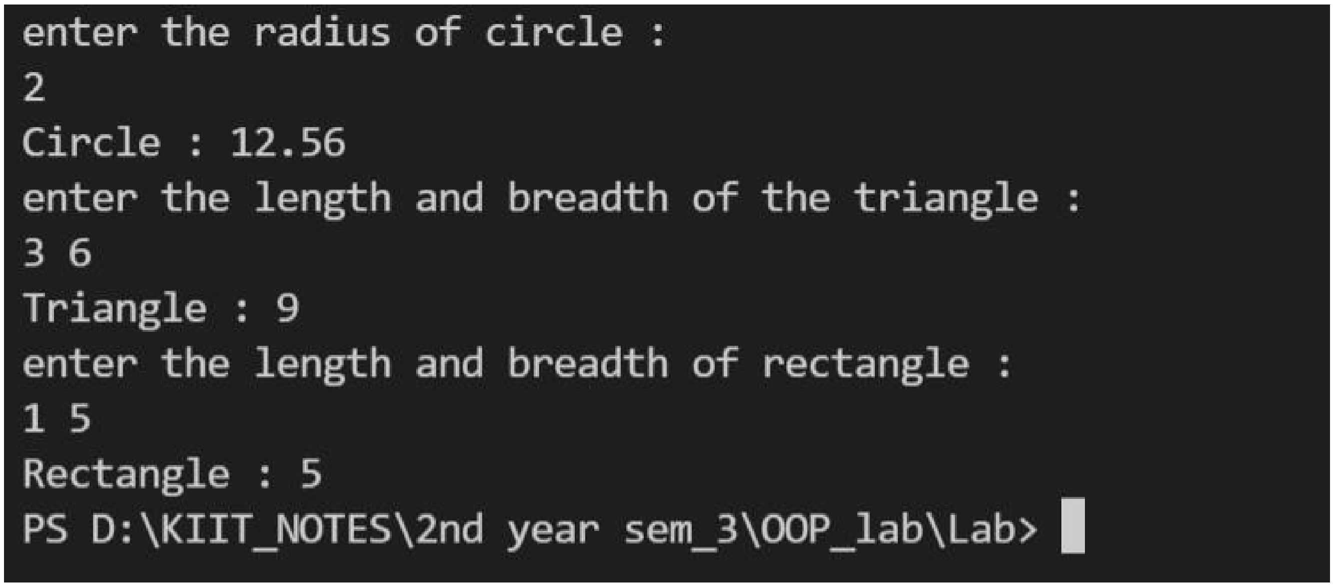
Name - Akriti Choudhary
Roll number - 2005776
Lab9
Subject - OOP lab
Class - B14
Branch - CSE
Date- 11/10/2021

Question1)Create a class ‘shape’. Derive three classes from it: Circle, Triangle and Rectangle. Include the relevant data members and functions in all the classes. Find the area of each shape and display it.

```
#include <iostream>
using namespace std;
class shape
{
protected:
   float areac;
   float areat;
   float arear;
};
class circle : public shape
{
public:
   void carea()
   {
      int a;
      cout << "enter the radius of circle :\n";
      cin >> a;
      areac = 3.14 * a * a;
      cout << "Circle : " << areac << endl;
   }
};
class triangle : public shape
{
public:
   void tarea()
   {
      int a;
      int b;
      cout << "enter the length and breadth of the triangle :\n";
      cin >> a >> b;
      areat = 0.5 * a * b;
      cout << "Triangle : " << areat << endl;
   }
};
class rectangle : public shape
{
public:
   void rarea()
   {
      int a;
      int b;
      cout << "enter the length and breadth of rectangle :\n";
      cin >> a >> b;
      arear = a * b;
      cout << "Rectangle : " << arear << endl;
   }
};
```

```
int main()
{
   circle obc;
   obc.carea();
   triangle obt;
   obt.tarea();
   rectangle obr;
   obr.rarea();
   return 0;
}
```

