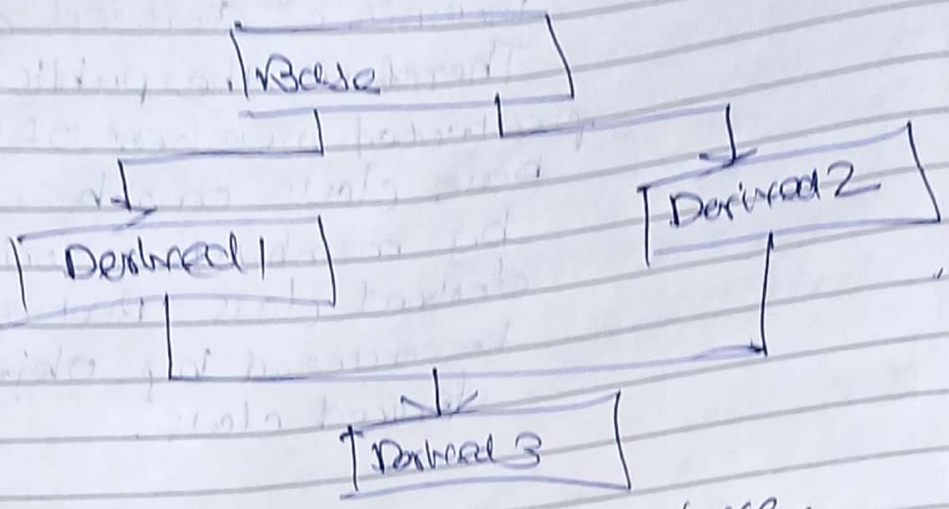
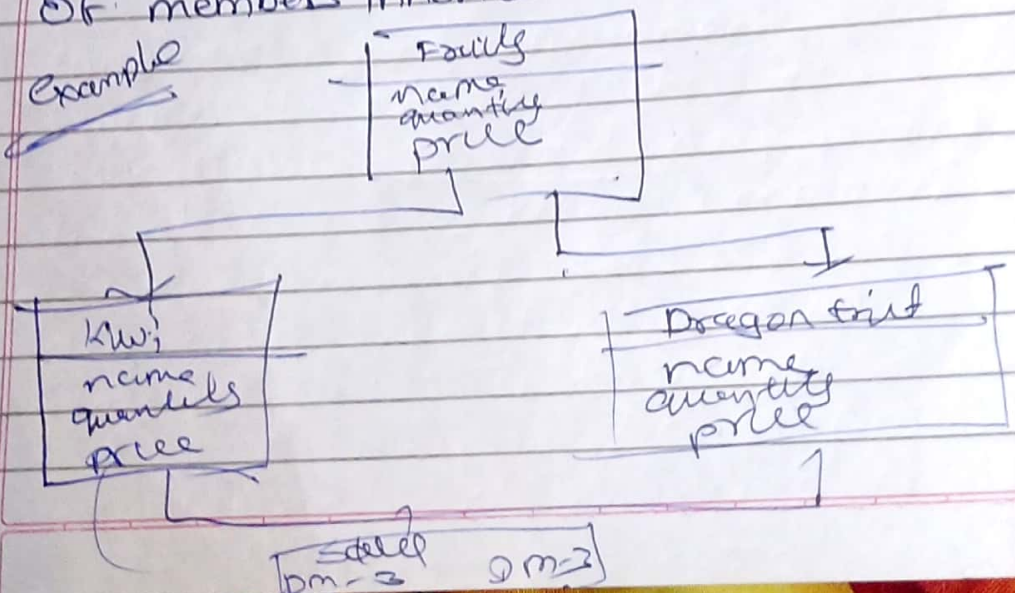


figure: multipath Inheritance.

- In this figure, base class inherited by both Derived 1 and Derived 2 class all the public and protected members are inherited into Derived 3 twice through both derived 1 and derived 2 class.
- Therefore, Derived 3 would have duplicate sets of members inherit.

