Alliance University

**Department of Computer Science & Engineering Design**

# Lab Manual

**Course Code: CSL308**
**Course Title: OBJECT ORIENTED PROGRAMMING WITH C++ LAB**
**Semester : 3rd**

**(common to all the sections)**

**Academic Year:- 2022-23**

**LIST OF EXPERIMENTS**

1. a) Create a Structure STUDENT with the fields name, roll_num, marks (marks in 5 different subjects), total and average and write the functions to perform following operations:

1. Input the details of n students.
2. Calculate the total and average marks for each student.
3. Sort the details of students based on roll number.

2. Create a structure EMPLOYEE with the field's employee name, id, grade, basic, da, hra, gross salary, net salary. Input the details of n employees and calculate the gross salary, net salary and tax paid for each of them. Write the functions to perform the following operations:

1. Sort the employee details based on id.
2. Display the details of all the employees of a given grade and sort based on employee id.
3. Display the details of employee who pays Highest Tax.

Display the details of employee who pays Lowest Tax.

3 a. Write program to implement function handling in C++ to add two numbers sing with arguments and with return values.

b) Write a C++ program to compute area of Square, Circle, Triangle and Rectangle using function overloading concept.

4.a. write a program in C++ to prepare a student Record using class and object.
b. write a program to Implement of constructors

5 a) Write a C++ program to implement single inheritance using employee 1 as base class and employee 2 as derived class. Calculate and display the gross salary for both the employee class object, considering sales percentage for employee 2 class object.

b) Write a C++ program to create a class called student with data members username, id and age using inheritance. Create the class ug_students and pg_students having data members called semester, fee_stipend, CGPA. Enter data for at least 10 students. Find the semester wise average age for all ug and pg students.

6) a) Write a C++ program to demonstrate hierarchical inheritance for Employee class as base class (name and e_id as data member) and derived classes as Manager Class (title and dues as data member), Scientist class(pubs as data member) and Laborer class(no data member). Read and display the data for multiple derived class objects.

b) Write a C++ program to add three Distance class objects.

7 ) Write a C++ program to demonstrate multiple inheritance

8. Write a C++ program to compute the area of triangle and rectangle using pure virtual function..

9) write program to swap the numbers using the concept of function template

10. a Write a C++ program to Overload Binary + operator to add two Distance class objects.

b) Write a C++ program to set duration and distance class objects by passing them as parameter to friend function

11. write a program to handle Division by zero Exception in C++

12) Write a C++ program to create a Bank Application.

**1. a)** Create a **Structure** STUDENT with the fields name, roll_num, marks (marks in 5 different subjects), total and average and write the functions to perform following operations:

1. Input the details of n students.
2. Calculate the total and average marks for each student.
3. Sort the details of students based on roll number.

**program**

```
//program to student record
#include<iostream>
using namespace std;
//declaring functions
int i,j,n;
void input();
void calculate();
void sort();
void display();
// structure declartion
struct student
{
int rollno;
char name[10];
int m[10],total;
float avg=0;
} s[3],temp;
//functions definition
void input()
{
for(i=1;i<=n;i++)
{
cout<<"enter rno,name";
cin>>s[i].rollno>>s[i].name;
cout<<"enter marks 5";
for(j=1;j<=5;j++)
{
cin>>s[i].m[j];
}
}
}
void calculate()
{
   int t=0;
for(i=1;i<=n;i++)
{
for(j=1;j<=5;j++)
{
```

```
t=t+s[i].m[j];
}
s[i].total= t;
s[i].avg=s[i].total/5;
}
}
void sort()
{
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
{
if(s[i].rollno>s[j].rollno)
{
temp=s[i];
s[i]=s[j];
s[j]=temp;;
}
}
}
}
void display()
{
cout<<"student Information"<<endl;
for(i=1;i<=n;i++)
{
cout<<"Rno=";
cout<<s[i].rollno;
cout<<endl<<"Name=";
cout<<s[i].name<<endl;
for(j=1;j<=5;j++)
{
cout<<"Mark "<<j<<"=";
cout<<s[i].m[j]<<endl;
}
cout<<endl<<"total=";
cout<<s[i].total;
cout<<endl<<"Average=";
cout<<s[i].avg<<endl;
}
}
int main()
{
cout<<"enter no of students records";
cin>>n;
input();
calculate();
sort();
display();
return 0;
}
```