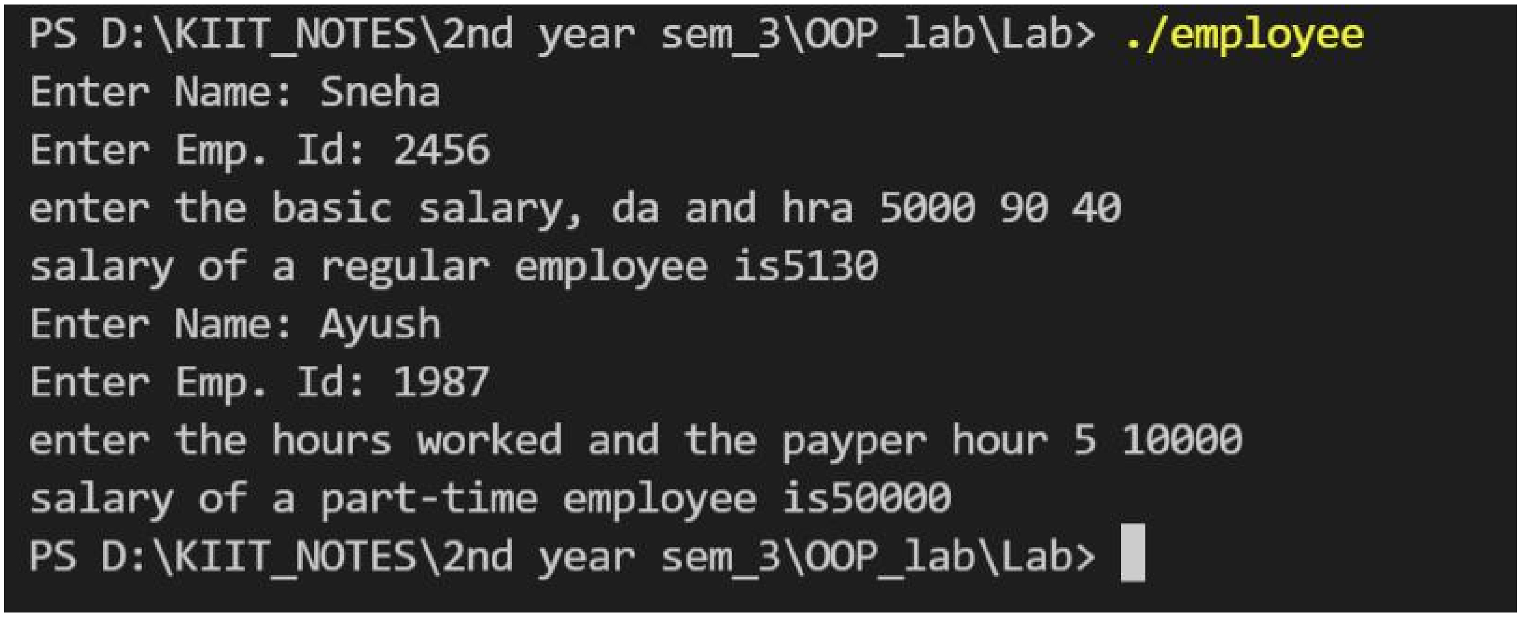


```
enter the radius of circle :
2
Circle : 12.56
enter the length and breadth of the triangle :
3 6
Triangle : 9
enter the length and breadth of rectangle :
1 5
Rectangle : 5
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

Question2)Create a class which stores employee name,id and salary Derive two classes from 'Employee' class: 'Regular' and 'Part-Time'. The 'Regular' class stores DA, HRA and basic salary. The 'Part-Time' class stores the number of hours and pay per hour.
Calculate the salary of a regular employee and a par-time employee.

```
#include <iostream>
using namespace std;
class Employee
{
protected:
   string name;
   int empid;
   int salary;

public:
   void input()
   {
      cout << "Enter Name: ";
      cin >> name;
      cout << "Enter Emp. Id: ";
      cin >> empid;
   }
};
class Regular : public Employee
{
protected:
   int da, hra, bsal;

public:
   void reg()
   {
      cout << "enter the basic salary, da and hra";
      cin >> bsal >> da >> hra;
   }
   void cal()
   {
      salary = bsal + da + hra;
      cout << "salary of a regular employee is" << salary << endl;
   }
};
class parttime : public Employee
{
protected:
   int hrs, payperhr;

public:
   void part()
   {
      cout << "enter the hours worked and the payper hour";
      cin >> hrs >> payperhr;
   }
```

```
    void cal()
    {
        salary = hrs * payperhr;
        cout << "salary of a part-time employee is" << salary << endl;
    }
};
int main()
{
    Regular r;
    parttime p;
    r.input();
    r.reg();
    r.cal();
    p.input();
    p.part();
    p.cal();
}
```

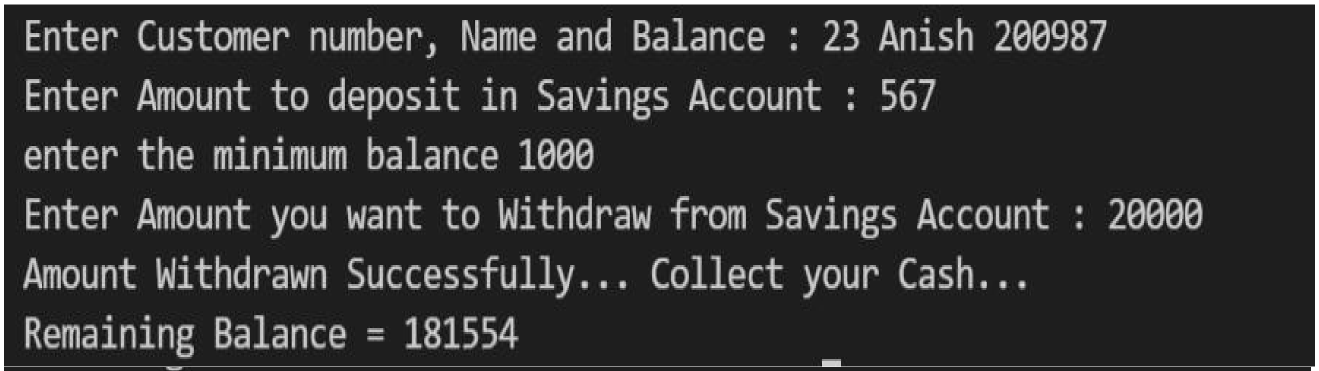


