Experiment - 1

1. WAP to declare a class student having data members as name, accept & display data for one student

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

class Student {
    string name;
    int roll_no;

public:
    void accept ();
    void display ();
};
void Student :: accept () {
    cout << "enter name and roll no.";
    cin >> name >> roll_no;
}

void Student :: display () {
    cout << "\n Student name :" << name;
    cin << "\n Student roll no :" << roll_no;
}

int main () {
    Student S1;
    S1.accept ();
    S1.display ();
    return 0;
}
```

2. WAP to declare a class book having data members as id, name, price. Accept data for 2 books & display data of book having greater price

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

class Book {
    int id;
    string name;
    float price;
    public:
        void accept () {
    cout << "ID: "; cin >> id; cin.ignore ();
    cout << "Name: "; getline (cin, name);
    cout << "Price: "; cin >> price;
}
void display () {
    cout << "\n ID: " << id << "\n Name: " << name
    << "\n Price: " << price << endl;
}
float getPrice () { return price; }
};
int main () {
    Book b1, b2;
    cout << "Book 1: \n"; b1.accept ();
    cout << "\n Book 2: \n"; b2.accept ();

    cout << "\n Book with higher price: \n";
    if (b1.getPrice () >= b2.getPrice ()) b1.display ();
    else b2.display ();
    return 0;
}
```

5. WAP to declare a class time having data members as H, M, & S. Accept data for ~~the~~ object & display total time in seconds

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

class Time {
    int H, M, S;

public:
    void accept () {
        cout << " enter hours, minutes, and seconds:
        cin >> H >> M >> S;

    }
    int totalSeconds () {
        return H * 3600 + M * 60 + S;
    }
};

int main () {
    Time t;
    t.accept ()
    cout << " total time in seconds: " << t.tot
    seconds () << endl;

    return 0;

}
```

14/9

Exp - 2

① WAP to declare a class 'city' having data members city name & population. Accept this data for 5 cities & display name of city having highest population.

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

class city {
public:
string name;
int p;
void accept()
{
cout << "enter city name & population";
cin >> name >> p;
}
};
int main()
{
city c[5];
for(int i=0; i<5; i++)
{
c[i].accept();
}
int maxIndex = 0;
for (int i=1; i<5; i++) {
if (c[i].p > c[maxIndex].p) {
maxIndex = i;
}
}
cout << "city with highest population:" << c[
Index].name << endl;
return 0;
}
```

② WAP to declare a class account having data members as account no. & balance. Accept this data for 10 accounts and give interest of 10% where balance is equal or greater than 5000 and display them.

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

class Account {
  int acc_no;
    float balance;

public:
    void getData () {
    cout << "enter account number : ";
    cin >> acc_no;
    Cout << "enter balance : ";
    cin >> balance;
  }
  void applyInterest () {
      if (balance >= 5000) {
        cout << "account number: " <<
      balance += balance * 0.10;
    }
  }

  void display () {
      if (balance >= 5000) {
        cout << "Account Number: " << acc_no
      }      << "Balance w/ interest:" << balance << end
    }
};

int main () {
  Account acc[10];
  cout << "enter details for 10 accountants: \n";
```

```
for (int i=0; i<10 ; i++) {
    cout << "\n Account " << i+1 << " :\n ";
    acc[i] .getData ();
}
cout << "\n Accounts with balance >= 5000 qualifying for
    adding 10 % interest:\n ";
for (int i=0, i<10 ; i++) {
    acc[i] .applyInterest ();
    acc[i] .display ();
}
return 0;
}
```

(8) WAP to declare a class staff having data members as name and post. Accept this data for 5 staff and display names of staff who are "HOD".

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

class Staff {
  public:
    string name;
    string post;

  void getData () {
      cout << "enter name : ";
       cin >> name;
      cout << "enter post : ";
       cin >> post;
  }
    void showIfHOD () {
      if (post == 'HOD' || post == 'hod')
       cout << 'HOD Name: " << name << endl;
  }
};
int main () {
    Staff s[5];

for (int i=0; i<5; i++) {
    cout << "\n Enter details of staff " << i+1 << ":
    s[i].getData ();
}
    cout << "\n List of staff who are HOD:\n";
    for (int i=0; i<5; i++) {
      s[i].showIfHOD ();
    }
    return 0;
}
```

Exp - 3

① WAPJO declare class Book containing data members as book-title, author name & price. Accept & display the information for one object using a pointer to that - object

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

class Book {
    string title;
    string author;
    float price;

public:
    void accept () {
    cout << "enter book title";
    getline (cin, title);
    cout << "enter author name: ";
    getline (cin, author);
    cout << "enter price";
    cin >> price;
    cin.ignore ();
}
    void display () {
    cout << "\n Book Details: \n";
    cout << "Title: " << title << "\n";
    cout << "Author " << author << "\n";
    cout << "Price: $" << price << "\n";
}
};
int main () {
    Book b;