Example



Scanned with CamScanner

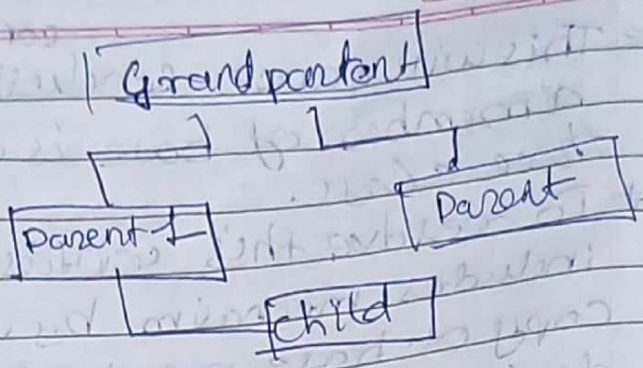
22316

* This will cause confusion/ambiguity when a member of base is used by derived three class.
* To resolve this confusion, C++ involves/includes mechanism by which only one copy of base members will be included in derived three class.
* This feature is called virtual Base class.
* When class is made virtual, C++ necessary care to inherit only one copy that class.
* The keyword virtual precedes the base class access specifier when it is inherited by derived class.
* In short, C++ has a mechanism for elimination of multiple copies of base class members called virtual base class

```
class Base
{
};
class Derived 1 : virtual public base
{
};
class Derived 2 : virtual public base
{
};
class Derived3 : public derived1, public
                 derived2
{
};
```

07/10/2023

example,




```
#include<iostream>
using namespace std;
{

};
class Parent1 : virtual public Grandparent
{

};
class Parent 2 : virtual public Grandparent
{

};
class child : public Parent1, public parent2
{

};
```

* Consider hybrid inheritance as shown in the figure the child class has two direct base classes parent 1 and parent 2 which themselves have a common base class as grand parent.
The child inherits the members Grandparent via two seperate paths all the public and protected members of grandparent are inherited into child twice.

Scanned with CamScanner

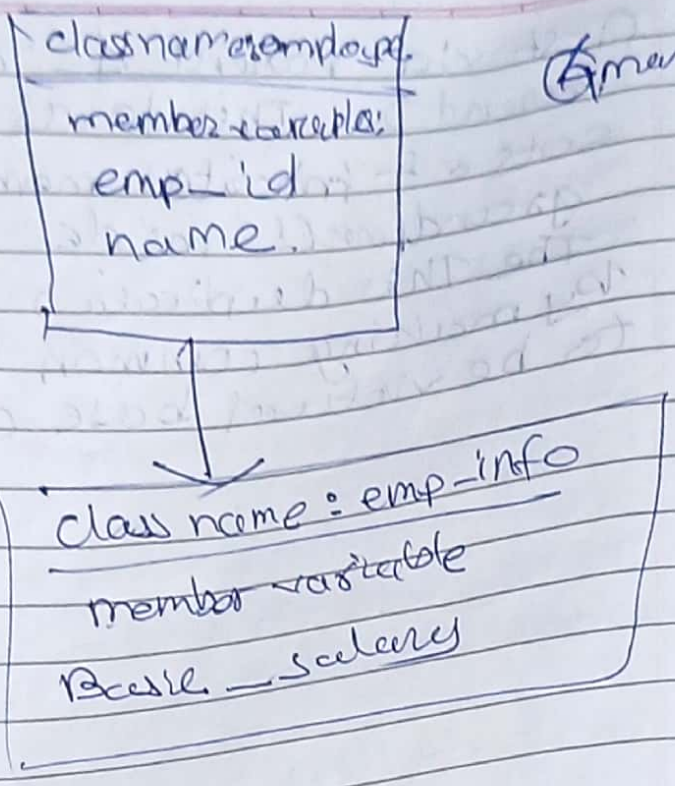
22316

Page No.
Date

first via Parent 1 and again via Parent 2. This leads to duplicate sets of inherited members of childrens grandparents inside the class.
The This duplication can be avoided by making common base class known to be virtual base class.

