Experiment 1 :-

1) Write a c++ program to declare a class student having data members as name, Roll no, Accept and Display data for one student.

→
```
#include <iostream>
using namespace std;
class student
{
  int Roll_no;
  String name;
  void Public:
    void accept()
    {
    cout << " Enter the name and
            roll no of a student ";
     cin >> name >> roll_no;
    }
    void display()
    {
    Cout << " End The name of the student:
                                  << name;
    Cout << " The roll_no. of the student:
                                  << roll_no;
    }
};
```

```
int main()
{
    Student s1;
    s1.accept();
    s1.display();
    return 0;
}
```

→ Output :-

```
Enter the name and roll no of Student
DishaAgarwal
60
The name of Student: DishaAgarwal
The roll_no of the student : 60.
```

2) Write a c++ program to declare a class Book having data members as id, name, price. Accept data for 2 books and display data and book having greater price.

→

```
#include (iostream)
using namespace std;
class Book
{
  Public:
  int price, id;
   String name;
   Public:
   void accept()
   {
    cout << "Enter the name, id
            and price of the book";
    cin >> name >> price >> id;
   }
   void display()
   {
    cout << "Name of the Book:" << name;
    cout << "Price of the Book:" << price;
    cout << "Id of the Book:" << id;
   }
};
int main()
{
   Book b1, b2;
    b1.accept();
    b2.accept();
```

getline(cin, name);

```
if(b1.price > b2.price)
{
    b1.display();
}
else
{
    b2.display();
}
}
```

→ Output :-

Enter the name, id, price of the book
a 459 1234

Enter the name, id, price of the book
b 500 1245

Name of the Book: b
Price of the Book: 500
Id of the Book: 1245

VS code

3) Write a program to declare a class time having data members as H, M, S. Accept and display data for one object and total time in seconds.

→
```
#include (iostream)
using namespace std;
class Time
{
public:
int H;
int M;
int S;
public:
void accept()
{
cout<<"Enter time in hours, minutes, and seconds";
cin>>H;
cin>>M;
cin>>S;
}
void void Calculation()
{
H=H*3600;
M=M*60;
S=S+H+M;
}
void display()
{
cout<<"Total time in seconds="<<S;
}
```

```
int main()
{
    Time t1;
    t1.accept();
    t1.Calculation();
    t1.display();
    return 0;
}
```

→ Output :-

Enter the time in hours, minutes, seconds : 2 46 37

Total Time in Seconds = 9997

15/7

Experiment 2 :-

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

→ Program :-

```
#include (stdio.h)
using namespace std;
class city
{
  public:
   string name;
    int p;
  void accept()
  {
    cout<<"Enter the name of city:";
    cin>>name;
    cout<<"Enter the population
          of the city:";
     cin>>p;
   }
};
int main()
{
  city c[5];
```

```
for(int i=0; i<5; i++)
{
  c[i].accept();
}
int max=c[0].p;
for(int i=0; i<5; i++)
{
  if(c[i].p>max)
  {
    max=i;
  }
}
cout<<"City population "<<c[max]
                          .name;
return 0;
}
```

→ Output :-

Enter the Name of city : Mumbai
Enter the population of the city : 45
Enter the Name of city : Kolkata
Enter the population of city : 34
Enter the Name of city : Chennai
Enter the population of city : 75
Enter the Name of city : Banglore
Enter the polulation of city : 43
Enter the Name of city : Pune
Enter the Population of city : 98
City population Pune.

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

→ Program :-

→
```
#include <iostream>
using namespace std;
class Account
{
  public:
  int acc_no;
  float balance;
  void accept()
  {
    cout << "Enter the account no.";
    cin >> acc_no;
    cout << "Enter the balance";
    cin >> balance;
  }
};
int main()
{
  Account b[10];
  for (int i=0; i<10; i++)
  {
    b[i].accept();
```

```
cout << " The balances are: " << endl;
for (int i = 0; i < 10; i++)
{
    if (b[i].balance >= 5000)
    {
        float interest = b[i].balance * 0.1;
        b[i].balance = b[i].balance +
                       interest;
        cout << b[i].balance << endl;
    }
}
return 0;
}
```

→ Output :-

```
Enter the account no. 123
Enter the balance 5000
Enter the account no. 234
Enter the balance 6789
Enter the account no. 324
Enter the balance 2314
Enter the account no. 678
Enter the balance 5356
Enter the account no. 345
Enter the balance 7890
Enter the account no. 45
Enter the balance 234
Enter the account no. 56
Enter the balance 1223
Enter the account no. 45.
Enter the balance 3456.
```

Enter the account no. 67.
Enter the balance 2345
Enter the account no. 12
Enter the balance 2334

The balances are:
5500
7467.9.
5891.6.
8679.

3) 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".

→ Program :-.

```
#include<iostream>
using namespace std;
class staff
{
  public:
  string name, post;
  void accept()
  {
    cout<<"Enter the name of the staff";
    cin >> name;
    cout<<"Enter the post of the staff";
    cin >> post;
  }
```

```
void display()
{
    cout<< name <<endl;
}
};
int main()
{
    staff s[5];
    for(int i=0; i<5; i++)
    {
        s[i].accept();
    }
    cout<<"The names of the staff who
        are HOD are:"<<endl;
    for(int i=0; i<5; i++)
    {
        if(s[i].post=="HOD")
        s[i].display();
    }
    return 0;
}
```

29/7/25

→ Output:-

Enter the name of the sta of Riya
Enter the post of the staff HOD
Enter the name of the staff Piya
Enter the post of the staff Accountant
Enter the name of the staff Diya
Enter the post of the staff HOD
Enter the name of the staff Priya
Enter the port of the staff Teacher
Enter the name of the staff Ina
Enter the port of the staff Accountant
The names of the staff who are HOD are
Riya
Diya

Classmate
Date
Page

Experiment 3:-

1) Write a program to declare a class 'book' containing data members as book-title, author-name and price. Accept and display the information for one object using a pointer to that object.