    Book *ptr = &b;
```

parameterized
default, copy

```
ptr -> accept ();
ptr -> display ();

return 0;

}
```

② WAP to declare a class student having data members roll no. & percentage. using 'this' pointer invoke member functions to accept & display this data for one object of the class

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

class Student {
private:
    int roll_no;
    float percentage;

public:
    void accept () {
        cout << "enter roll no: ";
        cin >> this -> roll_no;
        cout << "enter percentage: ";
        cin >> this -> percentage;
    }
    void display () {
        cout << "\n Roll NO: " << this -> roll_no << endl;
        cout << "percentage: " << this -> percentage << "%" << endl;
    }
};

int main () {
    Student s;
    s.accept ();
    s.display ();

    return 0;
}
```

(3) WAP to demonstrate the use of nested

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

class OuterClass {
    int outerData;

public:
    OuterClass(int val) : outerData(val)

    class InnerClass {
        int innerData;

    public:
        InnerClass(int val) : innerData(val)

        void display(OuterClass& outer) {
            cout << "Inner Data: " << innerData << e
            cout << "Accessing Outer Data from inner class: "
                 << outer.outerData << endl;
        }
    };
};

int main() {
```

Exp-4

1. WAP to swap two nos. from same class using object as function argument. Write swap function as member function

code:

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

class Number {
  private:
    int value;

  public:
    Number(int v) { value = v; }

    void swapNumbers(Number &n) {
        int temp = value;
        value = n.value;
        n.value = temp;
    }
    void display() {
        cout << value << endl;
    }
};
int main() {
  int a, b;
  cout << "enter first no:";
  cin >> a;
  cout << "enter second no:";
  cin >> b;
```

```
Number num1(a), num2(b);

cout << "\n Before swapping :" << endl;
cout << "Num 1 = "; num1.display();
cout << "Num 2 = "; num2.display();

num1.swapNumbers(num2);

cout << "\n After swapping :" << endl;
cout << "Num 1 = "; num1.display();
cout << "Num 2 = "; num2.display();

return 0;
}
```

2. WAP to swap two nos from same class using concept of friend function.

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

class Number {
  private :
    int value;

  public :
  Number (int v) { value = v; }

  friend void swapNumbers (Number &n1, Number &n2)
{ int temp = n1.value;
  n1.value = n2.value;
  n2.value = temp;
}

int main () {
int a, b;
 cout << "enter first no:";
 cin >> a;
 cout << "enter 2nd no:";
 cin >> b;

Number num1 (a), num2 (b);

cout << "\n Before swapping:" << endl;
cout << "Num1 = "; num1.display ();
cout << "Num2 = "; num2.display ();

swapNumbers (num1, num2);
```

```
cout << "\n After swapping : " << endl ;
cout << " Num 1 = " ; num1.display ();
cout << " Num 2 = " ; num2.display ();

  return 0 ;

}
```

3 WAP to swap two numbers from diff... class using friend function.

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

class B;
class A {
  private:
    int valueA;

  public:
    A(int v) { valueA = v; }

    friend void swapNumbers (A &x, B &y);
    void display () { cout << valueA << endl;
};
class B {
  private:
    int valueB;

  public:
    B(int v) { valueB = v }

friend void swapNumbers (A &x, B &y);
    void display () { cout << valueB << endl
};

void swapNumbers (A &x, B &y) {
    int temp = x.valueA;
    x.valueA = y.valueB;
    y.valueB = temp;
}
```

```
int main ( ){
int a, b;
cout << " enter no. in class A: ";
cin >> a;
cout << " enter no. in class B: ";
cin >> b;

A objA(a);
B objB(b);

cout << " \n Before swapping: " << endl;
cout << "A:" ; objA.display ();
cout << " B : "; objB.display ();

swapNumbers (objA, objB);

cout << " \n After swapping: " << endl;
cout << "A:" ; objA.display ();
cout << " B : "; objB.display ();

return 0;

}
```

4. WAP to create Result1 & Result2 which store marks of the student. Read the value of a member both the class objects & compute the average of

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

class Result2;

class Result1 {
    private:
        float marks1;
    public:
        void readMarks() {
            cout << "enter marks for result1:";
            cin >> marks1;
        }
    friend void computeAverage(Result1 r1, Result2 r2);
};

class Result2 {
    private:
        float marks2;
    public:
        void readMarks() {
            cout << "enter marks for result2:";
            cin >> marks2;
        }
    friend void computeAverage(Result1 r1, Result2 r2);
};

void computeAverage(Result1 r1, Result2 r2) {
    float avg = (r1.marks1 + r2.marks2) / 2;
    cout << "Average marks:" << avg << endl;
}
```