```
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab> ./employee
Enter Name: Sneha
Enter Emp. Id: 2456
enter the basic salary, da and hra 5000 90 40
salary of a regular employee is5130
Enter Name: Ayush
Enter Emp. Id: 1987
enter the hours worked and the payper hour 5 10000
salary of a part-time employee is50000
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

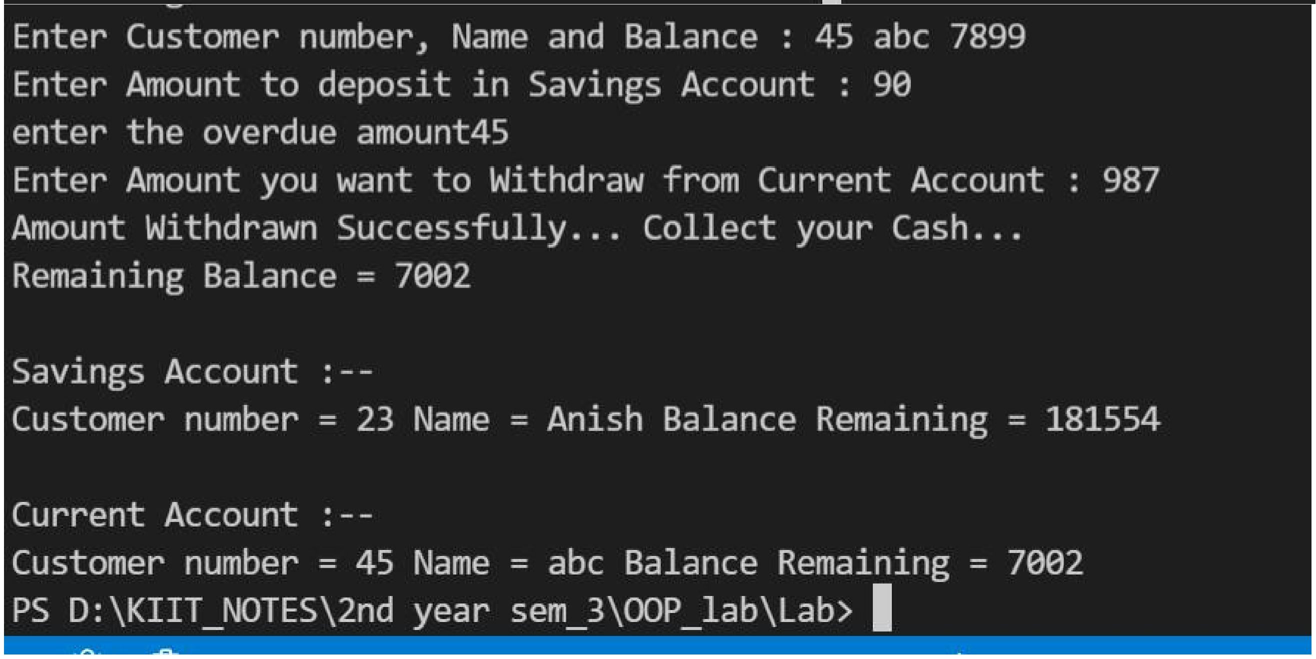
Question3)Create a class which stores account number, customer name and balance. Derive two classes from 'Account' class: 'Savings' and 'Current'. The 'Savings' class stores minimum balance. The 'Current' class stores the over-due amount. Include member functions in the appropriate class for
-deposit money
-withdraw [For saving account minimum balance should be checked.]
[For current account overdue amount should be calculated.]
-display balance

```
#include <iostream>
using namespace std;
class Account
{
protected:
   int custno;
   string custname;
   int balance;

public:
   void input()
   {
      cout << "Enter Customer number, Name and Balance : ";
      cin >> custno >> custname >> balance;
   }
};
class Savings : public Account
{
protected:
   int minbalance;

public:
   void depositSavings()
   {
      int dep;
      cout << "Enter Amount to deposit in Savings Account : ";
      cin >> dep;
      balance += dep;
   }
   void withdrawSavings()
   {
      cout << "enter the minimum balance";
      cin >> minbalance;
      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 balance ... " << endl;
      else
      {
        balance -= with;
        cout << "Amount Withdrawn Successfully... Collect your Cash... \nRemaining Balance = " <<
balance << endl;
      }
   }
   void Display()
   {
      cout << "\nSavings Account :--\nCustomer number = " << custno << " Name = " << custname << "
Balance Remaining = " << balance << endl;
   }
};
class Current : public Account
{
protected:
   int overdue;

public:
   void depositCurrent()
   {
      int dep;
      cout << "Enter Amount to deposit in Savings Account : ";
      cin >> dep;
      balance += dep;
   }
   void withdraw()
   {
      cout << "enter the overdue amount";
      cin >> overdue;
      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 amount ... " << endl;
      else
      {
        balance -= with;
        cout << "Amount Withdrawn Successfully... Collect your Cash... \nRemaining Balance = " <<
balance << endl;
      }
   }
   void Display()
   {
      cout << "\nCurrent Account :--\nCustomer number = " << custno << " Name = " << custname << "
Balance Remaining = " << balance << endl;
   }
};
int main()
{
   Savings obs;
   Current obc;
   obs.input();
   obs.depositSavings();
   obs.withdrawSavings();
   obc.input();
```

```
    obc.depositCurrent();
    obc.withdraw();
    obs.Display();
    obc.Display();
}
```



```
Enter Customer number, Name and Balance : 23 Anish 200987
Enter Amount to deposit in Savings Account : 567
enter the minimum balance 1000
Enter Amount you want to Withdraw from Savings Account : 20000
Amount Withdrawn Successfully... Collect your Cash...
Remaining Balance = 181554
```