→ Program:-

→
```
#include <iostream>
using namespace std;
class book
{
  public:
  string btitle, atitle;
  int price;
  void accept()
  {
    cout<<"Enter the name of
          the book";
    cin>>btitle;
    cout<<"Enter the name of
          the author";
    cin>>atitle;
    cout<<"Enter the price of the
          book";
    cin>>price;
  }
```

```
void disp()
{
    cout << " The name of the book is " << title << endl;
    cout << " The name of the author is " << atitle << endl;
    cout << " The price of the book is " << price << endl;
  }
};
int main()
{
    book b;
    book *p;
    p = &b;
    p -> accept();
    p -> disp();
    return 0;
  }
```

→ Output :-

Enter the name of the book hanny
Enter the name of the author 'K'
Enter the price of the book 345
The name of the book is hanny
The name of the author is jk
The name of the book is 345

2) WAP to declare a class 'Student' having data members roll_no, and percentage. Using 'this' pointer invoke member functions to accept and display this data for one object of the class.

→ Program :-

→
```
#include <iostream>
using namespace std;
class Student
{
public:
int roll, per;
void accept()
{
cout << "Enter the roll. no.";
cin >> this -> roll;
cout << "Enter the percentage";
cin >> this -> per;
}
void display()
{
this -> accept();
cout << "The roll. no is" <<
this -> roll << endl;
cout << "The percentage is"
<< this -> per << "%" << endl;
}
};
```

```
int main()
{
    student s1;
    s1.display();
    return 0;
}
```

→ Output :-

Enter the roll no. 60
Enter the percentage 96
The roll. no is 60
The percentage is 96%.

the use of

3) Write a program to demonstrate nested class.

→ Program :-

```
→ #include <iostream>
  using namespace std;
  class student
  {
    public:
    string name;
    int roll;
    void accept()
    {
      cout << "Enter the name
              of the student \n";
      cin >> name;
      cout << "Enter the roll no.
              of the student \n";
      cin >> roll;
    }
    void display()
    {
      cout << " The name of the
              student is " << name
              << " \n";
      cout << " The roll no. of
              the student is:
              << roll << " \n";
    }
```

```
class marks
{
  public:
    int m1, m2;
    void accept1()
    {
      cout << "Enter the marks of
              the first subject\n";
      cin >> m1;
      cout << "Enter the marks of
              the second subject\n";
      cin >> m2;
    }
    void display1()
    {
      float per = ((m1+m2)/200.0)
                  *100;
      cout << "The percentage of
              2 subjects is: "
           << per << endl;
           << "%"
    }
};
int main()
{
  student s;
  student::marks m;
  s.accept();
  m.accept1();
  s.display();
  m.display1();
  return 0;
}
```

Q
5/8/25

→ Output :-

Enter the name of the student
Disha
Enter the roll. no of the student
60
Enter the marks of the first subject.
98.
Enter the marks of the second subject.
99.
The name of the student is: Disha
The roll. no of the student is: 60
The percentage of 2 subjects is: 98.5%.

Experiment -4 :-

1) WAP to swap 2 numbers from same class using object as function argument. Write swap function as member function.

→
```
#include<iostream>
using namespace std;
class numbers
{
 private:
  int a,b;
 public:
  int temp;
  void accept() {
  cout<<"Enter the first number:\n";
   cin>>a;
  cout<<"Enter the second number\n";
   cin>>b;
  }
   void display() {
   cout<<"After swapping:"<<endl;
   cout<<"First number ="<<a<<endl;
   cout<<"Second number ="<<b;
  }
   void swap (numbers &n) {
        n.temp = n.a;
        n.a = n.b;
        n.b = n.temp;
   }
};
```

```
int main() {
  numbers n;
  n.accept();
  n.swap(n);
  n.display();
  return 0;
}
```

→ Output :-

```
Enter the first number : 9
Enter the second number : 2
After swapping :
First number = 2
Second number = 9
```

2) WAP to swap 2 numbers from same class using concept of friend function.

→

```
#include<iostream>
using namespace std;
class number {
  private:
    int a, b, temp;
  public:
    void accept()
    {
      cout<<"Enter the first
            number : ";
      cin>>a;
      cout<<"Enter the second
            number : ";
      cin>>b;
    }
```

```
void display()
{
cout<<"\n\n After swapping: "<<endl;
cout<<" First number = "<<a<<
cout<<" Second number = "<<b<<endl;
                                endl;
}
friend void swap(number &n);
};
void swap(number &n)
{
    n.temp = n.a;
    n.a = n.b;
    n.b = n.temp;
}
int main(){
number n;
n.accept();
swap(n);
n.display();
return 0;
}
```

→ Output :-

Enter the first number : 7

Enter the second number : 4

After swapping :

First number = 4

Second number = 7

3) WAP to swap 2 numbers from different class using friend function.