```
int main ()
{
  Result1 obj1;
  Result2 obj2;

  obj1.readMarks();
  obj2.readMarks();

  computeAverage(obj1, obj2);

  return 0;
}
```

5. WAP to find the greatest number among numbers from two different classes using friend function.

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

class B;

class A {
private:
    int numA;

public:
    A(int n) { numA = n; }
    friend void findGreatest(A a, B b);
};

class {
private:
    int numB;

public:
    B(int n) { numB = n; }
    friend void findGreatest(A a, B b);
};

void findGreatest(A a, B b) {
    if (a.numA > b.numB)
        cout << "Greatest no: " << a.numA
             << endl;
    else if (b.numB > a.numA)
        cout << "Greatest no: " << b.numB << endl;
    else
        cout << "Both nos are equal: " << a.numA << endl;
}
```

```
int main() {
  int x, y;
  cout << "enter number for class A: ";
  cin >> x;
  cout << "enter no. for class B: ";
  cin >> y;

A objA(x);
B objB(y);

findGreatest (objA, objB);

  return 0;

}
```

Q.

1919

Exp-4 (Contd)

~~Exp-5~~

① Create two classes, ClassA & ClassB each a private integer. Write a friend function that can access private data from both classes and the sum

```
#include <stdio.h>
using namespace std;

class ClassB;

class ClassA {
private:
    int numA;
public:
    classA(int a) {
    numA = a;
    }
friend int sum(ClassA, ClassB);
};

class ClassB {
private:
    int numB;
public:
    ClassB(int b) {
    numB = b;
    }
};
friend int sum(ClassA, ClassB);

int sum(ClassA objA, ClassB objB) {
    return objA.numA + objB.numB;
}
```

```
int main() {
    int a, b;
    cout << "enter first no. (Class A): ";
    cin >> a;
    cout << "enter second no. (Class B): ";
    cin >> b;

    ClassA objA(a);
    ClassB objB(b);

    cout << "Sum: " << sum(objA, objB) << endl;

    return 0;
}
```

② WAP with a class Number that contains a integer. Use a friend function swapNumbers to swap the private values of two Number objects.

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

class Number {
  private:
    int value;
  public:
    Number(int v) {
        value = v;
    }
    void display() {
        cout << value;
    }
    friend void swapNumbers(Number &n1, Number &n2);
};
void swapNumbers(Number &n1, Number &n2) {
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp;
}
int main() {
    int a, b;
    cout << "enter first no: ";
    cin >> a;
    cout << "enter second no: ";
    cin >> b;

    Number num1(a), num2(b);
```

```
cout << "\n Before swapping:" << endl;
cout << "num1 = ";
num1.display();
cout << "num2 = ";
num2.display();

swapNumbers(num1, num2);

cout << "\n After swapping:" << endl;
cout << "num1 = ";
num1.display();
cout << "num2 = ";
num2.display();

return 0;
}
```

3. Define two classes box & cube, each having a volume. Write a friend functions findGreater() that determines which object has a larger

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

class cube;
class box {
private:
    int v
public:
    Box (int v)
    friend void findGreater (Box, Cube);
};
class cube {
private:
    int volume;

public:
    cube (int v)
    friend void findGreater (box, cube);
};
void findGreater (Box b, Cube c) {
    if (b.volume > c.volume) {
    cout << " box has a larger volume: << b.volu
}
else if (c.volume > b.volume) {
cout << " cube has a larger volume: " << c.volume << e
}
else {
    cout << " both have the same volume: " << b.volu
}
}
```

```
int main() {
    int boxVol, cubeVol;
    cout << "enter volume of box: ";
    cin >> boxVol;
    cout << "enter volume of cube: ";
    cin >> cubeVol;

    Box b(boxVol);
    cube c(cubeVol);
    findGreater(b, c);

    return 0;
}
```

Complex

4. Create a class ~~Student~~ w/ real and imaginary parts as private members. use a friend function to add two complex nos. and return as a new Complex object

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

class Complex {
private:
    float real;
    float imag;

public:
    Complex(float r=0, float i=0) {
        real = r;
        imag = i;
    }
    friend Complex addComplex(Complex, Complex);
    void display() {
        cout << real << " , " << imag << " , " << en
    }
};

Complex addComplex(Complex c1, Complex c2) {
    Complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;

    return result;
}
```

```
int main () {
    float r1, i1, r2, i2;
cout << "enter real & imaginary part of first
        complex no.: ";
cin >> r1 >> i1;
cout << "enter real & imaginary part of second complex
        no.: ";
cin >> r2 >> i2;

Complex c1 (r1, i1);
Complex c2 (r2, i2);

Complex sum = add Complex (c1, c2);

cout << "sum of complex numbers: ";
sum.display ();

return 0;

