3.3 Object Oriented Programming (C++)

Total marks: 75 (Semester end examination - 60, Internal assessment - 15)

Date

Single level Inheritance.

* Program of single level Inheritance using Public derivation.

```
#include <iostream.h>
#include <conio.h>

class base
  {
      private :
                int x;  // private data member
                        so not inheritable

      public :
                int y;  // public data member
                        so inheritable.
                void enter();  // member function.
                int return_x();
                void disp_x();

  };
```

Date

```
// definition of function enter()

void base :: enter()
  {
     cout << "\n  Enter the values of x and y:";
     cin >> x >> y;
  }

// definition of function return_x()

int base :: return_x()
  {
     return x;
  }

// definition of function disp_x()

void base :: disp_x()
  {
     cout << "\n\n  x = " << x << endl;
  }
```

Total marks: 75 (Semester end examination - 60, Internal assessment - 15)

Date

```
class derived : public base        // public derivation
  {
    private :
              int z;    // private data member
    public :
              void add();   // member function
              void display();
  };
// function definition add()
void derived :: add()
  {
     z = y + return_x();        // return_x() returns private data of base
  }
// definition of function display()
void derived :: display()
  {
     cout << "\n\n x = " << return_x();
     cout << "\n\n y = " << y << endl;
     cout << "\nZ = " << z << endl;
  }
```

Date

```
void main()
{ derived obj;
  clrscr();
  obj.enter();
  obj.add();
  obj.disp_x();
  obj.display();
  obj.y = 50;   // public data member y
                   of base class initialized
  obj.add();       (accessed) using derived class
  obj.display();   object obj.
  getch();
}.
```

output: Enter the values of x and y: 10 20

x = 10	x = 10
y = 20	y = 50
z = 30	z = 60.

Here, the class derived inherits the class base publically. The private data of base class is not inheritable.

The derived class object obj is used to access the member functions and data members of the base class, whereas the derived member functions are used to access the private data of base class.

Total marks 75 (... Internal assessment 15)

Date

Multiple Inheritance.

Multiple inheritance allows us to combine the features of several existing classes into a derived class.

The syntax of a derived class with multiple base classes is :

```
class derived : visibility Base1 , visibility Base2 ,---
    {
      - - - -
      - - - -   // Body of derived class.
    };
```

where visibility may be either public, private or protected and the base classes are comma separated.

* Program of multiple Inheritance using public derivation.

```
#include <iostream.h>
#include <conio.h>

class ONE
  {  protected :
       int x ;  // to make it available to the derived class
```

Date

```
public :
        void disp_1(void)
        {
            cout << x << "\n";
        }
};
class TWO
{
    protected :
        int y; // to make it
               available to the
               derived class.
    public :
        void disp_2(void)
        {
            cout << y << "\n";
        }
};
```

Date

```
// multiple base class inheritance
class derived : public ONE, public TWO
    {
        public:
            void enter (int a, int b)
                {
                    x = a;
                    y = b;
                }
    };
void main ()
    {
        int value1, value2;
        derived obj;  // object of derived class declared.
        clrscr();
        cout << " Enter two integer values: ";
        cin >> value1 >> value2;
        obj. enter (value1, value2);  // function call.
        cout << "\n You may have entered: \n\n";
        obj. disp1();
        obj. disp-2();
    }
```

output: Enter two integer values
80 60
You have entered:
80
60

Date

Another example.

If we had a specific class to print on screen (Output) and we wanted our classes CRectangle and CTriangle to also inherit its members in addition to those of CPolygon we could write:

```
class CRectangle : public CPolygon , public COutput,
                   {
                   };

class CTriangle : public CPolygon, public Coutput
                  {
                  };

#include <iostream.h>
class CPolygon
    { protected :
                  int width , height;
      public :
                  void set_values (int a, int b)
                    {
                      width = a;
                      height = b;
                    }
    };
```

Date

```
class Output
{
    public:
        void output(int);
};
void Output :: output(int i)
{
    cout << i << endl;
}
class Rectangle : public Polygon, public Output
{
    public : int area()
    {
        return (width * height);
    }
};
class Triangle : public Polygon, public Output
{
    public : int area()
    {
        return (width * height/2);
    }
};
```

Date

```
int main()
{
    Rectangle rect;
    Triangle tgl;
    rect.set_values (4,5);
    tgl.set_values (4,5);
    rect.output (rect.area());
    tgl.output (tgl.area());
    return 0;
}
```

output : 20
10.

