OOP LABORATORY 8

Name: **ANIRBAN HAZRA**
Section: **B-12**
Roll : **2005643**

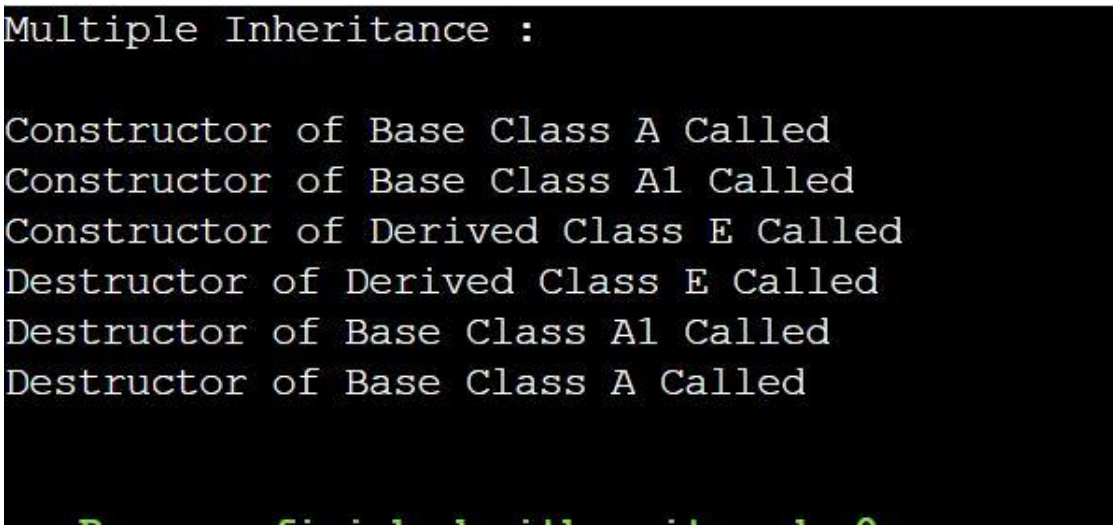
1. WAP to demonstrate the order of call of constructors and destructors in case of multiple inheritance.

PROGRAM CODE:-

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

class A
{
   protected:

   int a;
   public:

   A()
   {
      cout << "Constructor of Base Class A Called " << endl;
   }

   ~A()
   {
      cout << "Destructor of Base Class A Called " << endl;
   }

};

class A1
{
   protected:

   int a1;
   public:

   A1()
   {
      cout << "Constructor of Base Class A1 Called " << endl;
   }
```

```
    ~A1()
    {
        cout << "Destructor of Base Class A1 Called " << endl;
    }

};

class E : public A, public A1
{
    protected:
    int e;

    public:

    E()
    {
        cout << "Constructor of Derived Class E Called " << endl;
    }

    ~E()
    {
        cout << "Destructor of Derived Class E Called " << endl;
    }

};

int main()
{
    cout << "Multiple Inheritance : \n" << endl;
    E obj;

    return 0;
}
```

OUTPUT:-



2. WAP to demonstrate the order of call of constructors and destructors in case of multi-level inheritance.

PROGRAM CODE:-

```
#include<iostream>
using namespace std;
class A
{
   protected:
         int a;
   public:
   A()
   {
          cout << "Constructor of Grandparent Class A Called " << endl;
   }
   ~A()
   {
          cout << "Destructor of Grandparent Class A Called " << endl;
   }
};
class B : public A
{
   protected:
         int b;
   public:
   B()
   {
         cout << "Constructor of Parent Class B Called " << endl;
   }
   ~B()
   {
         cout << "Destructor of Parent Class B Called " << endl;
   }
};
class D : public B
{
   protected:
         int d;
   public:
   D()
   {
         cout << "Constructor of Child Class D Called " << endl;
   }
   ~D()
   {
         cout << "Destructor of Child Class D Called " << endl;
   }
};
```

```
int main()
{
   cout << "Multilevel : \n" << endl;
   D obj;
   return 0;

}
```

OUTPUT:-

```
Multilevel :

Constructor of Grandparent Class A Called
Constructor of Parent Class B Called
Constructor of Child Class D Called
Destructor of Child Class D Called
Destructor of Parent Class B Called
Destructor of Grandparent Class A Called
```

3. WAP to demonstrate the order of call of constructors and destructors in case of virtual base class .

PROGRAM CODE:-

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

class A
{

   public:

      A()
      {
         cout << "Constructor of Grandparent Class A Called " << endl;
      }

      ~A()
      {
         cout << "Destructor of Grandparent Class A Called " << endl;
      }

};
```

```
class B : public virtual A
{
   public:

      B()
      {
         cout << "Constructor of Parent Class B Called " << endl;
      }
      ~B()
      {
         cout << "Destructor of Parent Class B Called " << endl;
      }
};

class C : virtual public A
{
   public:

      C()
      {
         cout << "Constructor of Parent Class C Called " << endl;
      }
      ~C()
      {
         cout << "Destructor of Parent Class C Called " << endl;
      }
};

class D : public B, public C
{
   public:

      D()
      {
         cout << "Constructor of Child Class D Called " << endl;
      }
      ~D()
      {
         cout << "Destructor of Child Class D Called " << endl;
      }
};


int main()
{
   cout << "Multipath :\n" << endl;
   D obj;

   return 0;
}
```

OUTPUT:-

```
Virtual Base Classes :

Constructor of Grandparent Class A Called
Constructor of Parent Class B Called
Constructor of Parent Class C Called
Constructor of Child Class D Called
Destructor of Child Class D Called
Destructor of Parent Class C Called
Destructor of Parent Class B Called
Destructor of Grandparent Class A Called
```

4. Extend the program ii. of inheritance to include a class sports, which stores the marks in sports activity. Derive the result class from the classes 'test' and 'sports'. Create objects using parameterized constructors .Calculate the total marks and percentage of a student.