}
```

5. Create a class Student w/ private data members and three subject marks. Write a friend function calculateAverage(Student) that calculates and displays the average marks

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

class Student {
private:
    string name;
    float mark1, mark2, mark3;

public:
Student(string n, float m1, float m2, float m3) {
    name = n;
    mark1 = m1;
    mark2 = m2;
    mark3 = m3;
}
friend void calculateAverage(student s);
};

void calculateAverage(student s) {
    float avg = (s.mark1 + s.mark2 + s.mark3)/3.0;
    cout << "\n Student Name: " << s.name;
    cout << "\n Average Marks: " << avg << endl;
}
int main() {
    string name;
    float m1, m2, m3
    cout << "enter student name: ";
    getline(cin, name);
```

```
cout << "Enter marks for three subjects: ";
cin >> m1 >> m2 >> m3;

Student sda(name, m1, m2, m3);

calculateAverage (sda);

return 0;

}
```

6. Create three classes: Alpha, Beta and Gamma each w/ a private data member. Write a friend that can access all three and print sum.

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

class Beta;
class Gamma;

class Alpha {
  private:
    int a;
  public:
    Alpha (int x) { a = x; }
    friend void sumValues (Alpha, Beta, Gamma);
};

class Beta {
  private:
    int b;
  public:
    Beta (int y) { b = y; }
    friend void sumValues (Alpha, Beta, Gamma);
};

class Gamma {
  private:
    int c;
  public:
    Gamma (int z) { c = z; }
    friend void sumValues (Alpha, Beta, Gamma);
};

void sumValues (Alpha x, Beta y, Gamma z) {
```

```
    int sum = x.a + y.b + z.c;
    cout << " sum of all values " << sum << endl;
}
int main() {
    int p, q, r;
    cout << "enter value for Alpha: ";
    cin >> p;
    cout << "enter value for Beta : ";
    cin >> q;
    cout << "enter value for Gamma : ";
    cin >> r;

Alpha objA (p);
Beta objB (q);
Gamma objC (r);

sumValues (objA, objB, objC);

return 0;
}
```

7. Create a class point w/ private members x & y. Write a friend function that calculates and the distance b/w two point objects

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

class Point {
private:
    float x, y;

public:
    Point(float a, float b) {
        x = a;
        y = b;
    }
    friend float distance(Point p1, Point p2);
};
float distance(Point p1, Point p2) {
    return sqrt((p2.x - p1.x) * (p2.x - p1.x) +
                (p2.y - p1.y) * (p2.y - p1.y));
}
int main() {
    float x1, y1, x2, y2;
    cout << "enter x & y for first pt: ";
    cin >> x1 >> y1;
    cout << "enter x & y for second pt: ";
    cin >> x2 >> y2;

    Point p1(x1, y1), p2(x2, y2);

    cout << "Distance" << distance(p1, p2);
```

8. Create two classes: BankAccount and Audit. BankAccount holds private balance information. Write a friend function in Audit that accesses and prints balance information for auditing.

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

class Audit;

class BankAccount {
  private:
    double Balance;

  public:
    BankAccount(double b)
    friend void auditAccount(BankAccount);
};
class Audit {
};
void auditAccount(BankAccount acc) {
    cout<<"Balance: " << acc.balance<<endl
}
int main() {
    double bal;
    cout<<"enter account balance:";
    cin>>bal;

BankAccount account(bal);
auditAccount(account);

return 0;
```

Experiment – 5

1. WAP to find the sum of nos b/w 1 to n using a constructor where the value

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

class number {
    int num;
public:
    number () {
        cout << "enter a no : " << endl;
        cin >> num;

    int sum = 0;
    for (int i = 1; i < num ; i++) {
        sum = sum + i;
    }
    cout << " Sum of number upto << num << is:
        << sum << endl;

    }
};
int main () {
    number n;
    return 0;
}
```

Output :

enter a no : 3

the sum of numbers upto 3 is 6.

OOP Chapter 1 & 2 Notes

1. WAP to declare a class "student" having data members as name & percentage. Write a constructor to initialize these data members. Accept & display data for one student

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

class Student {
    string name;
    float per;
    public:
    Student (string n, float p) {
        name = n;
        per = p;
    }

void display () {
    cout << "Name : " << name << endl;
    cout << "Percentage : " << per << endl;
} };
int main () {
    Student s ("Soham", 94.3);
    s. display ();

    return 0;
}
```

Output:

Name : Soham
Percentage : 94.3

3. Define a class "college" members variables as roll no, name, course. WAP using constructor with default value as "computer engineering" for course. Accept & display this data for 2 objects of class.

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

class college {
    int roll;
    string name;
    string course;
    public:
    college (int r=00, string n = "Unknown", string
             c = "Computer Engineering")
    {
        roll = r;
        name = n;
        course = c;
    }
    void display() {
        cout << "Roll no:" << roll << endl;
        cout << "Name:" << name << endl;
        cout << "course" << course << endl;
    }
};
int main() {
    college s1 (46, "Soham");
    college s2 (44, "Varad");

    s1.display();
    s2.display();

    return 0;
}
```