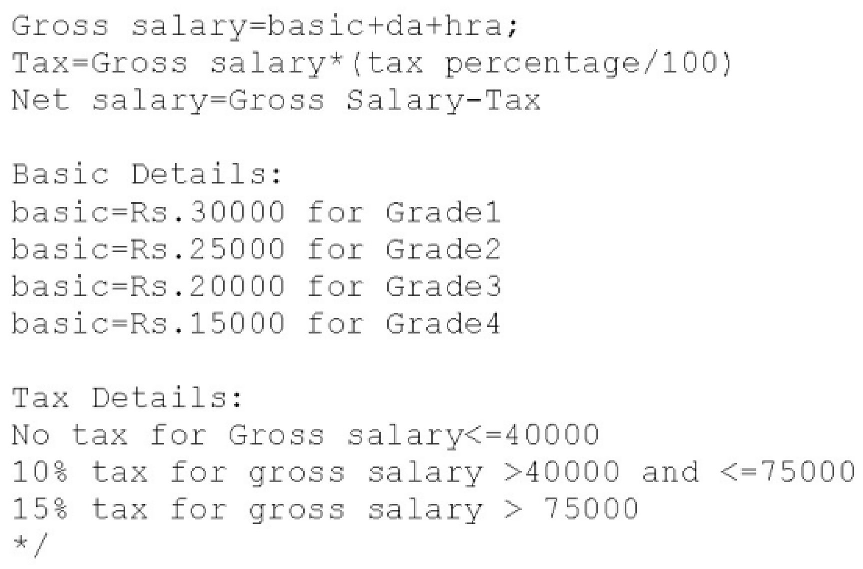
**Output**

```
enter no of students records
2
enter rno,name12
123
kavin
enter marks 5
56
89
78
67
90
enter rno,name
122
karthik
enter marks 5
89
67
56
90
100

student Information

Rno=122
Name=karthik
Mark 1=89
Mark 2=67
Mark 3=56
Mark 4=90
Mark 5=100

total=782
Average=156
Rno=123
Name=kavin
Mark 1=56
Mark 2=89
Mark 3=78
Mark 4=67
Mark 5=90

total=380
Average=76
```

**2.** Create a structure EMPLOYEE with the field's employee name, id, grade, basic, da, hra, gross salary, net salary. Input the details of n employees and calculate the gross salary, net salary and tax paid for each of them. Write the functions to perform the following operations:

1. Sort the employee details based on id.
2. Display the details of all the employees of a given grade and sort based on employee id.
3. Display the details of employee who pays Highest Tax.

Display the details of employee who pays Lowest Tax.



```
Gross salary=basic+da+hra;
Tax=Gross salary*(tax percentage/100)
Net salary=Gross Salary-Tax

Basic Details:
basic=Rs.30000 for Grade1
basic=Rs.25000 for Grade2
basic=Rs.20000 for Grade3
basic=Rs.15000 for Grade4

Tax Details:
No tax for Gross salary<=40000
10% tax for gross salary >40000 and <=75000
15% tax for gross salary > 75000
*/
```