→
```
#include<iostream>
using namespace std;
class num2;
class num1{
    private:
        int a;
    public:
    void accept(){
    cout<<"Enter the first number:";
    cin>>a;
    }
    friend void swap(num1&,num2&);
};
class num2
{
    private:
    int b;
    public:
    void accept()
    {
    cout<<"Enter the second number:";
    cin>>b;
    }
    friend void swap(num1&,num2&);
};
void swap(num1&x,num2&y)
{
```

```
int temp;
temp = x.a;
  x.a = y.b;
  y.b = temp;
cout<<"\n After swapping :"<<endl;
cout<<"First number ="<<x.a<<endl;
cout<<"Second number ="<<y.b<<endl;
}
int main()
{
  num1 n1;
  num2 n2;
  n1.accept();
  n2.accept();
  swap(n1,n2);
  return 0;
}
```

→ Output :-

Enter the first number = 5

Enter the second number = 6

After swapping :

First number = 6

Second number = 5

4) WAP to create 2 classes Result1 and Result2 which stores the marks of the students. Read the value of marks for both the class objects and compute the average of 2 results.

→
```
#include<iostream>
using namespace std;
class result1
{
  private:
  string name;
  float marks;
  public:
  void accept()
  {
  cout<<"Enter Student name:";
  getline(cin,name);
  cout<<"Enter the total marks
```

```
obtained in first semester out
    of 100:";
cin >> marks;
};
friend void average(result1,
                    result2);
};
class result2
{
    private:
    float marks;
    public:
    void accept().
    {
    cout<<"Enter the total marks
        obtained in second
        semester out of 100:";
        cin >> marks;
    }
    friend void average
                (result1,
                 result2);
};
void average (result1 x, result2 y)
{
    float avg;
    avg = (x.marks + y.marks)/2;
    cout<<"\n Average of both the
        result = "<<avg;
}
int main()
{
```

```
result 1 r1;
result2 r2;
 r1.accept();
 r2.accept();
 average (r1,r2);
  return 0;
 }
```

→ Output :-
Enter student name : Disha Agarwal
Enter the total marks obtained in first semester out of 100 : 98
Enter the total marks obtained in second semester out of 100 : 96
Average of both the results = 97

5) WAP to find the greatest number among 2 numbers from 2 different classes using friend function.

→
```
#include (iostream)
using namespace std;
class num2;
 class num1
 {
  private:
   int a;
   public:
   void accept()
   {
   cout<<("Enter the first no.");
  cin >>a;
```

```
friend void grt(num1, num2);
};
class num2
{
  private:
    int b;
  public:
    void accept()
    {
      cout << "Enter the second number:";
      cin >> b;
    }
    friend void grt(num1, num2);
};
void grt(num1 x, num2 y)
{
  if (x.a > y.b)
  {
    cout << x.a << " is greater than "
         << y.b;
  }
  else
  cout << y.b << " is greater than "
       << x.a;
}
int main()
{
  num1 n1;
  num2 n2;
  n1.accept();
```

```
n2.accept();
grt(n1, n2);
return 0;
}
```

→ output :-

Enter the first number : 8
Enter the second number : 10
10 is greater than 8.

6) Create 2 classes, class A and class B, each with a private. Write a friend function sum() that can access private data from both classes and return the sum.

→

```
#include <iostream>
using namespace std;
class B;
class A {
  private:
    int a;
  public:
    void accept()
    {
    cout<<"Enter the first number:";
    cin>>a;
    }
    friend void sum(A, B);
};
class B {
}
```

```
private:
  int b;
  public:
  void accept()
  {
    cout << "Enter the second number:";
    cin >> b;
    }
    friend void sum(A, B);
};
  void sum(A x, B y)
  {
    int sum;
    sum = x.a + y.b;
    cout << "\n sum of the 2
                numbers = " << sum;
  }
  int main()
  {
      A n1;
      B n2;
      n1.accept();
      n2.accept();
      sum(n1, n2);
    }
```

→ Output:-

Enter the first number: 5
Enter the second number: 6
Sum of the 2 numbers = 11.

4) WAP with a class Number that contains a private integer. Use a friend function swap(Number &, Number &) to swap the private values of 2 Number objects.

```
#include <iostream>
using namespace std;
class Number
{
  private:
  int a,b;
   public:
  void accept()
  {
   cout<<"Enter the first no.:";
   cin>>a;
   }
  void accept1()
  {
   cout<<"Enter the second no.:";
   cin>>b;
  }
   void display()
  {
   cout<<"\nAfter swapping"<<endl;
  cout<<"First number ="<<a<<endl;
 }
 void display()
 {
 cout<<"Second number ="<<b;
```

7) WAP with a class Number that contains a private integer. Use a friend function swap Number (Number &a, Number &b) to swap the private values of 2 Number objects.

→
```
#include <iostream>
using namespace std;
class Number
{
  private:
  int a, b;
  public:
  void accept()
  {
   cout<<"Enter the first no.:";
   cin>>a;
  }
  void accept1()
  {
   cout<<"Enter the second no.:";
   cin>>b;
  }
  void display()
  {
   cout<<"\n After swapping:"<<endl;
   cout<<"First number="<<a<<endl;
  }
  void display()
  {
   cout<<"Second number="<<b;
  }
```

```
friend void swap (Number &x,
                  Number &y);
};
void swap (Number &x, Number &y)
{
    int temp;
    temp = x.a;
    x.a = y.b;
    y.b = temp;
}
int main()
{
    Number n1;
    Number n2;
    n1.accept();
    n2.accept();
    n1.disp
    swap (n1, n2);
    n1.display();
    n2.display();
    return 0;
}
```

→ Output :-

Enter the first number : 7.
Enter the second number : 4.
After swapping :
First number = 4.
Second number = 7.