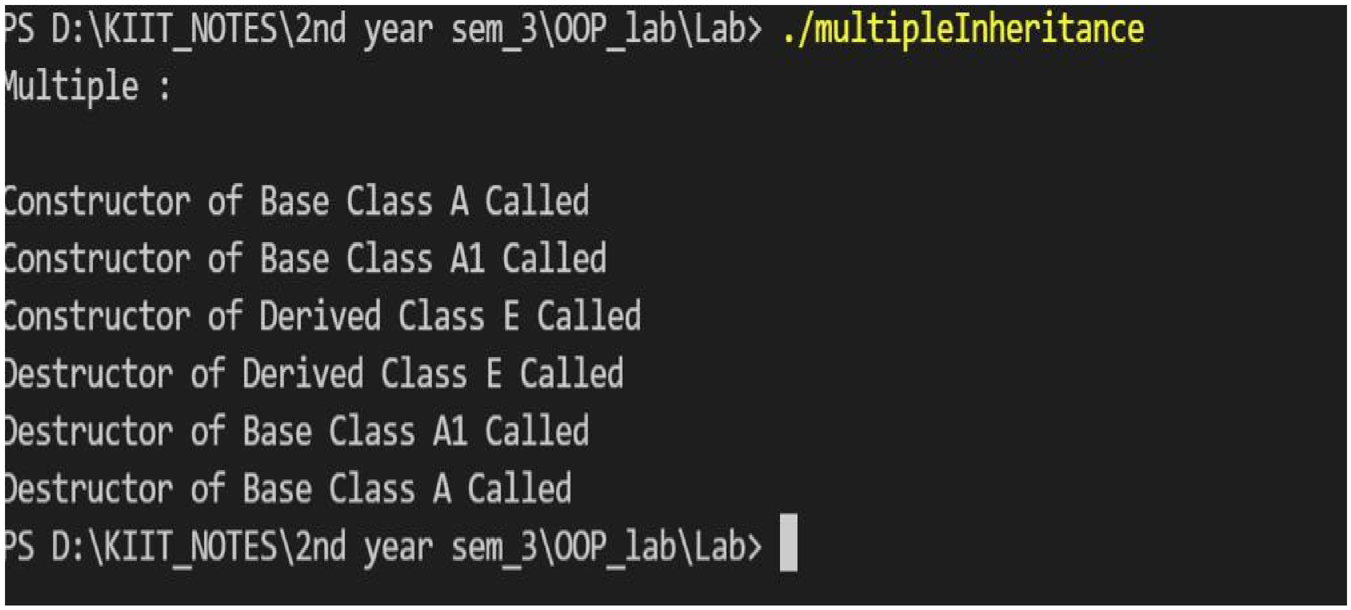


```
Enter Customer number, Name and Balance : 45 abc 7899
Enter Amount to deposit in Savings Account : 90
enter the overdue amount45
Enter Amount you want to Withdraw from Current Account : 987
Amount Withdrawn Successfully... Collect your Cash...
Remaining Balance = 7002

Savings Account :--
Customer number = 23 Name = Anish Balance Remaining = 181554

Current Account :--
Customer number = 45 Name = abc Balance Remaining = 7002
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

Question4)WAP to demonstrate the order of call of constructors and destructors in case of multiple inheritance.