**Note: fix HRA =25% and DA=20%**

```
//program to Employee record
#include<iostream>
using namespace std;
//declaring functions
int i,j,n;
void input();
void calculate();
void sort();
void display();
void highesttax();
void lowesttax();
// structure declartion
struct employee
{
int empid;
char empname[10];
int grade;
float da,hra,tax,basic,grosssal,netsal;
} e[10],temp;
//functions definition
// get values from user
```

```
void input()
{
for(i=1;i<=n;i++)
{
cout<<"enter empid ,empname";
cin>>e[i].empid>>e[i].empname;
cout<<"enter grade";
cin>>e[i].grade;
}
}

void calculate()
{
for(i=1;i<=n;i++)
{
// basic
if (e[i].grade==1)
e[i].basic=30000;
else if (e[i].grade==2)
e[i].basic=25000;
else if (e[i].grade==3)
e[i].basic=20000;
else if (e[i].grade==4)
e[i].basic=15000;
else
cout<< " grade sholud within 1-4";
//hra,da

e[i].hra =(e[i].basic)*(0.25);
e[i].da= (e[i].basic)*(0.20);

//gross salary
e[i].grosssal= e[i].basic+ e[i].hra+ e[i].da;
//tax
 if (e[i].grosssal <=40000)
 e[i].tax=0;
if ((e[i].grosssal >40000) && (e[i].grosssal <=75000))
 e[i].tax= e[i].grosssal *(0.10);
if ((e[i].grosssal >75000))
 e[i].tax= e[i].grosssal *(0.15);
//netsalary
cout<<e[i].tax;
e[i].netsal= e[i].grosssal- e[i].tax;
}
}
// 1.Sort the employee details based on emp id
void sort()
{
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
```

```
{
if(e[i].empid>e[j].empid)
{
temp=e[i];
e[i]=e[j];
e[j]=temp;
}
}
}
}

// 2. Display the details of all employees after sort
void display()
{
cout<<"Employee Information"<<endl;
for(i=1;i<=n;i++)
{
cout<<"Employee name=";
cout<<e[i].empname;
cout<<endl<<"Empid=";
cout<<e[i].empid;
cout<<endl<<"Grade=";
cout<<e[i].grade;
cout<<endl<<"Basic salary=";
cout<<e[i].basic;
cout<<endl<<"Gross salary=";
cout<<e[i].grosssal;
cout<<endl<<"Net salary=";
cout<<e[i].netsal;
}
}
//3a. Display the employee details paying highest tax.
void highesttax()
{
cout<<"Employee Information of highest tax payer"<<endl;
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
{
if(e[i].tax<e[j].tax)
{
temp=e[i];
e[i]=e[j];
e[j]=temp;
}
cout<<"Employee name=";
cout<<e[i].empname;
cout<<endl<<"Empid=";
cout<<e[i].empid;
cout<<endl<<"Grade=";
cout<<e[i].grade;
cout<<endl<<"tax value=";
cout<<e[i].tax;
}
}
}

//3b. Display the employee details paying lowest tax.
void lowesttax()
{
cout<<"Employee Information of  low tax payer"<<endl;
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
{
if(e[i].tax>e[j].tax)
{
temp=e[i];
e[i]=e[j];
e[j]=temp;
}
cout<<"Employee name=";
cout<<e[i].empname;
cout<<endl<<"Empid=";
cout<<e[i].empid;
cout<<endl<<"Grade=";
cout<<e[i].grade;
cout<<endl<<"tax value=";
cout<<e[i].tax;
}
}
}
int  main()
{
  cout<< "enter number of records"<<endl;
 cin>>n;
 input();
 calculate();
 sort();
 display();
 highesttax();
lowesttax();
return 0;
}
```

**Ouput**

```
enter number of records
2
enter empid ,empname
111
birala
```

```
enter grade
1
enter empid ,empname
100
elonmusk
enter grade
2
Employee Information
Employee name=elonmusk
Empid=100
Grade=2
Basic salary=25000
Gross salary=36250
Net salary=36250

Employee name=birala
Empid=111
Grade=1
Basic salary=30000
Gross salary=43500
Net salary=39150

Employee Information of highest tax payer
Employee name=birala
Empid=111
Grade=1
tax value=4350

Employee Information of  low tax payer
Employee name=elonmusk
Empid=100
Grade=2
tax value=0
```

**3 a. Write program to implement function handling in C++ to add two numbers using with arguments and with return values.**

Solution:

```
#include <iostream>

using namespace std;

//function definition
int addition(int a,int b)
{
        return (a+b);
}

int main()
```

```
{
            int num1, num2;


            //read numbers
            cout<<"Enter first number: ";
            cin>>num1;
            cout<<"Enter second number: ";
            cin>>num2;


            //print addition
            cout<<"Addition is: "<<addition(num1,num2)<<endl;

            return 0;
}
```

**b) Write a C++ program to compute area of Square, Circle, Triangle and Rectangle using function overloading concept.**

Solution:

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

int area(int s)
{
    return(s*s);
}
float area(float r)
{
    return(3.14*r*r);
}

float area(float bs,float ht)
{
   return((bs*ht)/2);
}


int area(int l,int b)
{
    return(l*b);
}


int main()
{
        int s,l,b;
        float r,bs,ht;
        cout<<"Enter side of a square:";
        cin>>s;
        cout<<"Enter radius of circle:";
        cin>>r;
        cout<<"Enter base and height of triangle:";
        cin>>bs>>ht;
        cout<<"Enter length and breadth of rectangle:";
        cin>>l>>b;

        cout<<"Area of square is"<<area(s);
        cout<<"\nArea of circle is "<<area(r);
        cout<<"\nArea of triangle is "<<area(bs,ht);
        cout<<"\nArea of rectangle is "<<area(l,b);
}
```

**4**.a. write a program in C++ to prepare a student Record using class and object.

b. write a program to Implement of **constructors**

**//a. program to create student record using class and object**

```
             class student
            {
               int rno;
              char name[10];
            public:
              void getdata()