8) Define 2 classes Box and Cube, each having a private volume, Write a friend function findGreater(Box, Cube) that determines which object has a larger volume

->
```
#include<iostream>
using namespace std;
class Cube
{
  private:
   float volume;
   public:
   void accept()
   {
     cout<<"Enter the volume of
          a cube:";
      cin>>volume;
   }friend void findGrt(cube, Box);
};
class Box
{
  private:
    float vol;
    public:
    void accept()
    {
    cout<<"Enter the volume of
         a box:";
    cin>>vol;
    }
```

```
friend void findgrt(cube, Box);
};
void findgrt(cube x, Box y)
{
  if(x.volume > y.vol)
  {cout<<"\n Cube has
          longer volume";
  } else {
  cout<<"\n Box has longer
      volume";
  }
}
int main()
{
  cube c;
  Box b;
  c.accept();
  b.accept();
  findgrt(c,b);
  return 0;
}
```

→ Output :-

```
Enter the volume of a cube : 567.88
Enter the volume of a Box : 985.66
Box has longer volume
```

9) Create a class complex with real and imaginary parts as private members. Use a friend function to add 2 complex numbers and return the result as a new complex object.

→
```
#include<iostream>
using namespace std;
class Complex
{
  private:
   int r,i;
  public:
  void accept()
  {
   cout<<"Enter the real part:";
   cin>>r;
   cout<<"Enter the imaginary part:";
   cin>>i;
  }
  void disp()
  {
   cout<<r<<"+"<<i<<"i"<<endl;
  }
  friend complex sum(complex,complex);
};

complex sum(complex x, complex y)
{
```

```
complex temp;
temp.r = x.r + y.r;
temp.i = x.i + y.i;
return temp;
}
int main()
{
Complex c1, c2, c3;
cout<<"Enter the first Complex
                    number:\n";
c1.accept();
cout<<"Enter the second complex
                    number:\n";
c2.accept();
cout<<"\n\n Complex Number 1:
                    \n";
c1.disp();
cout<<"\n Complex Number 2:
                    \n";
c2.disp();
c3 = sum(c1,c2);
cout<<"\n\n sum of the
                    2 complex numbers=";
c3.disp();
return 0;
}
```

→ Output :-

```
Enter the first complex number:
Enter the real part : 5
Enter the imaginary part : 6
```

Enter the second complex numbe
Enter the real part: 8
Enter the imaginary part:

Complex Number 1:
5+6i.
Complex Number 2:
8 +2i.

Sum of the 2 complex numbers =
13 +8i.

10) Create a class Student with private data members: name and three subject marks. Write a friend function calculateAverage (Student) that calculates and displays the average marks.

→ #include (iostream).
using namespace std;
class Student
{
string name;
int math;
int phy;
int chem;
public:
void accept()
{
cout<<"Enter student name: ";
getline (cin, name);

```
cout<<"Enter marks obtained in
        maths:";
cin>>math;
cout<<"Enter marks obtained in
        physics:";
cin>>phy;
cout<<"Enter marks obtained in
        chemistry:";
cin>>chem;
}
friend void calculateAverage
                (Student);
};
void calculateAverage(Student x).
{
    float avg;
    avg = (x.math+x.phy+
            x.chem)/3;
    cout<<"\n Average marks="
                <<avg;
}
int main()
{
Student s;
s.accept();
calculateAverage(s);
return 0;
}
```

→ Output :-

Enter student name : DishaAgarwal
Enter marks obtained in maths : 98.
Enter marks obtained in physics : 97.
Enter marks obtained in chemistry : 9

Average marks = 95.

11) Create . 3 classes : Alpha, Beta, and gamma, each with a private data member. Write a single friend function that can access all three and print their sum.

→ #include <iostream>
using namespace std;
class Alpha
{
int a;
public :
void accept()
{
cout << " Enter the first number";
cin >> a;
}
friend void sum(Alpha, Beta, Gamma
};

```
class Beta
{
    int b;
    public:
    void accept()
    {
        cout << "Enter the second number: ";
        cin >> b;
    }
    friend void sum(Alpha, Beta, Gamma);
};
class Gamma
{
    int g;
    public:
    void accept()
    {
        cout << "Enter the third number: ";
        cin >> g;
    }
    friend void sum(Alpha, Beta, Gamma);
};
void sum(Alpha x, Beta y, Gamma z)
{
    int sum;
```

```
sum = x.a + y.b + z.g;
cout << "\n sum = " << sum;
}
int main()
{
    Alpha a;
    Beta b;
    Gamma g;
    a.accept();
    b.accept();
    g.accept();
    sum(a, b, g);
    return 0;
}
```

→ Output :-
Enter the first number : 6
Enter the second number : 3
Enter the third number : 9
Sum = 18

12) Create a class point with private members x and y. Write a friend function that calculates and returns the distance between 2 point objects.

→
```
#include <iostream>
#include <math>
using namespace std;
class Point
{
```

Balguru Swami.

```
int x, y;
public:
void accept()
{
cout << "Enter the x co-ordinate:";
cin >> x;
cout << "Enter the y co-ordinate:";
cin >> y;
}
void disp()
{
cout << "(" << x << "," << y
<< ")" << endl;
}
friend double diff(Point,
Point);
};
double diff(Point x, Point y)
{
double temp;
return temp = sqrt(pow
(x.x - y.x), 2) +
pow((x.y - y.y), 2));
}
int main()
{
Point p1, p2;
cout << "For the point P:\n";
p1.accept();
cout << "For the point Q:\n";
p2.accept();
```

```
cout << "P= ";
p1.disp();
cout << "Q";
p2.disp();
cout << "In Difference between
      the 2 points = " << diff
      (p1, p2) << "units.";
```

⇒ Output :-

For the Point P :
Enter the x-co-ordinate : 5
Enter the y co-ordinate : 6
For the Point Q :
Enter the x-co-ordinate : 7
Enter the y-co-ordinate : 8
P = (5, 6)
Q = (7, 8)
Difference between the 2 points = 2.82843 units.