Scanned with CamScanner

* WAP (Single inheritance)




```
#include <iostream>
using namespace std;
class employee
{
public:
int emp_id;
char name[10];
};
class emp_info : public employee //inheritance
{
int basic_salary;
public:
void getdata()
{
cout<<"Enter emp_id";
cin>>emp_id;
cout<<"Enter name";
cin>>name;
cout<<"Enter Salary";
cin>>basic_salary;
```

Scanned with CamScanner

22316

```
void putdata()
{
cout << "Emp_id=" << emp_id;
cout << "Emp_name=" << name;
cout << "Salary=" << basic_salary;
};
void main()
{
  emp_info e;
  e.getdata();
  e.putdata();
}
```

Output

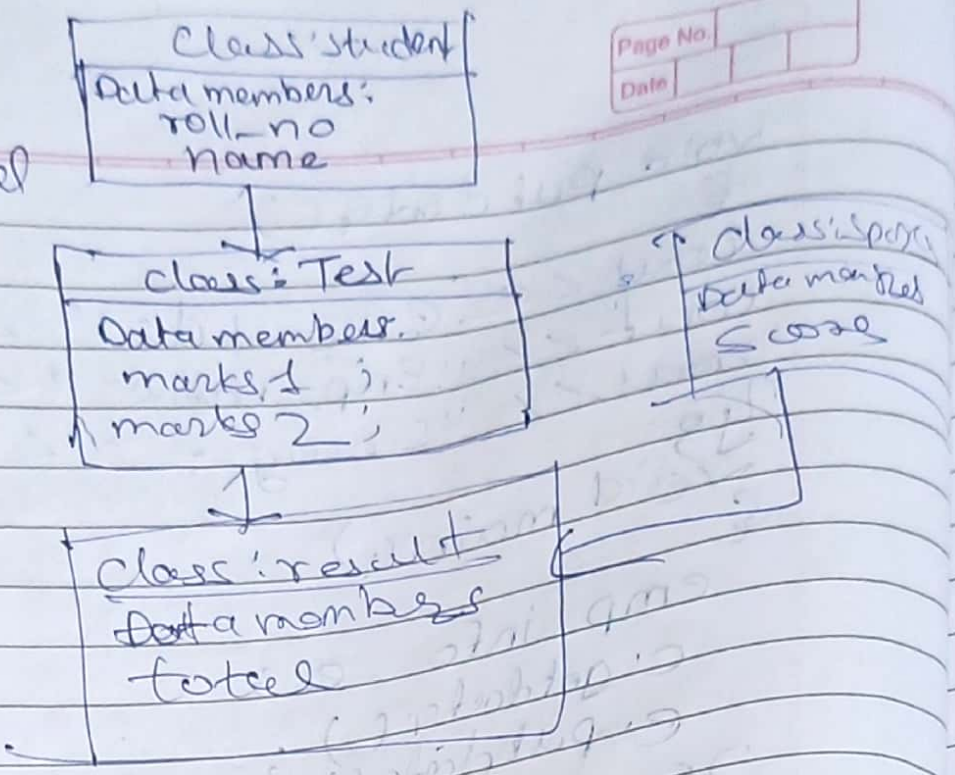
```
Emp
Enter emp_id : 1320
Enter name : Elvish Bhai
Enter salary = 69000
Emp_id = 1320
Emp_name : Elvish Bhai
Emp_salary : 69000
```

Scanned with CamScanner

WAP

Q2)

Hierarchical using Scoped resolution operator




```
#include <iostream>
using namespace std;
class student
{
 public:
 int roll_no;
 char name[50];
};
class test : public student
 {
  public:
  int marks1, marks2;
 };
class sports
  {
   public:
    int score;
  };
Class result : public sports, public test
 {
  public:
   int total;
```