            {

            Cout<<"enter rno,name";
            cin>> name>>rno;
            }
            Void putdata
            {
             cout<<"Name="
            cout<< name
            cout <<"Rno="
            cout<<rno;
           };
   void main()
{
student s1;
  s1.getdata();
  s1.putdata();
}
```

**//b. C++ program to calculate the area of a rectangle using constructor**

```
// parametrized constuctor
include<iostream>
using namespace std;
class AreaofRect
{
    private:
        int length;
        int breadth;
    public:

         AreaofRect (int l,int b)
        { int c;
             length=l;
             breadth=b;
             c=length * breadth;
             cout<<"area of rectanlge="<<c<<endl;
        }
        int display()
        {
            cout<<"length="<<length<<endl;
             cout<<"breadth="<<breadth;
        }
};

int main ()
{
    AreaofRect  R(2,2);
    R.display();

    return 0;
}
```

**<u>Output</u>**

```
area of rectanlge=4
length=2
breadth=2
```

**5 a)** Write a C++ program to implement **single inheritance** using employee 1 as base class and employee 2 as derived class. Calculate and display the gross salary for both the employee class object, considering sales percentage for employee 2 class object.

```
 #include <iostream>
  using namespace std;
 class employee1    //single base class
 {
     public:
     int eno;
```

```
    char name[20], des[20];
    float bp, hra, da, pf, np;

    void getdata() {
        cout << "Enter the employee name:";
        cin>>name;
        cout << "Enter the designation:";
        cin>>des;
        cout << "Enter the basic pay:";
        cin>>bp;
        cout << "Enter the Humen Resource Allowance:";
        cin>>hra;
        cout << "Enter the Dearness Allowance :";
        cin>>da;
        cout << "Enter the Profitablity Fund:";
        cin>>pf;
    }

   void calculate()
    {
        np = bp + hra + da - pf;
    }

    };

class employee2 : public employee1    //single derived class
{
   private:
   float y;
   public:
   void data()
   {
     cout << "Enter the salse percentage= "; cin >> y;
   }
   void display()
   {
       cout << "\t Name " << name<< "\n";
        cout<< "\t Designation " << "\t" << des<< "\n";
        cout<<"\t Basic Pay " << bp<< "\n";
        cout<< "\t HRA " << hra<< "\n";
        cout<< "\t DA " << da<< "\n";
        cout<< "\t PF " << pf<< "\n";
        cout<< "\t Gross Pay " << np << "\n"<< "\n";
        cout << "Net Pay = " << (np+((np * y)/100));
   }
};
```

```
 int main()
 {
    employee2  a;     //object of derived class
    a.getdata();
    a.calculate();
    a.data();
    a.display();
      return 0;
 }
```

<u>Output</u>

```
Enter the employee name:
elon musk
Enter the designation:Enter the basic pay:
35000
Enter the Humen Resource Allowance:
3000
Enter the Dearness Allowance :
1000
Enter the Profitablity Fund:
800
Enter the salse percentage=
10
Name elon
        Designation    musk
        Basic Pay 35000
        HRA 3000
        DA 1000
        PF 800
        Gross Pay 38200

Net Pay = 42020
```

5.b) Write a C++ program to create a class called student with data members username, id and age using inheritance. Create the class ug_students and pg_students having data members called semester, fee_stipend, CGPA. Enter data for at least 10 students. Find the semester wise average age for all ug and pg students.

**//Multiple inheritance**

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

class student
{
    public:
          int reg, age;
          char name[20];
          void read_data();
};

class ugstudent: public student
{
   public:  int stipend,sem;
          float fees;
         void read_data();
};


class pgstudent:public student
{
   public:  int stipend,sem;
          float fees;
         void read_data();
};


void student::read_data()
{
    cout<<"\n Enter name:";
    cin>>name;
    cout<<"\n Enter Reg.no.";
    cin>>reg;
    cout<<"\n Enter age:";
    cin>>age;
 }
void ugstudent::read_data()
{
    student::read_data();
    cout<<"\nEnter the sem:";
    cin>>sem;
    cout<<"\nEnter the fees:";
    cin>>fees;
    cout<<"\nEnter the stipend:";
    cin>>stipend;
}
```

```
/* function to read additional details for pgstudents*/
void pgstudent::read_data()
{
    student::read_data();
    cout<<"\nEnter the sem:";
    cin>>sem;
    cout<<"\nEnter the fees:";
    cin>>fees;
    cout<<"\nEnter the stipend:";
    cin>>stipend;
}

/* main function */
int main()
{
    ugstudent ug[20];
    pgstudent pg[20];
    int i,n,m;
    float average;
    cout<<"\nEnter the no. of entries in the ugstudent class:";
    cin>>n;
    for(i=1;i<=n;i++)
      ug[i].read_data();
    for(int sem=1;sem<=8;sem++)
    {
       float sum=0;
       int found=0,count=0;
      for(i=1;i<=n;i++)
      {
          if(ug[i].sem==sem)
         {
            sum=sum+ug[i].age;
            found=1;
            count++;
         }
      }
       if(found==1)
       {
         average=sum/count;
         cout<<"\nAverage of age of sem "<<sem<<" is "<<average;

       }
    }
       cout<<"\nEnter the no. of entries of pgstudent class:";
       cin>>n;
```

```
        for(i=1;i<=n;i++)
        pg[i].read_data();
        for(int sem=1;sem<=8;sem++)
        {
           float sum=0;
           int found=0,count=0;
           for(i=1;i<=n;i++)
           {
              if(pg[i].sem==sem)
              {
              sum=sum+pg[i].age;
              found=1;
              count++;
              }
           }
          if(found==1)
          {
             average=sum/count;
             cout<<"\nAverage of age of sem "<<sem<<" is "<<average;
          }
        }
    return 0;
}
```