13) Create 2 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:
        float balance;
    public:
    void accept()
    {
    cout << "Enter the balance
                    information:";
        cin >> balance;
    }
    friend class Audit;
};
class Audit
{
    public:
    void disp(BankAccount x)
    {
        cout << "Balance Information:"
                    << x.balance;
    }
};
int main()
{
    BankAccount b;
    Audit a;
    b.accept();
    a.disp(b);
} return 0;
```

→ Output :-

→ Enter the balance information:
Balance information 9867.49.
Balance information = 9867.49.

Experiment 5 :-

1) i) Default Constructor :-

→
```
#include <iostream>
using namespace std;
class sum
{
    int n, s;
    public:
    sum()
    {
        n=10;
    }
    void calculate()
    {
        for(int i=1; i<=n; i++)
        {
            s = s+i;
        }
    }
    void display()
    {
        cout<<"The sum from 1 to n
            is :"<<n<<s;
    }
};
int main()
{
    sum s1;
    s1.calculate();
```

```
s1.display();
return 0;
}
```

→ Output :-

The sum from 1 to n is : 55.

ii) [Parametrized Constructor]

```
#include <iostream>
using namespace std;
class sum
{
    int n, s=0;
    public:
    sum(int num)
    {
        n=num;
    }
    void calculate()
    {
        for(int i=1; i<=n; i++)
        {
            s = s+i;
        }
    }
    void display()
    {
        cout << "The sum from 1 to
                n is : " << s;
    }
};
```

```
int main()
{
    Sum s1(10);
    s1.calculate();
    s1.display();
    return 0;
}
```

→ Output :-

The sum from 1 to n is : 55.

iii) [Copy Constructor]

```
#include<iostream>
using namespace std;
int class sum
{
    int n, s=0;
  public:
   sum(int num)
   {
      n=num;
   }
   sum(sum &s1)
   {
      n=s1.n;
      for(int i=1; i<=n; i++)
      {
         s=s+i;
      }
      cout<<"The sum from 1 to
            n is :"<<s;
};
}
```

```
int main()
{
    sum s1(10);
    sum s2(81);
    return 0;
}
```

→ Output :-

The sum from 1 to n is : 55.

2) i) Default :-

```
#include<iostream>
using namespace std;
class student
{
    string name;
    float per;
    public:
    student()
    {
        name = "Dishan";
        per = 99;
    }
    void display()
    {
        cout<<"The name of the
            student is : "<<name
            <<endl;
```

```
cout<<"The percentage of the student
        is :"<<per<<":%";
    }
};
int main()
{
    student s1;
    s1.display();
    return 0;
}
```

→ Output :-

The name of the student is : Disha
The percentage of the student is : 99%

ii) Parametrized :-

```
#include<iostream>
using namespace std;
class student
{
    string name;
    float per;
    public:
    student(string n, float p)
    {
        name = n;
        per = p;
    }
    void display()
    {
```

```
cout<<"The name of the student
      is : "<<name<<endl;
cout<<"The percentage of the
      student is : "<<per<<"%";
}
};

int main()
{ student s1("Disha", 99);
  s1.display();
  return 0;
}
```

→ Output :-

The name of the student is : Disha
The percentage of the student is : 99%

iii) Copy :-

```
#include<iostream>
using namespace std;
class student.
{
  string name;
  float per;
  public:
  student(string n, float p)
  {
    name = n;
  } per = p;
```

```
Student (Student &s1)
{
    name = s1.name;
    per = s1.per;
}
void display ()
{
    cout << " The name of the
          student is: " << name
          << endl;
    cout << " The percentage of
          the student is: " << per
          << " % " << endl;
}
};
int main()
{
    Student s1("Disha", 99);
    Student s2(s1);
    s2.display();
    return 0;
}
```

-> Output :-

The name of the student is: Disha
The percentage of the student is: 99%

3) #include <iostream>
using namespace std;
class college
{
string name, course;
int roll;
public:
college(string n, int r, string
c = "Computer
Engineering")
{
name = n;
course = c;
roll = r;
}
college(string n1, string c =
"Computer Engineering,
int r1 = 60)
{
name = n1;
course = c;
roll = r1;
}
void display()
{
cout<<("The name of the student is:"
cout<<"The course of the student is:"<<course
<<endl;
cout<<"The Roll no. of Student:"<<roll<<
endl;
};
}

```
int main()
{
    college c1("Devansh", 41),
    c1.display();
    college c2("Disha");
    c2.display();
    return 0;
}
```

→ Output :-

The name of the student is : Devansh
The course of the student is : Computer Engineering
The Roll no. of Student : 41.
The name of the student is : Disha.
The course of the student is : Computer Engineering.
The Roll no. of the student : 60.

4)
```
#include<iostream>
using namespace std;
class rectangle
{
    int l,b;
    public:
    rectangle()
    {
        l=2;
        b=5;
    }
    rectangle(int x)
    { l=x;
```