Output :
Roll no : 46
Name : Soham
Course : Computer engineering
Roll no : 44
Name : Varad
course : Computer engineering

4. WAP to demonstrate constructor overloading.

```
#include <iostream>
using namespace std;
class Student {
    int roll;
    string name;
public:
    Student() {
        name = "Unknown";
        roll = 0;
    }
    Student(string n, int r) {
        name = n;
        roll = r;
    }
    void display() {
        cout << "Name " << name << endl;
        cout << "Roll No: " << roll << endl;
    }
};
int main() {
    Student s1;
    Student s2 ("Soham");
    Student s3 ("Varad", 18);

    s1.display();
    s2.display();
    s3.display();

    return 0;
}
```

Output :
Name : Unknown
Roll no : 0
Name : Soham
roll no : 0
Name : Vasrad
roll no : 18

Experiment - 6

1. WAP to implement multilevel inheritance. Assume suitable data

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

class department {
protected:
    string dname;
};

class student : protected department {
protected:
    string sname;
    int sroll;
};

class marks : protected student {
    int m1, m2, percentage;
public:
    void accept () {
        cout << "enter department : " << endl
        cin >> dname;
        cout << "enter name : " << endl;
        cin >> name;
        cout << "enter marks 1 : " << endl;
        cin >> m1;
        cout << "enter marks 2 : " << endl;
        cin >> m2;
```

```
void calculate (){
int per (m1 + m2)/2
cout << " Department : " << dnam << endl;
cout << " Name : " << snam << endl;
cout << " Percentage : " << per << endl;
  }
};
int main (){
  marks m;
  m.accept()
  m.calculate();
  return 0;
}
```

Output:

enter department:
~~Soham~~ CSE
enter marks1:
88
enter marks2:
Department: CSE
Name: Soham
Percentage: 88

2 WAP to implement multiple inheritance. Assume suitable data

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

class department {
protected:
    string dname;
};
class student {
protected:
    string sname;
    int roll;
};

class marks : protected department, protected student {
    int m1, m2, percentage;
public:
    void accept() {
        cout << "enter department :" << endl;
        cin >> dname;
        cout << "enter name :" << endl;
        cin >> sname;
        cout << "enter M1 " << endl;
        cin >> m1;
        cout << "enter M2 :" << endl;
        cin >> M2
    }
    void calculate() {
        int per = (m1 + m2) / 2;
        cout << "dept. " << dname;
        cout << "name " << sname << endl;
        cout << "percentage :" << per << endl;
```

```
};
int main() {
marks m;
m.accept();
m.calculate();

}
```

5. WAP to implement hierarchial inheritance

```
#include <iostream>
using namespace std;
class Person {
  protected:
   string name;
   int age;
  public:
  void acceptPer() {
    cout << "enter name:" << endl;
    cin >> name;
    cout << "enter age:" << endl;
    cin >> age
  }
};
class student: public Person {
    int roll;
   float per;
   public:
   void acceptstud() {
    cout << "enter roll no:" << endl;
    cin >> roll;
    cout << "enter percentage" << endl;
    cin >> per;
```

```
void display stud () {
    cout << " Name " << name << endl;
    cout << " Age " << age << endl;
    cout << " Roll no " << roll << endl;
    cout << " percentage " << percent << endl;
}
};

class studd : public person {
    int emp_id;
    string subject;
    public:
    void acceptstaff () {
        cout << " enter employee ID: " << endl;
        cin >> emp_id;
        cout << " enter subject " << endl;
        cin >> subject;
    }
    void displaystaff () {
        cout << " name " << name << endl;
        cout << " age : " << age << endl;
        cout << " emp id " << emp_id << endl;
        cout << " subject " << subject << endl;
    }
};

int main () {
    Student s;
    Staff t;
    s.acceptdep ();
    s.acceptstud ();
    t.acceptper ();
    t.acceptstaff ();
```

s. display stud();
t. display staff();

return 0;
}

4. WAP to implement hybrid inheritance

#include <iostream>
using namespace std;

class college {
protected:
string cname;
};
class employee : protected college {
protected:
string emp_name;
int id;
};

class staff : public employee {
string name;
int deptid;
public:
void accept emp() {
cout << "enter clg name" << endl;
cin >> cname;
cout << "enter emp name :" << endl;
cin >> emp_name;
cout << "enter id" << endl;
cin >> id;
cout << "enter staff name :" << endl;
cin >> name;

```
    cout << " enter dept aid : " << endl;
    cin >> dept id;
}
void display emp () {
cout << "college : " << cname << endl;
cout << "emp name " << empname
cout << " ID " << id << endl;
cout << " staff name " << sname
cout << " dept id " << deptid << endl;
}
};
class student : protected college {
    string stu_name;
    int roll;
    public:
    void accept stud () {
     cout << " enter college name
     cin >> cname;
    cout << " enter name " << endl;
    cin >> stu_name;
    cout << " enter roll no " << endl;
    cin >> roll;
}
void displayStud () {
    cout << " college : " << cname << endl;
    cout << " student name : " << sname << endl;
    cout << " roll no : " << roll << endl;
int main () {
    staff stf;
    staff. accept Emp ();
    staff. display Emp ();
```

```
Student stud1;
stud1.acceptStud();
stud1.displayStud();

return 0;
}
```