Output

```
Enter the no. of entries in the ugstudent class:2
Enter name:kkk
Enter Reg.no.1234
Enter age:19
Enter the sem:2
Enter the fees:200000
Enter the stipend:5000
Enter name:www
Enter Reg.no.1235
Enter age:20
Enter the sem:2
Enter the fees:300000
Enter the stipend:5000
Average of age of sem 2 is 19.5

Enter the no. of entries of pgstudent class:1
Enter name:kll
Enter Reg.no.129
Enter age:21
Enter the sem:3
Enter the fees:35000
Enter the stipend:1000
Average of age of sem 3 is 21
```

**6) a) Write a C++ program to demonstrate hierarchical inheritance for Employee class as base class (name and e_id as data member) and derived classes as Manager Class (title and dues as data member), Scientist class(pubs as data member) and Laborer class(no data member). Read and display the data for multiple derived class objects.**

```
// hierarchical inheritance
#include <iostream>
using namespace std;
const int LEN = 80; //maximum length of names
class employee //employee class
{
   private:
   char name[LEN]; //employee name
   unsigned long number; //employee number
   public:
   void getdata()
   {
   cout << "\n Enter last name:"; cin >> name;
   cout << "Enter number: "; cin >> number;
   }
   void putdata() const
   {
      cout << "\n Name: "<< name;
      cout << "\n Number: " << number;
   }
};

class manager : public employee //management class
{
   private:
   char title[LEN]; //"vice-president" etc.
   double dues; //golf club dues
   public:
   void getdata()
   {
      employee::getdata();
      cout << "Enter title: "; cin >> title;
      cout << "Enter dues:"; cin >> dues;
   }
   void putdata() const
   {
      employee::putdata();
```

```
        cout << "\n Title: "<< title;
        cout << "\n Dues Details: " << dues;
    }
};

class scientist : public employee //scientist class
{
    private:
    int pubs; //number of publications
    public:
    void getdata()
    {
        employee::getdata();
        cout << " Enter number of publications: "; cin >> pubs;
    }
    void putdata() const
    {
        employee::putdata();
        cout << "\n Number of publications: "<< pubs;
    }
};

class laborer : public employee //laborer class
{
};

int main()
{
            manager m1, m2;
            scientist s1;
            laborer l1;
            cout << endl; //get data for several employees
            cout << "\n Enter data for manager 1";
            m1.getdata();
            cout << "\n Enter data for manager 2";
            m2.getdata();
            cout << "\nEnter data for scientist 1";
            s1.getdata();
            cout << "\nEnter data for laborer 1";
            l1.getdata();
            //display data for several employees
            cout << "\nData on manager 1";
            m1.putdata();
            cout << "\nData on manager 2";
            m2.putdata();
            cout << "\nData on scientist 1";
            s1.putdata();
            cout << "\nData on laborer 1";
            l1.putdata();
            cout << endl;
            return 0;
}
```

```
Input
Enter data for manager 1
 Enter last name:manoj
 Enter number: 1900
 Enter title: manager
 Enter dues:1000
 Enter data for manager 2
 Enter last name:kely
 Enter number: 1200
 Enter title: ddd
 Enter dues:2000
 Enter data for scientist 1
 Enter last name:prince
 Enter number: 1222
 Enter number of publications: 1
 Enter data for laborer 1
 Enter last name:sona
 Enter number: 111
```

```
Output

Data on manager 1
 Name: manoj
 Number: 1900
 Title: manager
 Dues Details: 1000
Data on manager 2
 Name: kely
 Number: 1200
 Title: ddd
 Dues Details: 2000
Data on scientist 1
 Name: prince
 Number: 1222
 Number of publications: 1
Data on laborer 1
 Name: sona
```

```
Number: 111
```