Date

Hierarchical Inheritance. (Protectedly derived class and accessibility of members from objects and within drived class (es))

```
// Illustration of protected derivation.
// Illustration of hierarchical inheritance.
#include <iostream.h>
#include <stdio.h>   // for gets()
#include <conio.h>
const LEN = 41;
class publication
    {
        protected :  // to provide access to derived classes.
                char title [LEN];
                float price;
    };
class book : protected publication
    { private : int pages;
      public :
                void getdata(void)
                { cout << "\n Enter the title :";
                gets (title);
                cout << "\n Enter the price :";
                cin >> price;
```

Date

```
cout << "\n Enter the number of pages : ";
cin >> pages;
}
void putdata (void)
{ cout << "\n Title : " << title;
  cout << "\n Price : " << price;
  cout << "\n Pages : " << pages;
}
};
class tape : protected publication
{ private :
        int play-time;
  public : void getdata (void)
        { cout << "\n Enter the title : ";
          gets (title);
          cout << "\n Enter the price : ";
          cin >> price;
          cout << "\n Enter the playtime
                  in minutes : ";
          cin >> play-time;
        }
```

Date

```
void putdata (void)
  { cout <<"\n Title :" <<title;
    cout <<"\n Price :" << price;
    cout <<"\n Play-time :" <<play-time <<" minutes";
  }
};

void main ()
  { book obj1;
    tape obj2;
    clrscr();
    obj1.getdata();   // enter the data
    obj2.getdata();
    obj1.putdata();   // display the data
    obj2.putdata();
  }
```

Output :

Enter the title: Programming in C

Enter the price: 150

Enter the number of pages : 460

Date

Enter the title : Firewall media.
Enter the price : 500
Enter the playtime in minutes : 150.

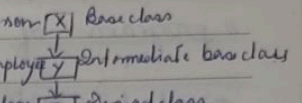
Title : programming in C
price : 150.
pages : 460.

Title : Firewall media.
price : 500.
play-time : 150 minutes.

In the above program the protected members (title and price) of class publication become protected member in both the derived classes i.e. class book and class tape.
The data members of the base class are accessed by the member functions of the derived classes which are invoked using the objects of these classes.

Date

4) Multilevel Inheritance.



The multilevel inheritance can be declared as →

```
class X
    {  // base class members.
    };
class Y : public X
    {  // Y derived from X
    };
class Z : public Y
    {  // Z derived from Y.
    };
```

ex. The information about the salary of an employee is stored in three different classes.

class person stores the name and age,
class employee stores the designation, basic pay, da, hra and cca. and
class calc-salary contains the total salary of the employee.
The class calc-salary can inherit the details of the employee and the name and age through multilevel inheritance.

Date

// program of multilevel inheritance

```
#include <iostream.h>
#include <stdio.h>  // for gets()
#include <conio.h>
const SIZE = 31;
class person.
{ protected :
        char name[SIZE];
        int age;
  public :
        void getdata1(void);
        void putdata1(void);
};
void person :: getdata1(void)
{
    cout << "Enter name of person";
    gets (name);
    cout << "Enter the age:";
    cin >> age;
}
```

to provide access to the derived class

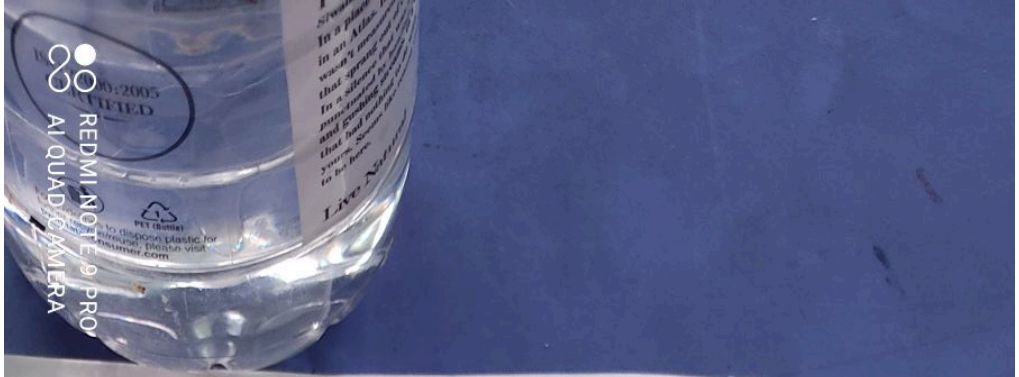
Date

```
void person :: putdata1 (void)
  {  cout <<" Name = " << name <<"\n";
     cout <<" Age  = " << age <<"\n";
  }

class employee : public person
  {  protected :
                char desig [size];
                float basic , da , hra , cca ;

     public :   void getdata 2(void);
                void putdata 2 (void);

  };
void employee :: getdata2 (void)
   {  cout <<" Enter designation:" ;
      gets (desig);
      cout <<" Enter the basic pay : ";
      cin >> basic ;
      cout << " Enter the dearness allowance:";
      cin >> da ;
      cout << " Enter the house rent :";
      cin >> hra ;
      cout << " Enter the city compensatory allowance :";
      cin >> cca ;
   }
```



