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.
4. 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.

Q1. 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.

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

struct Student{
    string name;
    int roll_num;
    int m[10];
    int total;
    float average = 0;
}s[3], temp;

int i, j, n;
void input();
void calculate();
void sort();
void display();


void input(){

    for ( i = 0; i < n; i++)
    {
        cout<<"Enter your name ";
        cin>>s[i].name;

        cout<<"Enter your roll no ";
        cin>>s[i].roll_num;

        for ( j = 1; j <= 5; j++)
        {
            cout<<"Enter marks "<<j<<" subject: ";
            cin>>s[i].m[j];
        }
    }
```

```
}

void calculate(){

    int t=0;

    for ( i = 0; i < n; i++)
    {
        for ( j = 0; j < 5; j++)
        {
            t = t + s[i].m[j];
        }
        s[i].total = t;
        s[i].average=s[i].total/5;
    }

}

void sort(){
    for ( i = 0; i < n; i++){
        for ( j = i+1; j < n; j++){
            if(s[i].roll_num>s[j].roll_num){
                temp = s[i];
                s[i] = s[j];
                s[j] = temp;
            }
        }
    }
}

void display(){
    cout<<""<<endl;
    cout<<"Student Information"<<endl;

    for ( i = 0; i < n; i++){

        cout<<"Roll No: "<< s[i].roll_num<<endl;
        cout<<"Name: "<< s[i].name<<endl;

        for ( j = 1; j <= 5; j++){

            cout<<"Marks in "<< j<<" subject is: "<<s[i].m[j]<<endl;
        }
        cout<<"Total is "<<s[i].total<<endl;
        cout<<"Average is "<<s[i].average<<endl;
    }

}
```

```
int main(){
 cout<<"Enter the number of students: ";
    cin>>n;
    input();
    calculate();
    sort();
    display();
 return 0;
}
```

Output

Enter the number of students: 1

Enter your name Aman

Enter your roll no 127

Enter marks 1 subject: 30

Enter marks 2 subject: 33

Enter marks 3 subject: 34

Marks in 4 subject is: 36

Marks in 5 subject is: 37

Total is 133

Average is 26

Q2. 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.

4. 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
*/
```

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

int i, j, n;
void input();
void calculate();
void sort();
void display();
void highesttax();
void lowesttax();

struct employee{
    int empid;
    char empname[10];
    int grade;
```

```
    float da, hra, tax, basic, grosssal, netsal;
} e[10], temp;

void input(){
    for (i = 1; i <= n; i++)
    {
        cout << "Enter empid: ";
        cin >> e[i].empid;
        cout << "Enter empname: ";
        cin>>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"<<endl;

        // 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;
            }
        }
    }
}

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;
        cout<<" "<<endl;
    }
}
```

```
void highesttax()
{
    cout<< " "<<endl;
    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;
        }
    }
}

void lowesttax()
{
    cout<< " "<<endl;
    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;
            cout<<" "<<endl;
        }
    }
}
int main()
{
    cout <<"Enter number of records: "<<endl;
    cin >> n;
    input();
    calculate();
    sort();
    display();
    highesttax();
    lowesttax();
    return 0;
}
```

Output

Enter number of records:

2

Enter empid: 121

Enter empname: Aman

Enter grade: 1

Enter empid: 120

Enter empname: Rohit

Enter grade: 2

43500Employee Information:

Employee name: Rohit

Empid: 120

Grade: 2

Basic salary: 25000

Gross salary: 36250

Net salary: 36250

Employee name: Aman

Empid: 121

Grade: 1

Basic salary: 30000

Gross salary: 43500

Net salary: 39150

Employee Information of highest tax payer:

Employee name: Aman

Empid: 121

Grade: 1

Tax value: 4350

Employee Information of low tax payer:

Employee name: Rohit

Empid: 120

Grade: 2

Tax value: 0

Q3 a. Write program to implement function handling in C++ to add two numbers using with arguments and with return values.

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

int addition(float a, float b){
    return a+b;
}

int main(){

    float a, b;
    cout<<"Enter the first number: ";
    cin>>a;
    cout<<"Enter the second number: ";
    cin>>b;

    float result;
    result = addition(a, b);
    cout<<"Sum of "<<a<<" + "<<b<<" = "<<result;

    return 0;
}
```

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

```cpp
#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);
}
```

Output

Enter side of a square: 2

Enter radius of circle: 1.1

Enter base and height of triangle: 1

1

Enter length and breadth of rectangle: 1

1

Area of square is 4

Area of circle is 3.7994

Area of triangle is 0.5

Area of rectangle is 1

4. a) Write a program in C++ to prepare a student Record using class and object.