**6 b) Write a C++ program to add three Distance class objects.**

```
//distance class:get and set the values
#include <iostream>

using namespace std;

class distances
{
    int feet;
    float inches;
    public:
    distances()
    {
        feet=0;
        inches= 0.0;
    }
    void getter();
    void setter();
    void display();
};

void distances::getter ()
{
    cout<<"enter feet and inches:";
    cin>>feet>>inches;
}

void distances::setter()
{
    while(inches>=12)
    {
    inches= inches-12;
    feet++;
    }
}


void distances::display()
{
    cout<<"feet: "<<feet<<endl;
    cout<<"inches:"<<inches<<endl;
}

int main()
{
distances d1;
d1.getter();
d1.setter();
d1.display();
return 0;
}
```

**//distance class:adding 2 objects**

```
#include <iostream>

using namespace std;

class distances
{
    int feet;
    float inches;
    public:
    distances()
    {
        feet=0;
        inches =0.0;
    }
    distances(int a,float b)
    {
        feet=a;
        inches =b;
    }

    void getter();
    distances setter(distances);
    void display();
};
void distances::getter()
{
    cout<<"enter feet and inches:";
    cin>>feet>>inches;
}
distances distances::setter(distances d2)
{
    distances temp;
    temp.inches=inches+d2.inches;
    while(temp.inches>=12)
    {
        temp.inches=temp.inches-12;
        temp.feet++;
    }
    temp.feet =feet+d2.feet+temp.feet;
    return temp;
}

void distances::display()
{
    cout<<"feet: "<<feet<<endl;
    cout<<"inches:"<<inches<<endl;
}

int main()
{
        distances d1,d2 (1,1),d3;
        d1.getter();
        d3=d1.setter (d2);
        d2.display();
        d3.display();

}
```

**//distance class-adding 3 objects**

```
#include <iostream>
using namespace std;
class distances
{
    int feet;
    float inches;
    public:
    distances()
    {
        feet=0;
        inches=0.0;
    }
distances(int a,float b)
{
    feet=a;
    inches=b;
}
void getter();
distances setter(distances,distances);
void display();
};
void distances::getter()
{
    cout<<"enter feet and inches:";
    cin>>feet>>inches;
}

distances distances::setter(distances d1,distances d2)
{
    distances temp;
    temp.inches=inches+d1.inches+d2.inches;
    while(temp.inches>=12)
    {
        temp.inches=temp.inches-12;
        temp.feet++ ;
    }
    temp.feet =feet+d1.feet+d2.feet+temp.feet;
    return temp;
}
void distances::display()
{
    cout<<"feet: "<<feet<<endl;
    cout<<"inches:"<<inches<<endl;
}

int main()
{
    distances d1,d2 (1,1),d3(2,2),d4; d1.getter();
    d4=d3.setter (d1,d2);//adding 3 objects d1.display();
    d2.display();
    d3.display();
    d4.display();
}
```

**7) Write a C++ program to demonstrate multiple inheritance**

```
#include <iostream>
using namespace std;
#include <string.h>
class person
{
        protected:
        int age;
        char name[50];
        public:
                person(int a, char *n)
                {
                        age=a;
                        strcpy(name,n);
                }
        void show()
        {
                cout<<"name: "<<name<<endl;
                cout<<"age: "<<age<<endl;
        }
};
class Employee
{
        protected:
                float salary;
        public:
                Employee(int s)
        {
                salary=s;
        }
        void show()
        {
        cout<<"salary: "<<salary<<endl;
        }
};
class Teacher: public person, Employee
{
        protected:
                char area[50];
        public:
                Teacher(int a, char *n, int s, char *ar): Employee(s), person(a, n)
                {
                        strcpy(area,ar);
                }
        void show()
        {
                person::show();
                Employee::show();
                cout<<"research_area: "<<area<<endl;
        }
};
int main()
{
        Teacher T1 (21,"ABC",7879,"Comp");
        T1.show();
        return 0;}
```