```
#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 : \n"
```

```
        << endl;
    E obe;
    return 0;
}
```

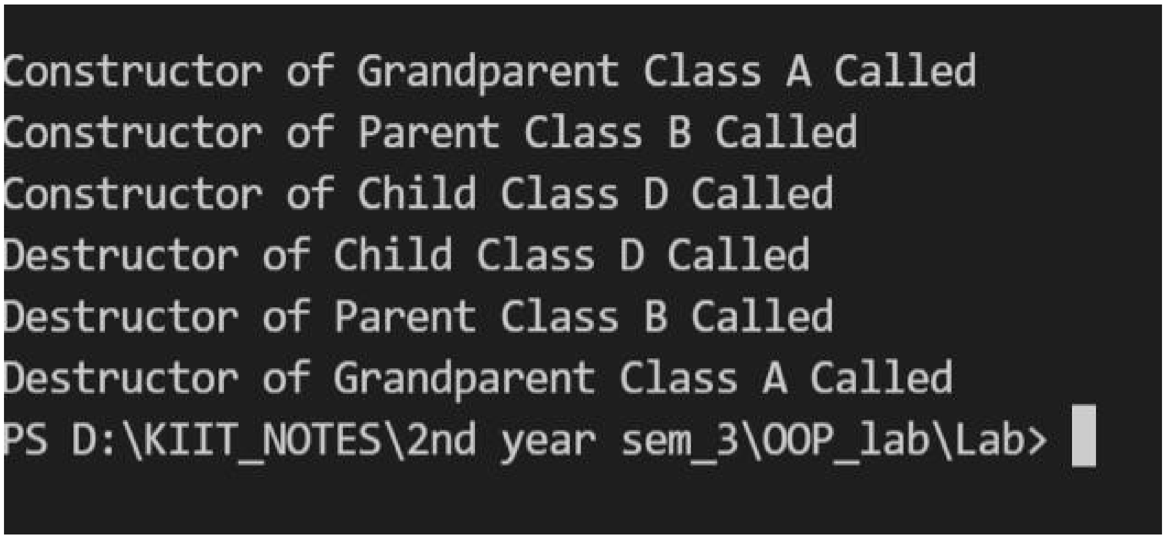


```
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab> ./multipleInheritance
Multiple :

Constructor of Base Class A Called
Constructor of Base Class A1 Called
Constructor of Derived Class E Called
Destructor of Derived Class E Called
Destructor of Base Class A1 Called
Destructor of Base Class A Called
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

Question5)WAP to demonstrate the order of call of constructors and destructors in case of multilevel inheritance.

```
#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 obd;
}
```

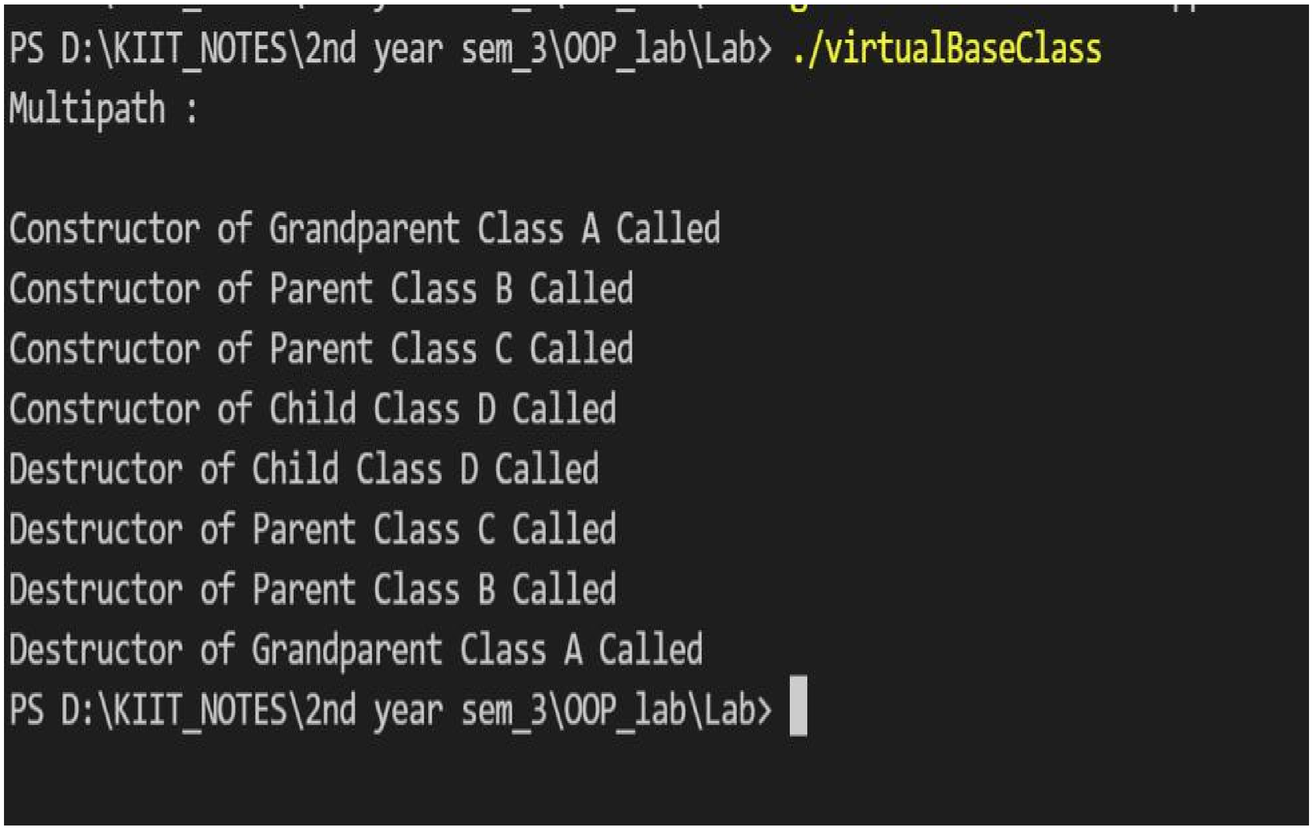


```
Constructor of Grandparent Class A Called
Constructor of Parent Class B Called
Constructor of Child Class D Called
Destructor of Child Class D Called
Destructor of Parent Class B Called
Destructor of Grandparent Class A Called
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

Question6)WAP to demonstrate the order of call of constructors and destructors in case of virtual base class .