```
    b=x;
}
rectangle(int x, int y)
{
    l=x;
    b=y;
}
rectangle(rectangle &r3)
{
    l=r3.l;
    b=r3.b;
}
void calculate()
{
cout<<"the area is: "<<(l*b)<<endl;
}
};
int main()
{
rectangle r1;
r1.calculate();
rectangle r2(3);
r2.calculate();
rectangle r3(6,6);
r3.calculate();
rectangle r4(r3);
r4.calculate();
return 0;
}
```

→ Output :-

The area is : 10
The area is : 9
The area is : 36
The area is : 36

Ans
11/11

Experiment 6 :-

1) Single Inheritance

```
#include (iostream)
using namespace std;
class Person
{
   protected:
   string name;
    int age;
  };
class Student : protected Person
  {
    int rollnumber;
     public:
     void accept()
     {
     cout << "Enter the name of
              the Student : ";
       cin >> name;
     cout << "Enter the age of the
              Student : ";
       cin >> age;
     cout << "Enter the roll no. of the
              student : ";
       cin >> rollnumber;

     }
     void display()
     {
```

```
cout<<"The name of the student is:"<<name
                                  <<endl;
cout<<"The age of the student is:"<<
                                  age<<endl;
cout<<"The roll no. of the student
                                  is:"<<rollnumber
                                  <<endl;
    }
};
int main()
{
    student s;
    s.accept();
    s.display();
    return 0;
    }
```

→ Output :-

```
Enter the name of the student : Disha
Enter the age of the student : 17
Enter the roll no. of the student : 60
The name of the student is : Disha
The age of the student is : 17
The roll no. of the student is : 60
```

2) Multiple Inheritance

```
#include<iostream>
using namespace std;
class Academic
{
  protected:
    int marks;
};
class Sports
{
  protected:
    int score;
};
class Result : protected Academic,
               protected Sports
{
  int total_score=0;
  public:
  void accept()
  {
    cout<<"Enter the marks of the
          student:";
    cin>>marks;
    cout<<"Enter the sports
          score of the student:";
    cin>>score;
  }
void calculate()
{
  total_score = marks + score;
```

```
cout<<"The marks of the student is :"<<marks<<endl;
cout<<"The sports score of the student is :"<<score<<endl;
cout<<"The total score of the student is :"<<total_score<<endl;
    }
};
int main()
{
    Result r;
    r.accept();
    r.calculate();
    return 0;
    }
```

→ Output :-

```
Enter the marks of the student : 99
Enter the sports score of the
 student : 95.
The marks of the student is : 99
The sports score of the student is : 95
The total score of the student is : 194
```

3) Multipiel Inheritance :-

```
#include <iostream>
using namespace std;
class Vehicle
{
    protected:
    string brand;
    int model;
};
class Car: protected Vehicle
{
    protected:
    string type;
};
class ElectricCar: protected Car
{
    int batterycapacity;
    public:
    void accept()
    {
    cout<<"Enter the brand of the
                    car:";
    cin>>brand;
    cout<<"Enter the model of the car:";
    cin>>model;
    cout<<"Enter the type of the
                    car:";
    cin>>type;
    cout<<"Enter the battery capacity
                    of the car:";
    cin>>batterycapacity;
```

```
void display()
{
cout<<"\n The brand of the car : "<<brand
                                  <<endl;
cout<<" The model of the car : "<<model<<endl;
cout<<" The type of the car : "<<type<<endl;
cout<<" The battery capacity of the car :"
      <<batterycapacity<<endl;
  }
};
int main()
{
  ElectricCar e;
   e.accept();
   e.display();
    return 0;
   }
```

→ Output :-

```
Enter the brand of the car : BMW
Enter the model of the car : 2008
Enter the type of the car : SUV
Enter the battery capacity of the car : 115

The brand of the car : BMW
The model of the car : 2008
The type of the car : SUV
The battery capacity of the car : 115
```

4) Hierarchial Inheritance :-

→
```
#include<iostream>
using namespace std;
class Employee
{
  protected:
    int empid;
    string name;
};
class manager : protected Employee
{
  string department; → void ac(){
  cout<<"Enter empid"; {
    cin>>empid;
    cout<<"Enter name";
    cin>>name;
    cout<<"Enter Department";
    cin>>department;
};
void display()
{
  cout<<"The empid is :"<<empid<<endl;
  cout<<"The name is :"<<name<<endl;
  cout<<"The department is :"<<department<<endl;
};
class Developer : public Employee
{
  string programminglanguage;
```

```
void accept()
{
cout<<"Enter the programming language :";
cin>>programminglanguage;
}
void display()
{
cout<<"The programming language is :"<<programminglanguage
<<endl;
}
};
int main()
{
~~Developer d;~~
d.Manager m;
m.accept();
m.display();
Developer d;
d.accept();
d.display();
return 0;
}
```

5) #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 acceptEmp()
{
cout << "Enter College 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 id :- " << endl;
cin >> dept_id;
}
void displayEmp()
{
cout << "College " : << name << endl;
cout << "Emp. name : " << emp_name << endl;
cout << "Id : " << id << endl;
cout << "Staff Name : " << name << endl;
cout << "Department Id : " << deptid << endl;
}
};
class student : protected College
{
String stu_name;
int roll;
public :
void accept ()
{
cout << "Enter name : " << stu_name;
cout << "Enter roll no : " << roll;
}
void display ()
{
cout << "The name : " << stu_name << endl;
cout << "The roll no : " << roll << endl;
};
```

```
int main()
{
    staff s;
    s.accept();
    s.display();
    student s1;
    s1.accept();
    s1.display();
    return 0;
}
```