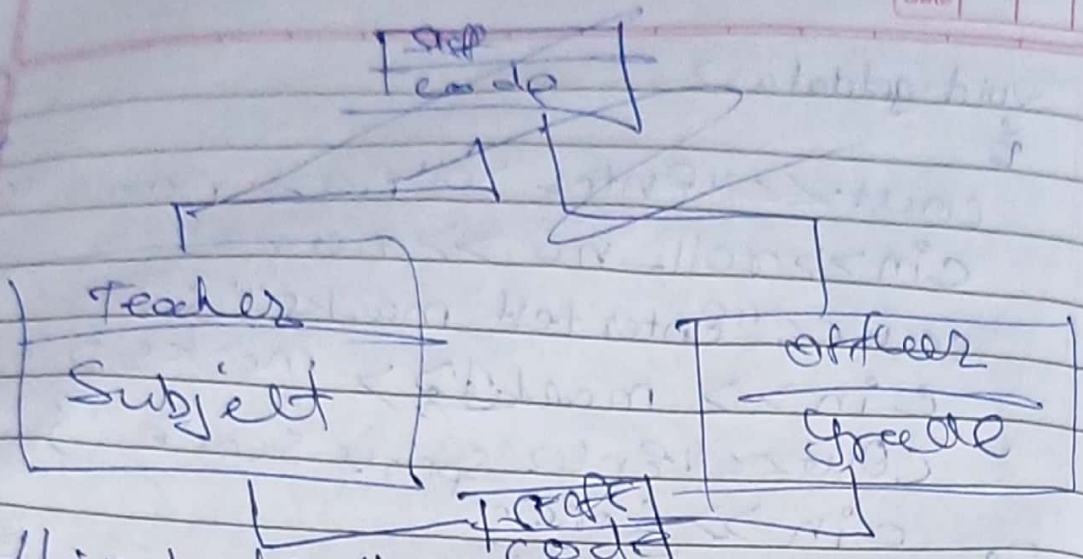
Scanned with CamScanner

22316

```
void getdata()
{
  cout<<"Enter roll no & name";
  cin>>roll_no>>name;
  cout<<"Enter test marks";
  cin>>marks1>>marks2;
  cout<<"Enter sports score";
  cin>>score;
}
void putdata()
{
cout<<"Roll no & name"<<roll_no<<name;
cout<<"Marks"<<marks1<<marks2;
cout<<"Score"<<score;
int total;
cout<<"total="<<total=marks1+marks2;
}
int main()
{
result x;
x.getdata();
x.putdata();
return 0;
};
```

2 casting

Output

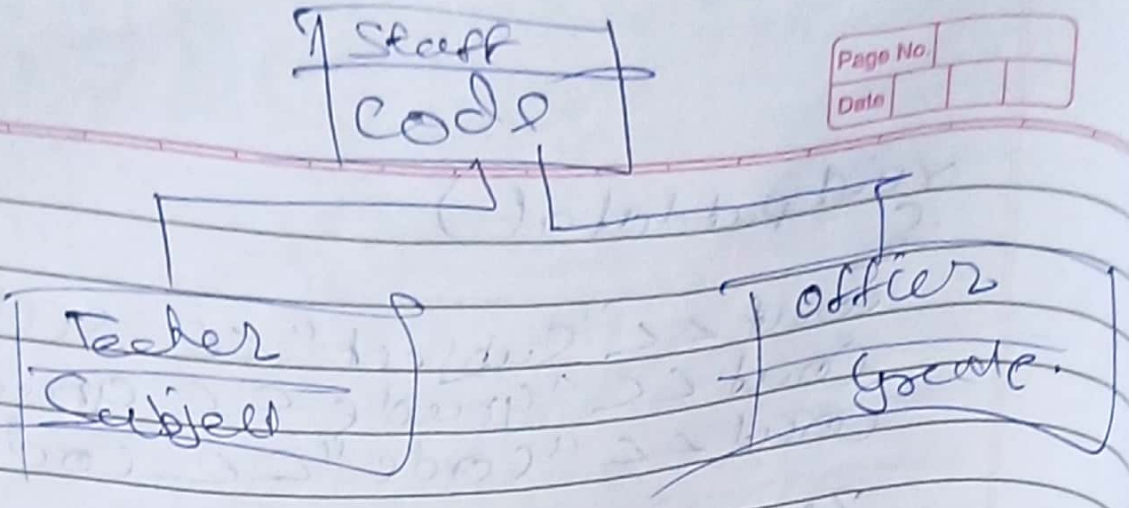



```
#include <iostream>
using namespace std;
class teacher{
  {
    public:
    int sub
    char sub[30];
  };
class officer
  {
  public:
  char grade[10];
  }
class staff : public officer, public teacher
  { public;
    int code;
  void getdata();
    {
      cout<<"Enter subject";
        cin>>sub;
      cout<<"Enter grade";
      cin>>grade;
      cout<<"Enter code:";
    cin>>code;
```

Scanned with CamScanner

```
void putdata()
{
cout << "Subject" << sub;
cout << "grade" << grade;
cout << "code" << code;
}
};
int main()
{
~~getdata x;~~
teacher x;
x.putdata()
x.~~put~~getdata()
return 0;
}
```