```
#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;
}
```

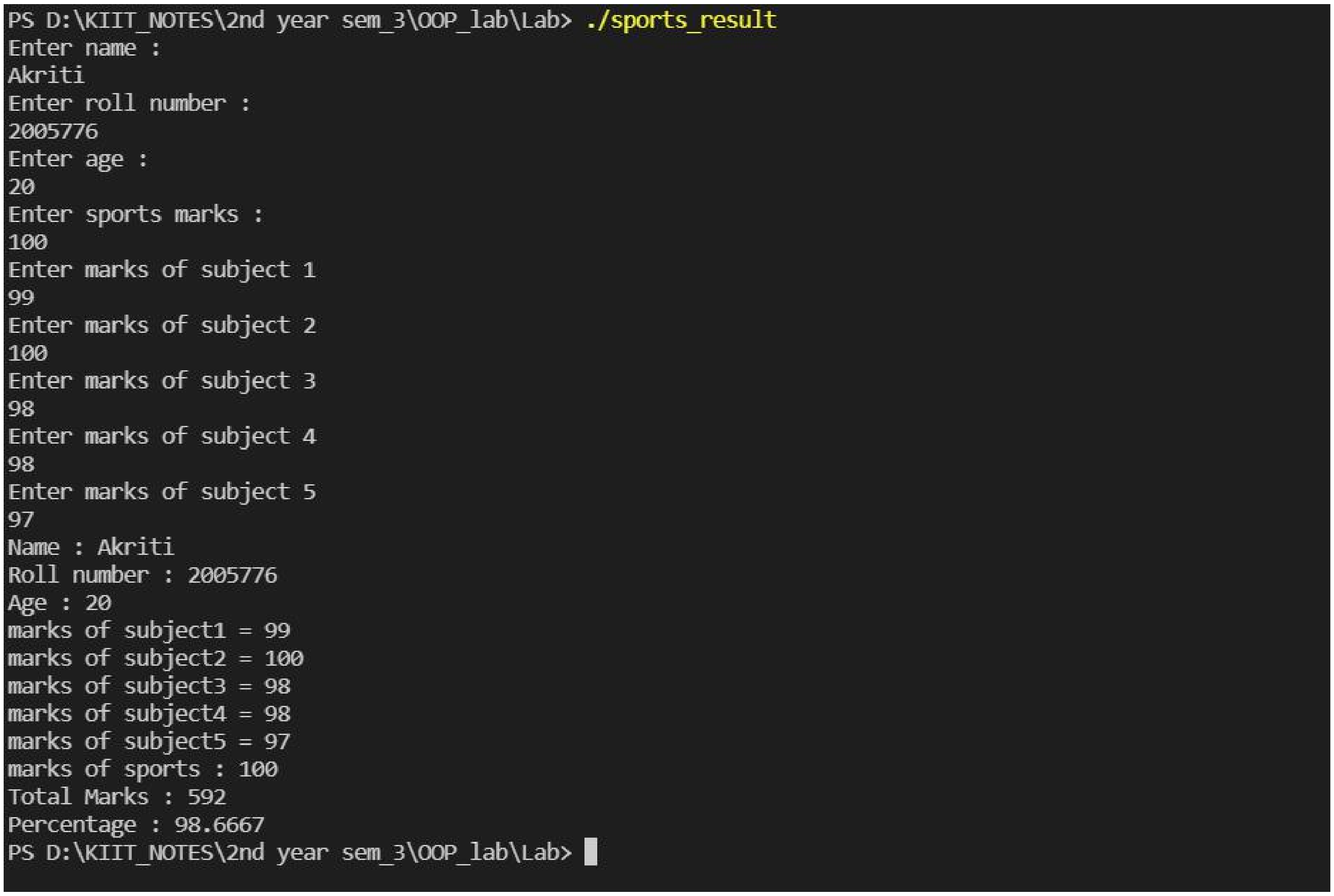