Date

```
void employee :: putdata2(void)
{   cout << " Designation : " << desig << "\n";
    cout << " Basic pay : " << basic << "\n";
    cout << " Dearness allowance : " << da << "\n";
    cout << " House rent : " << hra << "\n";
    cout << " City compensatory allowance : " << cca << "\n";
}

class calc_salary : public employee
{   float total_salary;
    public:
        void display(void);
};

void calc_salary :: display()
{   total_salary = basic + da + hra + cca;
    putdata1();
    putdata2();
    cout << " Total salary : " << total_salary << "\n";
}
```

Date

```
void main()
{   clrscr();
    calc_salary obj;
    obj.getdata1();
    obj.getdata2();
    cout<<"\n\n.....output.....\n\n";
    obj.display();
}
```

Date

5) Hybrid Inheritance.

Combining of multilevel and multiple Inheritance

The example of calculating the total salary of an employee. Assume that some additional benefit is given to the employee in case of an award winning (state/national or international) performance.

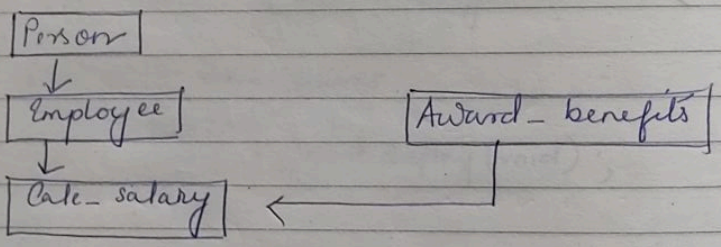


fig: Multilevel, Multiple Inheritance.

Program of Hybrid Inheritance.

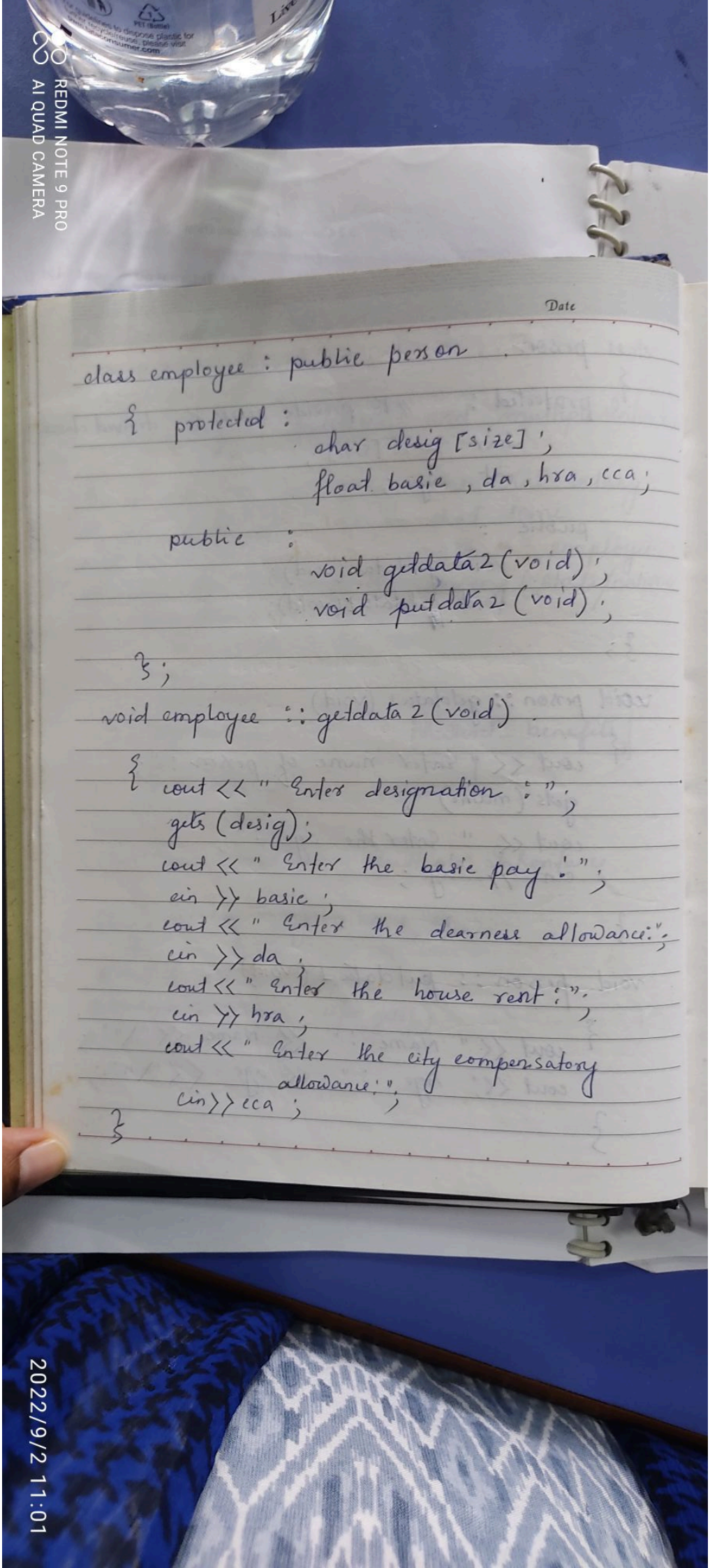
```
#include <iostream.h>
#include <stdio.h> //for gets()
#include <conio.h>
  const size = 31;
```