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

class student
{
    int rno;
    char name[10];

public:
    void getdata()
    {

        cout << "Enter rno, name ";
        cin >> rno >> name;
    }

    void putdata()
    {
        cout << "Name = ";
        cout << name<<endl;
        cout << "Roll no = ";
        cout << rno;
    }
};

int main()
{
    student s1;
    s1.getdata();
    s1.putdata();

    return 0;
}
```

Output

Enter rno, name 1

Aman

Name = Aman

Roll no = 1

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

```
#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 rectangle = " << c << endl;
    }
    int display(){
        cout << "length=" << length << endl;
        cout << "breadth=" << breadth;
    }
};

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

    return 0;
}
```

Output

area of rectangle = 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{
    public:
    string name;
    string designation;
    int eno;
    float bp, hra, da, pf, np;

    void getdata(){
        cout << "Enter the employee name: ";
        cin>>name;
        cout << "Enter the designation: ";
        cin>>designation;
        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 sale percentage: ";
        cin >> y;
    }

    void display()
    {
    cout << "\t Name " << name<< "\n";
    cout<< "\t Designation " << "\t" << designation<< "\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;
}
```

Output

Enter the employee name: Aman

Enter the designation: Clerk

Enter the basic pay: 30000

Enter the Humen Resource Allowance: 5000

Enter the Dearness Allowance: 3000

Enter the Profitablity Fund: 1000

Enter the sale percentage: 10

Name Aman

Designation Clerk

Basic Pay 30000

HRA 5000

DA 3000

PF 1000

Gross Pay 37000

Net Pay = 40700

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.

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

class Student{
public:
    string name;
    int id;
    int age;
    void read_data();
};

class ug_students:public Student{
public:
    int stipend, sem;
    float fees;
    void read_data();
};
```

```
class pg_students:public ug_students{
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 >> id;
    cout << "\n Enter age: ";
    cin >> age;
}

void ug_students::read_data(){
    Student::read_data();
    cout << "\n Enter semester: ";
    cin >> sem;
    cout << "\n Enter stipend: ";
    cin >> stipend;
    cout << "\n Enter fees: ";
    cin >> fees;
}

void pg_students::read_data(){
    Student::read_data();
    cout << "\n Enter semester: ";
    cin >> sem;
    cout << "\n Enter stipend: ";
    cin >> stipend;
    cout << "\n Enter fees: ";
    cin >> fees;
}

int main(){
    ug_students ug[20];
    pg_students 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;
}
```

Ouput

Enter the no. of entries in the ugstudent class: 1

Enter name: Aman

Enter Reg.no. 127

Enter age: 18

Enter semester: 3

Enter stipend: 0

Enter fees: 200000

Average of age of sem 3 is 18

Enter the no. of entries of pgstudent class:

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.

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

const int LEN = 80; // maximum length of names
class employee{

    private:
    char name[LEN];       // employee name
    unsigned long number; // employee number

    public:
    void getdata()
    {
        cout << "\n Enter the 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 << "\nNumber of publications: " << pubs;
    }
};

class laborer : public employee // laborer class
{
    //Nothing
};

int main(){

    manager m1, m2;
    scientist s1;
    laborer l1;
    cout << endl; // get data for several employees
    cout << "\nEnter data for manager 1 ";
    m1.getdata();

    cout << "\nEnter 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;
}
```

Output

Enter data for manager 1

Enter the name: Aman

Enter number: 121

Enter title: Manger

Enter dues: 10000

Enter data for manager 2

Enter the name: Elon

Enter number: 20

Enter title: CEO

Enter dues: 10000

Enter data for scientist 1

Enter the name: Albert

Enter number: 10

Enter number of publications: 200

Enter data for laborer 1

Enter the name: 22

Enter number: 21

Data on manager 1

Name: Aman

Number: 121

Title: Manger

Dues Details: 10000

Data on manager 2

Name: Elon

Number: 20

Title: CEO

Dues Details: 10000

Data on scientist 1

Name: Albert

Number: 10

Number of publications: 200

Data on laborer 1

Name: 22

Number: 21

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

```cpp
#include <iostream>

using namespace std;

class distances
{
    int feet;
    float inches;

public:
    distances()
    {
        feet = 0;
        inches = 0.0;
    }
    void input();
    void formula();
    void display();
};

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

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

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

int main()
{
    distances d1;
```

```
    d1.input();
    d1.formula();
    d1.display();
    return 0;
}
```

Output

Enter feet and inches: 1

12

feet: 2

inches: 0

//Adding two objects

```cpp
#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();
}
```

Output

Enter feet and inches: 2

1

Feet: 1

Inches: 1

Feet: 3

Inches: 2

//Adding three 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();
}
```

Output

Enter feet and inches: 2

1

feet: 1

inches: 1

feet: 2

inches: 2

feet: 5

inches: 4

Q7) 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;
}
```

Output

name: ABC

age: 21

salary: 7879

research_area: Comp

Q8) Write a C++ program to compute the area of triangle and rectangle using pure virtual function

```cpp
#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

Q9 . Write program to swap the numbers using the concept of function template.

```
#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

1

2

Before swapping 1 2

After Swapping 2 1

Q10. Write a C++ program to Overload Binary + operator to add two Distance class objects.

```
#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;
}
```

Output

Enter first distance:

Enter feet: 2

Enter inches: 12

Enter second distance:

Enter feet: 1

Enter inches: 24

Total Distance:

Feet: 6 Inches: 0

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

```
#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)<<endl;

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

    return 0;
}
```

Output

Duration: 5

Distance: 5

Q11. 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
```

Output

Exception occurred

Math error: Attempted to divide by Zero

Q12. Write A 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;
}
```