```
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab> ./virtualBaseClass
Multipath :

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
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

Question7)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.

```
#include <iostream>
#include <numeric>

using namespace std;

class student
{
private:
   string name;
   unsigned int roll_num;
   unsigned int age;

public:
   student()
   {
      name = " ";
      roll_num = 0;
      age = 0;
   }
   student(string n, int r, int a)
   {
      name = n;
      roll_num = r;
      age = a;
   }
   void displaySt()
   {
      cout << "Name : " << name << "\n";
      cout << "Roll number : " << roll_num << "\n";
      cout << "Age : " << age << "\n";
   }
};

class Test
{
public:
   int marks[5];

   Test()
   {
   }
   void input( )
   {
      for (int i = 0; i < 5; ++i)
      {
         cout << "Enter marks of subject " << i + 1 << "\n";
         cin >> marks[i];
```

```
        }
    }
    void displayTest()
    {
        cout << "marks of subject1 = " << marks[0] << "\n";
        cout << "marks of subject2 = " << marks[1] << "\n";
        cout << "marks of subject3 = " << marks[2] << "\n";
        cout << "marks of subject4 = " << marks[3] << "\n";
        cout << "marks of subject5 = " << marks[4] << "\n";
    }
};

class Sports
{
public:
    int sportsMarks;

    Sports()
    {
        sportsMarks = 0;
    }
    Sports(int s)
    {
        sportsMarks = s;
    }
    void displaySports()
    {
        cout << "marks of sports : " << sportsMarks << "\n";
    }
};

class Result : public student, public Test, public Sports
{
public:
    unsigned int total;
    float percent;
    Result() : student(), Sports(), Test()
    {
        total = 0;
        percent = 0.0;
    }
    Result(string n, int r, int a, int s)
        : student(n, r, a), Sports(s), Test()
    {
        total = 0;
        percent = 0.0;
        input();
    }
    void sum()
    {
       total = accumulate(marks, marks + 5, total);

        total += sportsMarks;
        percent = total / 6.0;
    }
    void displayRes()
    {
        displaySt();
        displayTest();
        displaySports();
        cout << "Total Marks : " << total << "\n";
        cout << "Percentage : " << percent << "\n";
    }
};
int main()
{
    string n;
    int r, a, s;
    cout << "Enter name :\n";
    cin >> n;
    cout << "Enter roll number :\n";
    cin >> r;
    cout << "Enter age :\n";
    cin >> a;
    cout << "Enter sports marks :\n";
    cin >> s;
    Result obj(n, r, a, s);
    obj.sum();
    obj.displayRes();
    return 0;
}
```