Date

```
class person
{ protected :    // to provide access to derived classes
      char name [size];
      int age;

  public :
      void getdata1(void);
      void putdata1(void);

};
void person :: getdata1 (void)
{ cout << " Enter name of person : ";
  gets (name);
  cout << " Enter the age :";
  cin >> age;
}
void person :: putdata1 (void)
{ cout << " Name :" << Name << "\n";
  cout << " Age :" << age << "\n";
}
```



Date

```
class employee : public person
{ protected :
        char desig [size] ;
        float basic , da , hra , cca ;
  public :
        void getdata2 (void) ;
        void putdata2 (void) ;
};
void employee :: getdata 2 (void)
{
  cout << " Enter designation : " ;
  gets (desig) ;
  cout << " Enter the basic pay : " ;
  cin >> basic ;
  cout << " Enter the dearness allowance: " ;
  cin >> da ;
  cout << " Enter the house rent : " ;
  cin >> hra ;
  cout << " Enter the city compensatory
          allowance : " ;
  cin >> cca ;
}
```

Date

```
void employee :: putdata2 (void)
  { cout << "Designation : " << desig << "\n";
    cout << " Basic pay : " << basic << "\n";
    cout << " Dearness allowance : " << da << "\n";
    cout << " House rent : " << hra << "\n";
    cout << " City compensatory allowance : " << cca << "\n";
  }
class award_benefits
  { protected :
        float special_increment ;
    public :
        void getdata3 (void);
        void putdata3 (void);
  }
void award_benefits :: getdata3(void)
  { cout << " Enter the special increment (if any): ";
    cin >> special_increment ;
  }
```

Date

```
void award_benefits :: putdata3 (void)
  { cout << "The special award increment :";
    cout << special_increment << "\n\n";
  }

// hybrid derivation

class cale_salary : public employee, public award_benefits
  {
    float total_salary; // private by default
    public :
          void display (void);

  };

void cal_salary :: display ()
  {
    total_salary = basic + da + hra + cca +
                   special_increment;
    putdata1(); // function call putdata1()
    putdata2(); //  "    "  putdata2()
    putdata3(); //  "    "  putdata3()

    cout << "Total salary :" << total_salary << "\n";
  }
```

Date

```
void main()
{ classcr();
  calc_salary obj ;  // object declared for class
                        calc_salary
  // enter the data
    obj. getdata1();
    obj. getdata2();
    obj. getdata3();
  // display the data
    cout << "\n\n * .... output .... *\n\n";
    obj. display ();
}
```

output : →

Enter name of person : D. Konwar.
Enter the age : 26.
Enter designation : Principal.
Enter the basic pay : 12000.
Enter the dearness allowance : 4800
Enter the house rent : 500
Enter the city compensatory allowance : 300.
Enter the special increment (if any) : 2000.

Date

output

Name : D. konDar.
Age : 26.
Designation : principal.
Basic pay : 12000.
Dearness allowance : 4800.
House rent : 500
City canpensatory allowance : 300.
The special award increment : 2000.
Total salary : 19600.