6) Virtual Base :-

→
```
#include <iostream>
using namespace std;
class college student
{
    protected:
        int student_id;
        string ccode;
    public:
        void accept()
        {
            cout << "Enter Student id";
            cin >> student_id;
            cout << "Enter college code";
            cin >> ccode;
        }
        void display()
        {
            cout << "Student Id: " << student_id
```

```
        cout << "College Code : " << code << endl;
    }
};

class Test : virtual public College
                    Student
{
    protected :
        float percentage;
    public :
        void accept()
        {
            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()
    {
      College Student :: 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 << endl;
      cout << "Total Marks:" << tot_marks
           << endl;
};
```

```
int main()
{
    Result r;
    cout<<"\n\n\t\t Enter Student Details \n\n";
    r.accept();
    cout<<"\n\n\t\t Student Result \n\n";
    r.display();
    return 0;
}
```

Q1

IIIII

Experiment 7 :-

1)
```
#include <iostream>
using namespace std;
class Area
{
    int l, b;
    int s;
    public:
    void area(int l1, int b1)
    {
        int a = l1 * b1;
    }
    void area (int s1)
    {
        int a = s1 * s1;
    }
    void display()
    {
        cout << " The are of the
                laboratory is : " << a;
        cout << " The are of the
                classroom is : " << a;
    }
};
int main()
{
    Area a1;
    a1.area(6,6);
    a1.area(12);
    a1.display();
    return 0;
}
```

2)
```
#include <iostream>
using namespace std;
class sum
{
  public:
    int total (int a[], int n)
    {
      int s=0;
      for(int i=0; i<n; i++)
      {
        s1= s+a[i];
        return s;
      }
    float total (float a[], int n)
    {
      float s=0;
      for(int i=0; i<n; i++)
        s = s+a[i];
        return s;
      };
int main()
{
    sum s;
    int marks[10]={45, 56, 67,
                   78, 89, 90,
                   76, 98, 82, 88};
    float grades[5]={9.2, 8.7, 9.5,
                     8.9, 9.0};
cout << "Sum of 10 students marks:"
     << s.total(marks, 10) << endl;
```

```
cout << "Sum of 5 student grade
        points : " << s.total(grades,
                              5)
     << endl;

    return 0;
}
```

3)
```
#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)
```
#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 increament:" << endl;
    s1.display();
    ++s1;
    cout << "After pre-increment:" << endl;
    s1.display();
    s1++;
    cout << "After post-increment:" << endl;
    s1.display();
    return 0;
}
```

Experiment - 8 :-

```
→ #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();
    return 0;
}
```

2) → #include <iostream>
#include <string>
using namespace std;

class I_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 << name
<< endl;
cout << "Password : " << password
<< endl;
}
};
class Emaillogin : public I_login
{
string email;
public:
void accept() override
{
cout << "\n ... Email login
Details ... " << endl;
cout << "Email Id :" << email << endl;

```
    Ilogin::display();
  }
};

class MembershipLogin : public Ilogin
{
    string memberID;
  public:
    void accept() override {
        cout << "Enter Membership ID: ";
        cin >> memberID;
        Ilogin::accept();
    }
    void display() override
    {
        cout << "\n... Membership login details ..."
             << endl;
        cout << "Membership ID: " <<
             memberID << endl;
        Ilogin::display();
    }
};
int main()
{
    Ilogin * login;
    Emaillogin e;
    Membershiplogin m;
    login = &e;
    login->accept();
    login->display();
```

```
login = &m;
login -> accept();
login -> display();
return 0;
}
```

Q2

```
login = &m;
login->accept();
login->display();
return 0;
}
```

Experiment-9 :-

1) →
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
    fstream new_file;
    fstream first_file;
    new_file.open("new_file", ios::in);
    first_file.open("first_file", ios::out);
    if(!new_file)
    cout<<"File creation failed";
    else if(!first_file)
    cout<<"File Creation failed";
     else
     {
     cout<<"Both files are created";
     string line;
       while(getline(new_file,line))
        first_file<<line<<endl;
        new_file.close();
         first_file.close();
        }
        return 0;
    }
```

2) → #include <iostream>
#include <fstream> → #include <cctype>
using namespace std;
int main()
{
ifstream file ("First.txt");
if (!file)
{
cout << "Error opening File" << endl;
}
char ch;
int digits = 0, spaces = 0;
while (file.get(ch))
{
if (isdigit(ch))
digits ++;
else if (isspace(ch))
spaces ++;
}
cout << "Digits: " << digits << endl;
cout << "Spaces: " << spaces << endl;
file.close();
return 0;
}

3)
```
#include (iostream)
#include (fstream)
#include (string)
using namespace std;
int main()
{
    ifstream file ("First.txt");
    if (!file)
    {
        cout << "Error" << endl;
    }
    string word;
    int count = 0;
    while (file >> word)
    count++;
    cout << "total words: " << count << endl;
    file.close();
    return 0;
}
```

4)
```
#include (iostream)
#include (fstream)
#include (string)
using namespace std;
int main()
{
    ifstream file ("First.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
     << "\n" : " << count << endl;
file.close();
return 0;
}
```