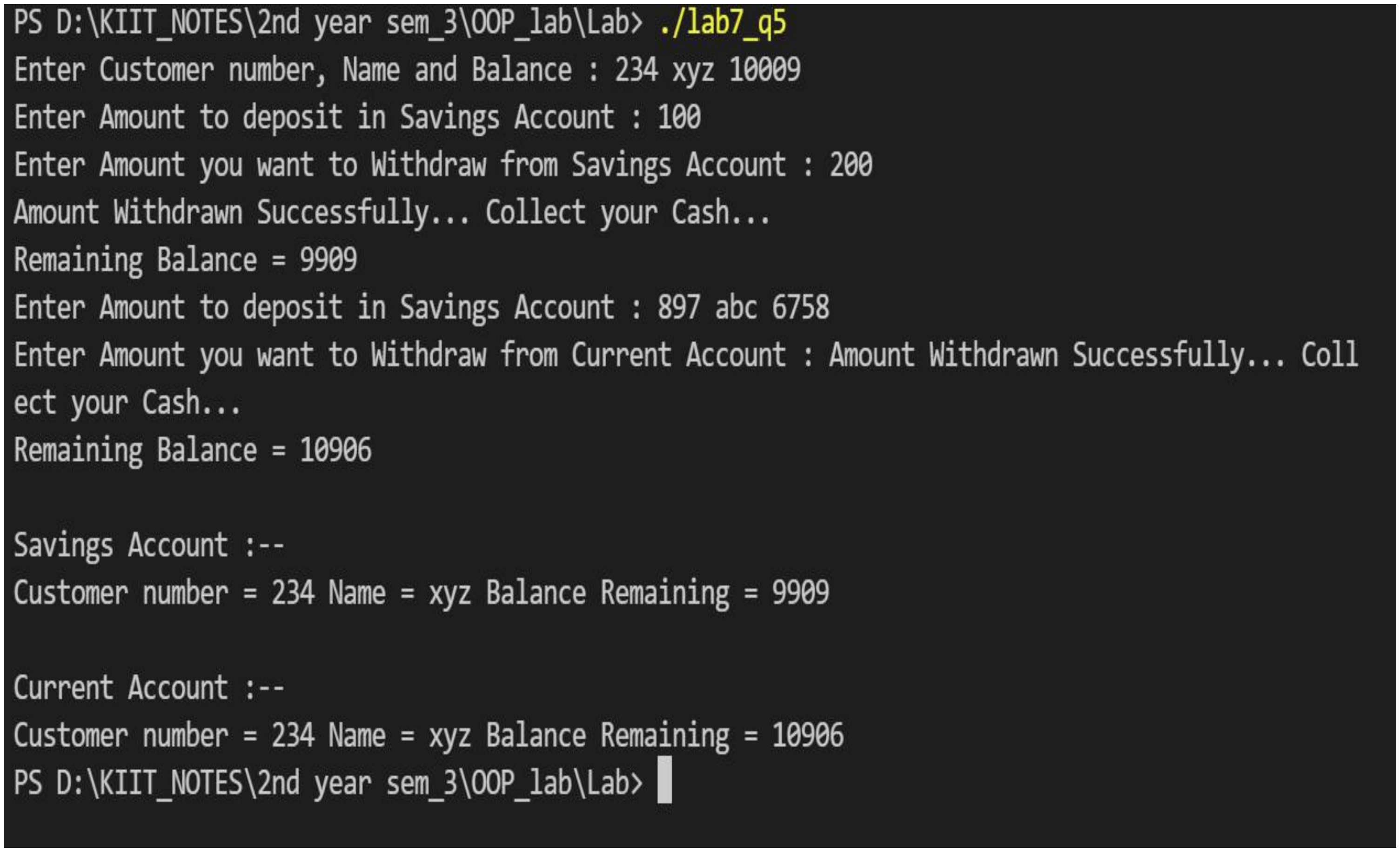


```
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab> ./sports_result
Enter name :
Akriti
Enter roll number :
2005776
Enter age :
20
Enter sports marks :
100
Enter marks of subject 1
99
Enter marks of subject 2
100
Enter marks of subject 3
98
Enter marks of subject 4
98
Enter marks of subject 5
97
Name : Akriti
Roll number : 2005776
Age : 20
marks of subject1 = 99
marks of subject2 = 100
marks of subject3 = 98
marks of subject4 = 98
marks of subject5 = 97
marks of sports : 100
Total Marks : 592
Percentage : 98.6667
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```

Question8)Rewrite the assignment vii. From Inheritance including the parameterized constructors in all the classes.

```
#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 balance ... " << endl;
      else
      {
         balance -= with;
         cout << "Amount Withdrawn Successfully... Collect your Cash... \nRemaining Balance = " <<
balance << endl;
      }
   }
   void Display()
```

```
    {
        cout << "\nSavings Account :--\nCustomer number = " << custno << " Name = " << 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)
    {
        overdue = ovdue;
    }
    void depositCurrent()
    {
        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 Current Account : ";
        cin >> with;
        if (balance - with < overdue)
            cout << "Amount can't be withdrawn as you will not be left with Over-due amount ... " << endl;
        else
        {
            balance -= with;
            cout << "Amount Withdrawn Successfully... Collect your Cash... \nRemaining Balance = " << 
balance << endl;
        }
    }
    void Display()
    {
        cout << "\nCurrent Account :--\nCustomer number = " << custno << " Name = " << 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(100, cno, nam, bal);
    Current obc(100, cno, nam, bal);
    obs.depositSavings();
    obs.withdraw();
    obc.depositCurrent();
    obc.withdraw();
    obs.Display();
    obc.Display();
}
```



```
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab> ./lab7_q5
Enter Customer number, Name and Balance : 234 xyz 10009
Enter Amount to deposit in Savings Account : 100
Enter Amount you want to Withdraw from Savings Account : 200
Amount Withdrawn Successfully... Collect your Cash...
Remaining Balance = 9909
Enter Amount to deposit in Savings Account : 897 abc 6758
Enter Amount you want to Withdraw from Current Account : Amount Withdrawn Successfully... Coll
ect your Cash...
Remaining Balance = 10906

Savings Account :--
Customer number = 234 Name = xyz Balance Remaining = 9909

Current Account :--
Customer number = 234 Name = xyz Balance Remaining = 10906
PS D:\KIIT_NOTES\2nd year sem_3\OOP_lab\Lab>
```