WAP to demonstrate virtual base class assume suitable data w/ figures

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

class CollegeStudent {
protected:
    int student_id;
    string ccode;
public:
    void accept() {
        cout << "enter Student ID ";
        cin >> student_id;
        cout << "entercollege code:";
        cin >> ccode;
    }

    void display() {
        cout << "Student ID:" << student_id
        cout << "college code " << ccode << endl;
    }
};

class Test : virtual public CollegeStudent {
protected:
    float percentage;
```

```
collegeStudent::accept();
cout<<"enter test percentage";
cin>>percentage;
}
void display(){
    collegeStudent::display()
    cout<<"Test percentage"<<percentage
    <<"%."<<endl;
}
};

class sports: virtual public collegeStudent{
protected:
    char grade;
public:
    void accept(){
        cout<<"enter sports grade:";
        cin>>grade;
    }
    void display(){
        cout<<"Sports grade:"<<grade
                <<endl;
    }
};

class Result: public Test, public sports{
    float tot_marks;
public:
    void accept(){
        collegeStudent::accept();
        cout<<"enter test percentage:";
        cin>>percentage;
        cout<<"enter sports grade;";
        cin>>grade;
```

```
cout << " enter total marks ";
cin >> tot_marks;

}
void display (){
    college student :: display ();
    cout << " test percentage " << percentage
                << " . " << endl;
    cout << " sports grade : " << grade
    cout << " total marks " << tot_marks

}
};
int main (){
    Result r;
    cout << " enter student details " << endl;
    r.accept ();
    cout << " Ly student result " << endl;
    r.display ();
    return 0;
}
```

Experiment - 7

1 WAP using func. overloading to calculate the area of a laboratory & area of class room

```
#include <iostream>
using namespace std;
class Area {
public:
    float calculate(float length, float
        breadth) {
        return length * breadth;
    }
    float calculate(float side) {
        return side * side;
    }
};
int main() {
    Area a;
    cout << "area of laboratory" << a.calcula
        (100) << endl;
    cout << "area of square" << a.calculate(5)
        << endl;
```

2. WAP using func. overloading to calculate the sum of 5 float values & sum of 10 integer values.

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

class Sum {
public:
int total (int a[], int n) {
    int s=0
for (int i=0; i<n; i++)
    s+= a[i];
    return s;
}

float total (float a[], int n){
    float s=0;
    for(int i=0; i<n; i++)
    s+= a[i];
    return s;
}
};

int main(){
    sum s;
int marks[10]= { 45,56,67,78,89,90,
                76,88,92,85};
float grades[5]= { 9.2,8.7,9.5,8.9,9
cout<<"sum of 10 student marks:"<<s.total
    (marks,10) << endl;
cout<<"sum of 5 student grade points:"<<
    s.total(grades,5) << endl;
return 0;
}
```

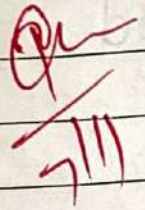
3. WAP to demonstrate the concept of friend & implement unary operator when used with the object so that the numeric data member of the class is negated.

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

class Teacher {
    int experience;
public:
    Teacher (int e) {
        experience = e;
    }

    void display () {
    cout << " experience : " << experience <<
                " year " << endl;

    }

    void operator- () {
        experience = - experience;
    }
};
int main () {
    Teacher t1 (10);
    t1.display ();
    -t1;
    cout << " After negation : ";
    t1.display ();
    return 0;
}
```

4. WAP to implement unary operator when used w/ two object so that the numeric data member of the class is incremented.

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

class Student {
    int count;
public:
    Student (int c=0) {
        count = c;
    }

    void operator ++ () {
        ++ count;
    }
    void operator ++ (int) {
        count ++;
    }
    void display {
        cout << "Student count " << count << endl;
    }
};

int main () {
    Student s1(50);
    cout << " Before increment " << endl;
    s1.display ();

    ++ s1;
    cout << " After pre-increment " << endl;
    s1.display ();
```

```
s1++
cout<<"After post increment"<<endl;
s1.display();
return 0;
}
```



Experiment - 8

WAP to overload the '+' operator so that 2 strings can be concatenated.