Experiment-10 :-

1) →
```
#include <iostream>
using namespace std;
template <class T> {
T sumArray (T arr[], int n)
{
    T sum = 0;
    for (int i=0; i<n; i++)
    {
        sum += sum + arr[i];
    }
    return sum;
    }
int main()
{
    int int_arr[5] = { 1, 2, 3, 4, 5 };
    float flo_arr[5] = { 1.1, 2.2, 3.3,
                         4.4, 5.5 };
    double dec_arr[5] = { 0.5, 1.5,
                          2.5, 3.5, 4.5 };
    cout << "Sum of integer array:";
        << sumArray (int_arr, 5)
        << endl;
    cout << " Sum of float array:"
        << sumArray (flo_arr, 5)
        << endl;
    cout << " Sum of Double array : "
    << sumArray (dec_arr, 5) <<
                                endl;
    return 0;
}
```

2) → #include <iostream>
using namespace std;
template <class T> void fun(T a)
{
T m = 0;
m = a * a;
cout << "The square of a integer is:"
<< m << endl;
}
template <> void fun(string str)
{
string m;
m = str + str;
cout << "The concatinated string is:"
<< m << endl;
}
int main()
{
int n;
cout << "Enter a value of n";
cin >> n; string s;
cout << "Enter a value of s";
cin >> s;
fun(n);
fun(s);
return 0;
}

3) →
```
#include <iostream>
#include <math.h>
using namespace std;
template <class T1, class T2>
class Calc
{
public:
  T1 x;
  T2 y;
Calc (T1 n, T2 m)
{
   x = n;
   y = m;
}
void sum()
{
cout << "Sum: " << x+y << endl;
}
void diff()   (sum diff)
{
cout << "Difference : " << x-y
  << endl;
}
void pro ()
{
  cout << "Product : " << x*y <<
    endl;
}
void div ()
{
  cout << "Quotient : " << x/y <<
                          endl;
}
```

```
void rem()
{ cout << "Remainder: " << x%y << endl;
}
void power()
{ cout << "x raised to y is: " << pow(x,y)
        << endl;
}
void min_num()
{ cout << "Min of Numbers: ";
  << fmin(x,y) << endl;
}
void max_num()
{ cout << "Max of Numbers: ";
  << fmax(x,y) << endl;
}
void sin_x()
{ cout << "Sin of x: " << sin(x)
    << endl;
}
void cos_y()
{ cout << "Cos of y: " << cos(y)
    << endl;
}
};
int main()
{
```

```
Calc c(int, int) n(100,200);
n.sum();
n.diff();
n.pro();
n.div();
n.rem();
n.power();
n.mix_num();
n.max_num();
n.sin_x();
n.cos_y();
return 0;
}
```

Qn

IIII

classmate
Date
Page

Experiment - 11 :-

1) →

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> v(10);
    int i;
    cout << "size = " << v.size() << endl;
    for(int i=0; i<10; i++)
    {
        v[i] = i+i;
    }
    for(int i=0; i<v.size(); i++)
    {
        v[i] = v[i]+1;
    }
    cout << "the modified values
                of the elements are: " <<
                endl;
    for(int i=0; i<v.size(); i++)
    {
        cout << v[i] << " ";
    }
    cout << "\n\n";
    for(int i=0; i<v.size(); i++)
    {
        v[i] = v[i] * 5;
    }
    cout << "the multiplication with
        a scalar element is: " << endl;
```

```
for(int i=0; i<v.size(); i++)
{
    cout<<v[i]<<" ";
}
cout<<"\n\n";
cout<<"The output in the
given format is :"<<endl;
for(int i=0;i<v.size();i++)
{
    cout<<v[i]<<" , ";
}
return 0;
}
```

2)
```
#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;
for(vector<int>::iterator it=
    v.begin(); it!=v.
                   end().
    ;++it)

  {
   cout<<*it<<" ";
  }

cout<<endl;
cout<<"Multiply by 10"<<endl;
for(vector<int>::iterator it=v.begin();
   it!=v.end();++it)
  {
   *it=(*it)*10;
  }
cout<<"New vector:"<<endl;
for(vector<int>::iterator it=
   v.begin(); it!=v.end();++it)
  {
   cout<<*it<<" ";
  }
cout<<endl;
return 0;
}
```

Experiment-12:-

1)->
```
#include <iostream>
#include <stack>
using namespace std;
stack<int> stack1;
void display()
{
  stack<int> temp = stack1;
  while(!temp.empty())
  {
    cout<<(temp.top())<<" ";
    temp.pop();
  }
  cout<<endl;
}
int main()
{
  int n;
  cout<<"Enter a no.: "<<endl;
  cin>>n;
  for(int i=0;i<n;i++)
  {
    int t;
    cout<<"Enter element at
          position "<<(i+1)<<endl;
    cin>>t;
    stack1.push(temp);
  }
  cout<<endl;
  cout<<"Top most element:"
      <<stack.top()<<endl;
  cout<<endl;
```

```
cout << "Stack elements (top to bottom)..." << endl;

    cout << endl;
    display();
  cout << endl;
    cout << "Pop Function " << endl;
      stack1.pop();
      display();
      stack1.pop();
       display();
      stack1.pop();
       display(); -> return 0;
      }
```

2) ->
```
#include <iostream>
#include <queue>
using namespace std;
int main()
{
  queue<int> q;
  for (int i = 0; i <= 10; i++)
  {
    q.push(i * 10);
  }
  cout << endl;
  cout << "Front Element " << q.front()
                              << endl;
Cout << "Back element " << q.back() <<
                              endl;

  Cout << endl;
    q.pop();
Cout << "After one pop front = " << q.front()
                              << endl;
```

```
cout << endl;
cout << "Queue elements (front to back): ";
while (!q.empty())
{
    cout << q.front() << " ";
    q.pop();
}
return 0;
}
```