**8. Write a C++ program to compute the area of triangle and rectangle using pure virtual function**

**//program**

```
#include <iostream>
using namespace std;
class Shape
{
        public:
                virtual void area() = 0;
};
class Triangle : public Shape
{

        private:
            int base;
            int height;
        public:
           Triangle(int b,int h)
           {
            base=b;
            height=h;
           }
                void area()
                {
                        cout<<"Area of Triangle is: "<<(0.5*base*height)<<endl;
                }
};
class Rectangle : public Shape
{
        private:
                int l;
                int b;
        public:
                Rectangle(int x, int y)
                {
                        l = x;
                        b = y;
                }
                void area()
                {
                        cout<<"Area of rectangle is: "<<(l*b)<<endl;
                }
};

int main()
{
        Shape *s;
        s = new Triangle(20,30);
        s->area();
        s = new Rectangle(10, 20);
        s->area();
        return 0;
```

```
}
```

**//output**

```
Area of Triangle is: 300
Area of rectangle is: 200
```

**9 . write program to swap the numbers using the concept of function template**

**//program**

```
#include <iostream>
using namespace std;
template <class T>
int swapping(T& x, T& y)
{
    T t;
    t = x;
    x = y;
    y = t;
    return 0;
}
int main()
{
    int a, b;
    cout<<"enter a and b values\n";
    cin>>a>>b;
    cout<<"Before swapping"<<" "<<a<<" "<<b<<endl;
    swapping(a, b);
    cout <<"After Swapping"<<" "<< a << " " << b << endl;
    return 0;
}
```

**//output**

```
enter a and b values
250
350
Before swapping 250 350
After Swapping 350 250
```

**10. a Write a C++ program to Overload Binary + operator to add two Distance class objects.**

Solution:

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

class Distance {
private:
    int feet, inches;

public:
    // function to read distance
    void readDistance(void)
    {
        cout << "Enter feet: ";
        cin >> feet;
        cout << "Enter inches: ";
        cin >> inches;
    }

    // function to display distance
    void dispDistance(void)
    {
        cout << "Feet:" << feet << "\t"
             << "Inches:" << inches << endl;
    }

    // add two Distance using + operator overloading
    Distance operator+(Distance& dist1)
    {
        Distance tempD; // to add two distances
        tempD.inches = inches + dist1.inches;
        tempD.feet = feet + dist1.feet + (tempD.inches / 12);
        tempD.inches = tempD.inches % 12;
        return tempD;
    }
};

int main()
{
    Distance D1, D2, D3;

    cout << "Enter first  distance:" << endl;
    D1.readDistance();
    cout << endl;

    cout << "Enter second distance:" << endl;
    D2.readDistance();
    cout << endl;

    // add two distances
    D3 = D1 + D2;

    cout << "Total Distance:" << endl;
    D3.dispDistance();

    cout << endl;

    return 0;
}
```

**10. b) Write a C++ program to set duration and distance class objects by passing them as parameter to friend function**

Solution:

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

class Distance {
    private:
        int meter;

        friend int addFive(Distance);

    public:
        Distance() : meter(0) {}

};

int addFive(Distance d) {

    d.meter += 5;
    return d.meter;
}

class Duration {
    private:
        int minute;

        friend int addFive(Duration);

    public:
        Duration() : minute(0) {}

};

int addFive(Duration d) {

    d.minute += 5;
    return d.minute;
}

int main() {
    Duration Du;
    cout << "Duration: " << addFive(Du);

    Distance D;
    cout << "Distance: " << addFive(D);

    return 0;
}
```

**11. write a program to handle Division by zero Exception in C++**

```
#include <iostream>
#include <stdexcept> // To use runtime_error
using namespace std;

// Defining function Division
float Division(float num, float den)
{
        // If denominator is Zero
        // throw runtime_error
        if (den == 0) {
                throw runtime_error("Math error: Attempted to divide by Zero\n");
        }

        // Otherwise return the result of division
        return (num / den);

} // end Division
```

```
int main()
{
        float numerator, denominator, result;
        numerator = 12.5;
        denominator = 0;

        // try block calls the Division function
        try {
                result = Division(numerator, denominator);

                // this will not print in this example
                cout << "The quotient is "
                        << result << endl;
        }

        // catch block catches exception thrown
        // by the Division function
        catch (runtime_error& e) {

                // prints that exception has occurred
                // calls what function
                // using runtime_error object
                cout << "Exception occurred" << endl
                        << e.what();
        }

} // end main
```