Scanned with CamScanner



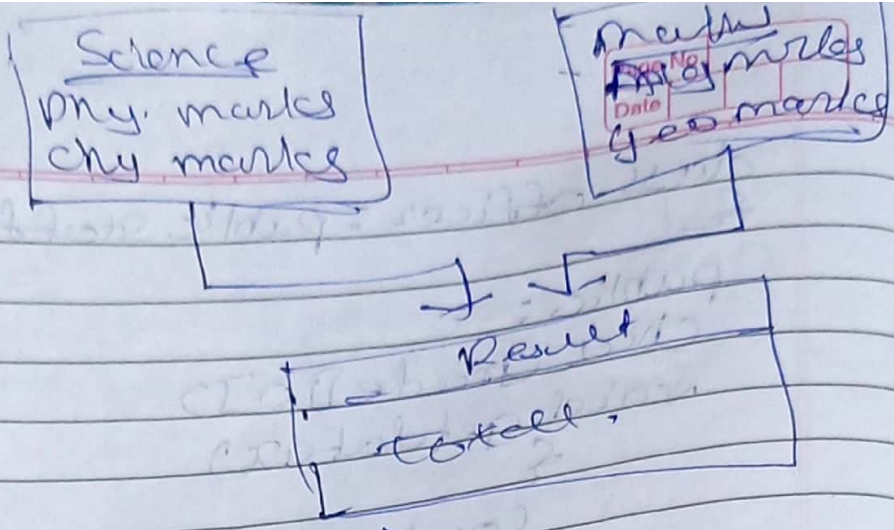

```
#include <iostream>
using namespace std;
class staff
{
public:
int code;
};
class teacher: public staff
{public:
char subject[50];
void getdata()
{
cout<<"Enter code";
cin>>code;
cout<<"Enter subject";
cin>>subject;
}
void putdata()
{
cout<<"Teacher";
cout<<"code="<<code
subject=<<subject
}
};
```

Scanned with CamScanner

22316

```
class officer : public staff
{
public:
char Grade[20];
void getdata2()
    {
     cout << "Enter Code ";
     cin >> code;
     cout << "Enter Grade ";
     cin >> grade;
    }
void putdata2()
    {
     cout << "Officer ";
     cout << "Grade & Code" << grade
          << code;
    }
};
int main()
{
  teacher X;          teacher x;
  officer Y;          officer y;
  X.getdata1();       x.getdata1();
  Y.getdata2();       x.putdata1();
  X.putdata1();       y.getdata2();
  Y.putdata2();       y.putdata2();
  return 0;           return 0;
}                     }
```

Scanned with CamScanner



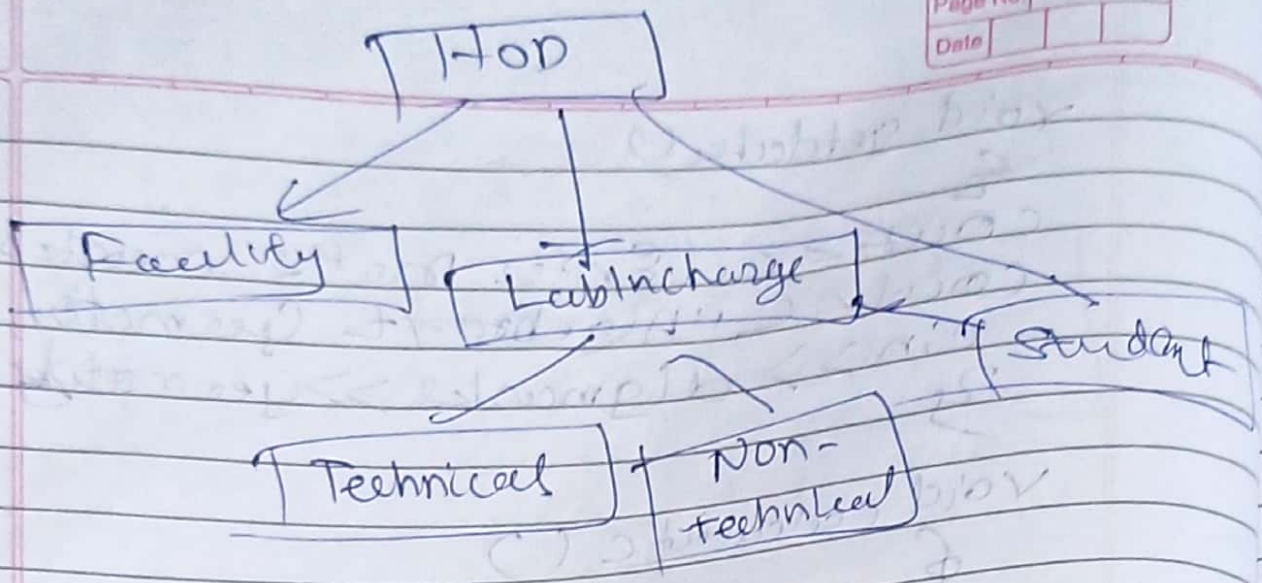

```
#include<iostream>
using namespace std;
class Science
{
public:
int phymarks;
int chymarks;
};
class maths
{
public:
int phamarks;
int geomarks;
};
class Result : public science, public
                maths.
{
public:
int total;
void putdata()
{
cout<<"Enter Sciences Marks
cout<<"Physics: \t chemistry:
cin>> phymarks >>chymarks;
cout<<"Science Phy:\t"<<phy<<
                chy\t  chy
```