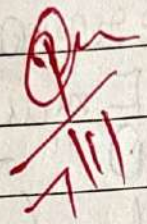
```
#include <iostream>
#include <string>
using namespace std;

class combine {
    string str;
public:
    combine(string s = "") {
        str = s;
    }
    combine operator+ (combine& obj) {
        return combine(str + obj.str);
    }
    void display() {
        cout << str << endl;
    }
};

int main() {
    combine s1("xyz"), s2("pqr"), s3;
    s3 = s1 + s2;
    cout << "concatenated string";
    s3.display()
}
```

2. WAP to create a base class login having data members name & password. Declare accept function virtual. Derive email login & membership login class from login display email login details & membership login details of the employee

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

class T login {
protected:
    string name, password;
public:
    virtual void accept() {
    cout << "enter name:";
    cin >> name;
    cout << "enter password:";
    cin >> password;
}

virtual void display() {
    cout << "Name" << name << endl;
    cout << "password" << password << endl;
};

class Emaillogin: public T login {
    string email;
public:
    void accept() override {
```

```
        cout << "enter email ID ";
        cin >> email;
        I login :: accept();
    }

void display() override {
    cout << "\n Email login details"
    cout << " email ID " << email << endl;
        I login :: display();
};}

class Membership login : public I login {
        string member ID;
    public:
        void accept() override {
            cout << " Membership login details"
                << endl;
    cout << " membership ID " << member ID << endl;
        I login :: display();
    }
};

int main() {
    I login * login;

    Email login;
    membership login;

    login = &e;
    login -> accept();
    login -> display();

    login = &m;
```

```
login -> accept();
login -> display();

return 0;

}
```



Experiment - 9

WAP to copy the contents of one file into another. Open in first textreadd mode & "second text". Assume first text is already created

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

int main () {
    ifstream infile ("first.txt");
    ofstream outfile ("second.txt");

    if (!infile) {
    cout << "error opening first.txt" << endl;
    return 1;
    }

    char ch;
    while (infile.get(ch)) {
        outfile.put(ch);
    }

    cout << "file copied successfully" << endl;

    infile.close();
    outfile.close();
    return 0;
```

2. WAP to count digits & spaces using file handling

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

int main () {
    ifstream file ("input.txt");
    if (!file) {
        cout << "error opening file";
        return 1;
    }
    char ch;
    int digits = 0, spaces = 0;

    while (file.get(ch)) {
        if (isdigit(ch))
            digits++;
        else if (isspace(ch))
            spaces++;
    }

    cout << "digits: " << digits;
    cout << "spaces: " << spaces << endl;

    file.close();

    return 0;
}
```

Q: WAP to count words using file handling

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
    ifstream file("first.txt");
    if (!file) {
        cout << "error" << endl;
        return 1;
    }

    string word;
    int count = 0;

    while (file >> word)
        count++;

    cout << "total words: " << count
    file.close();
    return 0;
}
```

12/11

4. WAP to count occurence of a given word using file handling

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

int main () {
    ifstream file ("file.txt");
    if (!file) {
        cout << "Error" << endl;
        return 1;
    }

    string word target;
    int count = 0;

    cout << "enter word to count:" << endl;
    cin >> target;

    while (file >> word) {
        if (word = target)
            count++;
    }
    cout << "Occurence of " << target
    << " : " << count << endl;

    file.close ();
}
```

Experiment -10

WAP to find sum of array using function template (e.g. Pass integer, float and double array of 10 elements)

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

template <typename T>
T findSum(T arr[], int n){
    T sum = 0
    for (int i=0; i<n; i++){
        sum += arr[i];
    }
    return sum;
}
int main(){
    int intArr[] = {1,2,3,4,5};
    float floatArr[] = {1.5, 2.5, 3.5, 4.5};
    double doubleArr[] = {2.1, 3.4, 5.6};

    int n1 = sizeof(intArr) / sizeof(intArr[0]);
    int n2 = sizeof(floatArr) / sizeof(floatArr[0]);
    int n3 = sizeof(doubleArr) / sizeof(doubleArr[0]);

    cout << "Sum of integer array: " << findSum(intArr, n1)
         << endl;
    cout << "Sum of float array: " << findSum(floatArr, n2) << endl;
    cout << "Sum of double array: " << findSum(doubleArr, n3) << endl;

    return 0;
}
```

2. a. Calculate the square of integer no. and string using template specialization.