**12. Program on Bank Application**

```
#include<iostream>
#include<conio.h>
#include<string>
#include<stdlib.h>
using namespace std;
int n,i,k,m,s;
class bank
{
   int acc_no;
   string name;
   string password;
   int balance;
   public:
      bank()
      {
         acc_no=0;
      }
      void create_account();
      void deposit();
```

```
        int generate_accno(int i);
        void withdraw();
        void transfer (bank);
        void display();
        static void sort(bank a[]);
        void delete_account();
};

int main()
{
bank b[100];
cout<<"enter the number of customers: ";
cin>>n;
int choice;
do
{
cout<<"1. create account 2. dispaly all accounts 3. deposit 4.withdraw 5.transfer 6.display based on
account number 7.sorting based on balance 8.delete account 9. exit "<<endl;
cout<<"enter your choice: "<<endl;
cin>>choice;
switch (choice)
{
case 1:for(i=0;i<n;i++)
        {
           b[i].create_account();

        }
        break;
case 2:for(i=0;i<n;i++)
        {
           b[i].display();
        }
        break;
case 3: cout<<"enter the account number: ";
        cin>>k;
        b[k].deposit();
        break;
case 4: cout<<"enter the account_no:";
        cin>>k;
        b[k].withdraw();
        break;
case 5: cout<<"enter from account_no:";
        cin>>k;
        cout<<"enter to account no:";
        cin>>m;
        b[k].transfer(b[m]);
        break;
case 6: cout<<"enter the account number to display the details:";
        cin>>k;
        b[k].display();
        break;
case 7: bank::sort(b);
        break;
case 8: cout<<"enter the account number to delete:";
        cin>>k;
        b[k].delete_account();
        break;
default:exit (0);
}
cout<<"press 1 to continue"<<endl;
cin>>s;
}
while(s==1);
return 0;
}
void bank::create_account() //create account
{
    cout<<"enter"<<i+1<<"customer's details:"<<endl;
    cout<<"name: "<<endl;
    cin>>name;
    acc_no=generate_accno(i);
    cout<<"password: "<<endl;
    cin>>password;
    cout<<"balance:"<<endl;
    cin>>balance;

}
int bank::generate_accno(int i)//generate unique account numbers
{
    acc_no=acc_no+i;
    i++;
    return acc_no;
}
void bank::display() //display the details of all customers
{
    cout<<"customer details: "<<endl;
    cout<<"name: "<<name<<endl;
    cout<<"account number:"<<acc_no<<endl;
    cout<<"balance: "<<balance<<endl;
}

void bank::deposit()//deposit the amount
{
    string p;
    int amount;
    cout<<"enter the password:";
    cin>>p;
    if (p==password)
    {
        cout<<"enter the amount to deposit:";
        cin>>amount;
        balance=balance+amount;
        cout<<"balance: "<<balance;
    }
    else
    {
        cout<<"incorrect password..!!"<<endl;

    }

}
void bank::withdraw()//withdraw the amount
{
    string p;
    int amount;
    cout<<"enter the password:";
    cin>>p;
    if (p==password)
    {
        cout<<"enter the amount to withdraw: ";
        cin>>amount;
        if (amount<balance)
        {
            balance=balance-amount;
            cout<<"remaining balance: "<<balance;

        }
        else
        {
            cout<<"insufficient balance..!"<<endl;

        }

    }
        else
        {
            cout<<"incorrect password..!!"<<endl;
        }

}
void bank::transfer(bank a)//transfer the amount from one accunt to another
{
    int amount;
    string p,q;
    cout<<"enter the password for from account: "<<endl;
    cin>>p;
    cout<<"enter the password for from account: "<<endl;
    cin>>q;
    if ((p==password) && (q==a.password))
    {
        cout<<"enter the amount to transfer:";
        cin>>amount;
        if (amount<balance)
        {
            a.balance=a.balance+amount;
            balance=balance-amount;
            cout<<balance<<endl;
            cout<<a.balance<<endl;

        }
        else
        {
            cout<<"insufficient balance";
        }
    }
    else
    {
        cout<<"invalid password";
    }

}
void bank::sort(bank b[])// sort the customer details based on balance
{
    int i,j;
    bank temp;
    for(i=0;i<n;i++)
    {
        for (j=0;j<n;j++)
        {
            if (b[i].balance<b[j].balance)
            {
                temp=b[i]; b[i]=b[j]; b[j]=temp;

            }

        }

    }
    for(i=0;i<n;i++)
    {
        b[i].display();
    }
}
void bank::delete_account()// delete a specific account
{
        name="XXX";
        acc_no=-1;
        balance=-1;

}
```