Scanned with CamScanner

```
void getdata()
{
cout<<"Enter Maths marks";
cout<<"Algebra/t Geometry";
cin>>algmarks>>geomarks;
}
void calculate()
{
int maths= algmarks+geomarks;
int science=phymarks+chymarks;
int total = maths +science;
cout<<" total = "<<total;
}
};
int main()
{
    Result z;
    z.putdata();
    z.getdata();
    z.calculate();
return 0;
}
```

Scanned with CamScanner

Q1




```
#include <iostream>
using namespace std;
class HOD
{
  int public:
    char name[50];
};

class faculty : public HOD
{ public:
  char name[50];
  char sub[20];
  void getdata()
  {
    cout<<"Enter name of HOD";
    cin>>name;
    cout<<"Enter faculty name";
     cin>>fname;
    cout<<"Enter subject";
    cin>>sub;
  }
  {void putdata()
  {
    cout<<name;
    cout<<fname;
    cout<<sub;
};
```

Scanned with CamScanner

22316

```
Class student : public HOD
{
public
Char cname[50];
int rollno;
void getdata_1()
{
cout<<"Enter HOD name";
cin>>name;
cout<<"Enter name of student;
cin>>sname;
cout<<"Enter roll no";
cin>>roll_no;
}
void putdata_1()
{
cout<<name;
cout<<sname;
cout<<roll_no;
};
Class LabIncharge : public HOD
{
public:
charlname[50];
char subject[50];
};
class technical : public LabIncharge
{ public:
void getdata_2()
{
cout<<"Enter HOD name";
cin>>name
cout<<"Enter Lab Incharge name";
cin>>lname;
cout<<"enter technical sub";
```

Scanned with CamScanner

```
    cin>>L_subject;
  }
  void putdata_2()
  {
    cout<<name;
    cout<<lname;
    cout<<L_subject;
  }
};
class Nontechnical : public labIncharge
{
  public:
    void getdata_3()
    {
      cout<<"Enter HOD name";
      cin>>name;
      cout<<"Enter lab incharge name";
      cin>>lname;
      cout<<"Enter non-technical sub";
      cin>>L_subject;
    }
    void putdata_3()
    {
      cout<<name;
      cout<<lname;
      cout<<L_subject;
    }
};
```