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

template <typename T>
T square (T value) {
    return value * value;
}

template <>
string square <string> (string value) {
    return value + value;
}

int main () {
    int intVal = 5;
    double doubleVal = 3.5;
    string strVal = "Hello";

cout << "square of integer" << square (intVal)
     << endl;
cout << "square of double" << square (doubleVal)
     << endl;
cout << "Square of string: " << square (strVal)
     << endl;

    return 0;
}
```

Experiment - 10 (cont'd)

5.
```
#include <iostream>
using namespace std;

template < class T >
class Calculator {
    T a, b;
public: {
    Calculator ( T x, T y) { a = x; b = y }
    void display () {
    cout << " Sum " << a+b << endl;
    cout << " Difference " << a-b << endl;
    cout << " Product :" << a * b << endl;
    cout << " Quotient :" << a / b << endl;
    }
};
int main () {
    Calculator <int> c1 (10,5);
    Calculator < float > c2 (10.5, 5.5);
    c1.display ();
    c2.display ();
    return 0;
}
```

Output: Sum : 15
Difference : 5
Product : 50
Quotient : 2
Sum : 16
Difference : 5

4. WAP to implement push & pop methods for a stack using class template

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

template < class T >
class Stack {
    T s[10];
    int top;
public:
    Stack() { top = -1; }
    void push ( T x ) {
        if ( top < 9 ) s[ ++top ] = x;
    }
    void pop () {
        if ( top >= 0 ) top --;
    }
    void display () {
        for (int i = top; i >= 0; i++)
            cout << s [i] << " ";
        cout << endl;
    }
};

int main ( ) {
    Stack <int> s1;
    s1.push (10);
    s1.push(20);
    s1.push(30);
    s1.pop();
    s1.display ();
    return 0; }
```

Output : 30 20 10
20 10

[2|1]

Experiment - 11

Q WAP to implement generic vectors, include the member functions:-

To create the vector

① To modify the value of a given element

② To multiply by a scalar value.

③ To display the vector in the form (10,20,30)

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

template <class T>
class MyVector {
vector <T> v;
public:
    void createVector (int n) {
        T val;
    for (int i=0; i<n; i++) {
        cin >> val;
        v.push_back (val);
    }
}
void modifyValue (int index, T newVal) {
    if (index >= 0 && index < v.size())
        v[index] = newVal;
}

void multiplyByScalar (T scalar) {
    for (int i=0; i<v.size(); i++)
        v[i] *= scalar;
```

```
}
void display(){
    cout<<"(";
    for(int i=0; i<v.size(); i++){
        cout<<v[i];
        if(i!=v.size()-1)
            cout<<",";
    }
    cout<<")"<<endl;
}
};
int main(){
    MyVector<int> obj;
    int n;
    cin>>n;
obj.createVector(n);
obj.display();
obj.modifyValue(1,50);
obj.display();
obj.multiplyByScalar(2);
obj.display();
return 0;
}
```

Output:

(10,20,30)
(10,50,80)
(20,100,60)

2/11

Experiment 12

a. WAP using STL to implement stack

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

int main () {
    stack <int> s;
    s.push (10);
    s.push (20);
    cout << " Top : " << s.top() << endl;
    s.pop ();
    cout << " Top after pop :" << s.top
    return 0;
}
```

Output : Top : 30
Top after pop : 20

b. WAP using STL to implement queue

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

int main () {
  queue <int> q;
  q.push (10);
  q.push (20);
  q.push (30);
  cout << "front:" << q.front () << endl;
  q.pop ();
  cout << "front after pop:" << q.front () << endl;
  return 0;
}
```

Output: front: 10
front after pop: 20

c. Implement sorting & searching w/ user-d
records such as person record (name
telephone no), item record (item code
quantity & cost)

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

struct Item {
    int code;
    string name;
    int quantity;
    float cost;
};

int main() {
    int n;
    cout << "enter no. of items: ";
    cin >> n;

    vector < item > Items(n);
    for (int i = 0; i < n; i++) {
        cout << " enter code, name, quantity, cos
        cin >> items[i].code >> items[
>> items[i].quantity >> items[i].cost
}
sort (items.begin(), items.end(), [](
Item b) {
    return a.name < b.name
```

```
};
cout << "\n Sorted list: \n";
for (auto i : items) {
    if (i.name == key) {
        cout << "found " << i.code << " " << i.name
<< " " << i.quantity << " " << i.cost <<
        endl;
    return 0;
}
}
cout << "Item not found! \n";
}
```

Output: Sorted list:

103 Notebook 20 45
101 Pen 10 5 5
102 Pencil 15 3
Found : 102 Pencil 15 3

12/11

Exp - 11 (without iterator)

d.
```
#include <iostream>
#include <vector>
using namespace std;

int main (){
vector<int> v = { 1,2,3,4,5,6,7,8,9,10};

cout << " Initial vector: " << endl;
vector<int>::iterator i;

for( i = v.begin() ; i != v.end() ; i++)
    cout << *I << " " << endl;
}

cout << " Multiply by 10 " << endl;
for( i = v.begin() ; i != v.end() ; i++)
    cout << *I << " " << endl;
}

return 0;
}
```