PROGRAM CODE:-

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

class student
{
protected:

        char name[20];
        int roll,age;
public:
  void getdata ()
  {

          cout << "Enter roll and name and age" << endl;
          cin >> roll >> name >> age;

  }

};

class test:public student
{
protected:
        int sub1;
        int sub2;
```

```
        int sub3;
        int sub4;
        int sub5;

public:

  void getmark ()
  {

        cout << "Enter 5 subjects marks : " << endl;
        cin >> sub1 >> sub2 >> sub3 >> sub4 >> sub5;

  }
  void details ()
  {

        cout << "\n\nName : " << name << " Roll number : " << roll << endl;
        cout << "Marks in 5 subjects : " << sub1 << ", " << sub2 << ", " << sub3;
        cout<< ", " << sub4 << ", " << sub5 << endl;
        cout<<"\nAge : "<< age;

 }

};

class sports
{
protected:

        int msports;
public:

  void getspo ()
  {

        cout << "Enter marks in sports : ";
        cin >> msports;

  }
};

class result:public sports, public test
{
        int total;
        float percent;
public:

  result(int tot, float per)
  {
        total=tot;
```

```
        percent=per;
  }
  void display ()
  {
        cout << "\nMarks in sports = " << msports << endl;
        total = sub1 + sub2 + sub3 + sub4 + sub5 + msports;
        percent = (total * 100) / 600;
        cout << "Total marks : " << total << "\nPercent = " << percent << endl;
  }
};

int
main ()
{
        result ob1(0,0);
        ob1.getdata ();
        ob1.getmark ();
        ob1.getspo ();
        ob1.display ();
        ob1.details ();
        ob1.display ();
        return 0;

}
```

OUTPUT:-

```
Enter roll and name and age
643
Anirban
18
Enter 5 subjects marks :
88 89 90 91 92
Enter marks in sports : 90

Marks in sports = 90
Total marks : 540
Percent = 90


Name : Anirban Roll number : 643
Marks in 5 subjects : 88, 89, 90, 91, 92

Age : 18
Marks in sports = 90
Total marks : 540
Percent = 90
```

5. Rewrite the assignment vii. From Inheritance including the parameterized constructors in all the classes.

PROGRAM CODE:-

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

class Account
{
   protected:
      int custno;
      string custname;
      int balance;
   public:
      Account(int cno,string cname,int bal)
      {
         custno = cno;
         custname = cname;
         balance = bal;
      }
};

class Savings : public Account
{
   protected:
      int minbalance;
   public:
      Savings(int minbal, int cno, string cna, int bal): Account(cno,cna,bal)
      {
         minbalance = minbal;
      }


      void depositSavings()
      {
         int dep;
         cout << "Enter Amount to deposit in Savings Account : ";
         cin >> dep;
         balance += dep;
      }
      void withdraw()
      {
         int with;
         cout << "Enter Amount you want to Withdraw from Savings Account : ";
         cin >> with;
         if(balance-with < minbalance)
            cout << "Amount can't be withdrawn as you will not be left with minimum";
            cout<<"balance ... " << endl;
```

```
        else
        {
            balance -= with;
            cout << "Amount Withdrawn Successfully... Collect your Cash…";
            cout <<"\nRemaining Balance = " << balance << endl;
        }
    }
    void Display()
    {
        cout << "\nSavings Account :--\nCustomer number = " << custno <<"Name=";
        cout<< custname << " Balance Remaining = " << balance << endl;
    }
};

class Current : public Account
{
  protected:
    int overdue;

public:
    Current(int ovdue, int cno, string cna, int bal): Account(cno,cna,bal)
    {
        Account(387,"Hamdan",1000);
        overdue = ovdue;
    }
    void depositCurrent()
    {
        int dep;
        cout << "Enter Amount to deposit in Current Account : ";
        cin >> dep;
        balance += dep;
    }
    void withdraw()
    {
        int with;
        cout << "Enter Amount you want to Withdraw from Current Account : ";
        cin >> with;
        if(balance-with < overdue)
            cout << "Amount can't be withdrawn as you will not be left with Over-due
            cout <<"amount ... " << endl;
        else
        {
            balance -= with;
            cout << "Amount Withdrawn Successfully... Collect your Cash…
            cout<<\nRemaining Balance = " << balance << endl;
        }
    }
    void Display()
    {
        cout << "\nCurrent Account :--\nCustomer number = " << custno<<"Name =";
```

```
        cout<< custname << " Balance Remaining = " << balance << endl;
    }
};
int main()
{
   int cno, bal;
   string nam;
   cout << "Enter Customer number, Name and Balance : ";
   cin >> cno >> nam >> bal;

   Savings obs(500,cno,nam,bal);
   Current obc(500,cno,nam,bal);

   obs.depositSavings();
   obs.withdraw();

   obc.depositCurrent();
   obc.withdraw();

   obs.Display();
   obc.Display();

   return 0;
}
```

OUTPUT:-

```
Enter Customer number, Name and Balance : 3452 Dushyant 500000
Enter Amount to deposit in Savings Account : 5000
Enter Amount you want to Withdraw from Savings Account : 4000
Amount Withdrawn Successfully... Collect your Cash...
Remaining Balance = 501000
Enter Amount to deposit in Current Account : 5000
Enter Amount you want to Withdraw from Current Account : 1000
Amount Withdrawn Successfully... Collect your Cash...
Remaining Balance = 504000

Savings Account :--
Customer number = 3452 Name = Dushyant Balance Remaining = 501000

Current Account :--
Customer number = 3452 Name = Dushyant Balance Remaining = 504000
```