Scanned with CamScanner

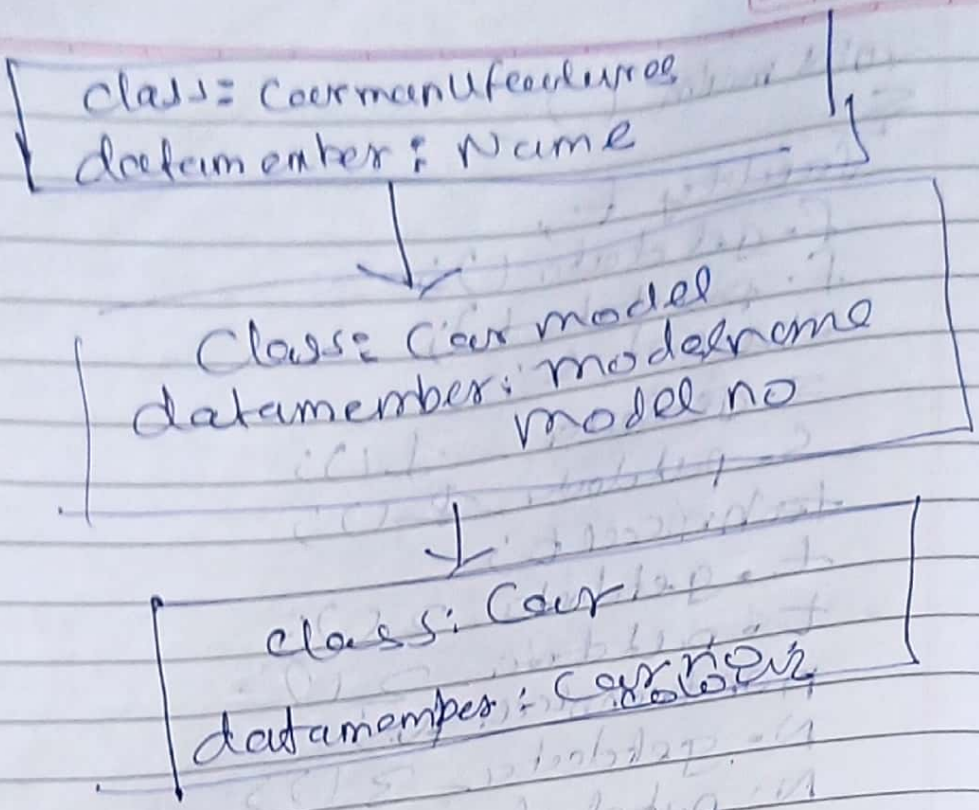
22316

```
_subject;
data_2()
name;
name;
L_subject;

public labIncharge

3()

nter HODname";
e;
-lab incharge name";
e;
er non-technical sub";
subject;

s()

me;
me;
subject;
```

```
void main()
{
  faculty f;
  f.getdata();
  f.putdata();
  student s;
  s.getdata_1();
  s.putdata_2();
  technical t;
  t.getdata_2();
  t.putdata_2();
  nontechnical N;
  N.getdata_3();
  N.putdata_3();
  return 0;
}
```

Scanned with CamScanner

Q1/



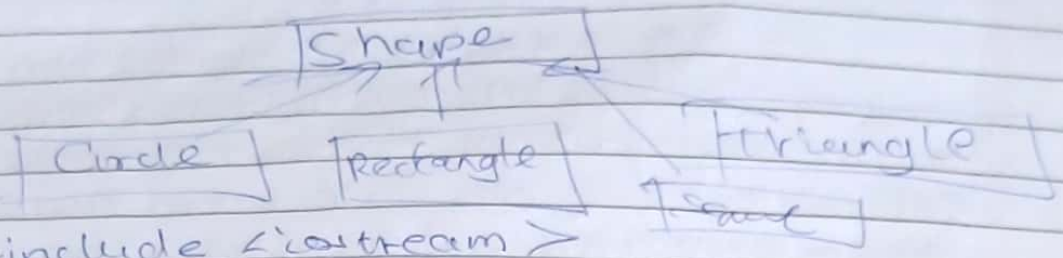
11/10/23

Abstract classes

* A class that combines atleast one pure virtual function is called abstract class.
* ~~class atleast consist~~ An abstract class does not create any object and it contains one or more function for which there is no definitions.
* A pure member virtual function is a member function that is declared as an abstract class but defined in derived class
* A class used only for deriving other classes is called Abstract class.
* An abstract class is designed only to act as base class that is to be inherited by other classes.
* A class containing pure virtual function is called as Abstract class.
* Abstract class is named by because we cannot define objects of a class containing

22316

a pure virtual function
* For example, shape is an abstract class which derives square, circle and rectangle.




```
#include <iostream>
using namespace std;
class shape
{
  public:
  float l;
  void getdata ()
      { cout<<"enter l";
        cin>>l;
      }
  virtual float area ();
};
class square : public shape
{
  float area ()
  {
     float a;
     a= l*b;     //return l*l;
     cout<< a;
  }
};
```

Scanned with CamScanner

```
class Circle : public Shape
{
  public float area()
  {
    float b;
    b = 3.14 * r * r;
    cout << b;
  }
};
int main()
{
  Square SX;
  Circle CPX;
  SX.getdata();
  CPX.getdata();
  CPX.area();
  return 0;
}
```

Scanned with CamScanner