15/7/25

Date

Experiment - 1

1) WAP to declare a class student having data members as name, Roll no. Accept & display data for one student.

→

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

class student
{
int roll_no;
string name;
Public

void accept()
{
cout << "enter student name & roll_no:";
cin >> name >> roll-no;
}
void disp()
{
cout << "name of student:" << name;
cout << "Roll no. of student:" << roll no;
};
}; int main()
{
student s1;
s1.accept();
s1.disp();

return 0;
}
```

Output :

```
int main()
{
  student s1;
  s1.accept();
  s1.disp();

  return 0;
}
```

- Output :

```
Enter student name:   Aseeda
Enter roll no :   32

Student details:
   Name:  Aseeda
   Roll no : 32.
```

2) Write a program 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 {
Public:
    int id;
    string name;
    float price;
```

```
void input (){
 cout<<"enter book id :";
 cin >> id;

 cin.ignore();

 cout << "enter book name:";
```

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

 class book
{
 public:
    int bp, bpgs;
string n;
 void accept()
 {
   cout <<"enter name & price:";
   cin >> n >>bp;
   cout << enter pages:";
   cin>>bpgs;
 }
 void display()
{
   cout <<"book name:" <<n <<endl;
   cout <<" book price:" <<bp<<endl;
   cout <<"book pages:" << bpgs <<endl;
}
};
```

```
int main ()
{
  book b1, b2;
  b1.accept();
  b2.accept();

  if (b1.bp > b2.bp)
  {
      cout << "\n book with higher price:\n";
      b1.display();
  }
    else
  {
    cout << "\n book with higher price:\n";
    b2.display ();
  }
    return 0;
}
```

- Output:

```
Enter name & price:  ABC  & 50
Enter pages        : 200
Enter name & price : XYZ  & 100
Enter pages        : 300

Book with higher price:

Book name = XYZ
Book price = 100
Book pages = 300.
```

3) WAP to declare a class time having data members as H, m & S. Accept data for one object & display total time in seconds.

→
```
#inclued<iostream>
#inclued

#include <iostream>
#include <string>

using namespace std;
class time
{
  int h, m, s, t;
public:
  void accept()
{
  cout << "enter hours:";
  cin >> h;
  cout << "enter minutes:";
  cin >> m;
  cout << "enter seconds:";
  cin >> s;
 }
  void display()
 {
  t = (h * 3600) + (m * 60) + s
   cout << "total time:" << t << endl;
 }
};
```

```
int main()
{
  time t1;
  t1.accept();
  t1.display();

  return 0;
}
```

- output:

```
Enter hours: 4
Enter minutes: 6
Enter seconds: 89

total time= 14849.
```

Experiment - 2

1 WAP to declare class 'city' having data members as ~~roll-no~~ & name ~~& class~~ & 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 population;
 void accept(){
 cout<< "enter city name:";
 cin>> name;
 cout << "enter population:";
 cin >> population;
 }
  void display(){
 cout<<"city:"<<name <<"population:"<<population <<endl;
}
};
 int main(){
 city cities[5];
for(int i=0; i<5; i++){
 cout <<"\n enter information of cities:"<<i+1<<"\n";
cities[i].accept();
}
int max=0;
for(int i=1; i<5; i++){
if(cities[i].population> cities[max].population){
 max=i;
```

```
}
}
cout<< "\n city with highest population :\n";
cities [max].display();
   return 0;
}
```

23 ▪ output :

· Enter information of cities 1 :

· enter city name: sde
enter population: 21

· Enter information of cities 2:
enter city name: gdf
enter population: 45.

· Enter information of cities 3:
enter city name: iyhg
enter population: 68

· Enter information of cities 4:
enter city name: hgf
enter population: 67

· enter information of cities 5:
enter city name: gdser
enter population: 88.

· city with highest population:
city: gdser
population: 88.

2) Write a program to declare a class 'account' having data members as account no. & balance. Accept this data for 10 account & give interest of 10% where balance is = or greater than 5000 & display them.

→

```
#include <iostream>
using namespace std;
class account {
int accno;
float balance;
Public:
void accept() {
cout<<"enter account number:";
cin >> accno;
cout<<" enter balance:";
cin >> balance;
}
void addinterest (){
if (balance >=5000) {
balance += balance * 0.10;
}
}
void display (){
if (balance >=5000){
cout<< "account no. :"<< accno<< "balance with
interest:" << balance << endl;
}
}
};
int main() {
```

```
account acc [10];
for (int i=0; i<10; i++){
cout << "\n enter details for account" << i+1 << ":\n";
acc[i].accept();
acc [i].disaddinterest();
}
cout << "\n accounts with balance >= 5000 after adding
  interest:\n";
for (int i=0; i<10; i++){
   acc [i].display();
}
  return 0;
}
```

- Output:
  - enter balai details for account 1:
    enter account number: 12
    enter balance: 345
  - enter details for account 2:
    enter account number: 56
    enter balance: 876
  - enter details for account 3:
    enter account number: 45
    enter balance: 897
  - enter details for account 4: 334
    enter account number: 334
    enter balance: 00897
  - enter details for account 5:
    enter account number: 55
    enter balance: 678

3)

=>

- enter details for account 6:
  enter account number: 666
  enter balance: 7788

- enter details for account 7:
  enter account number: 78
  enter balance: 8888

- enter details for account 8:
  enter account number: 45
  enter balance: 7890

- enter details for account 9:
  enter account number: 33
  enter balance: 5644

- enter details for account 10:
  enter account number: 33
  enter balance: 67865

- Accounts with balance >=5000 after adding interest:

account no: 666 balance with interest: 8566.8
account no: 78 balance with interest: 9777.8
account no: 45 balance with interest: 8679
account no: 33 balance with interest: 6208.4
account no: 33 balance with interest: 74851.5

Page No.
Date

3) Write a program to declare a class 'staff' having data members as name & post. Accept this data for 5 staff & display names of staff who are 'HOD'.

=>

```
#include <iostream>
#include <string>
using namespace std;
class staff{
string name;
string post;
public:
void accept(){
cout<<"enter name:";
cin>>post;
cout<<"enter post:";
cin>>post;
}
void display(){
if(post == 'HOD"){
cout<<"name:"<<name<<endl;
}
}
};
int main(){
staff s[5];
for(int i=0; i<5; i++){
cout<<"\n enter details for staff"<<i+1<<":\n";
s[i].accept();
}
cout<<"\n staff with post HOD:\n";
for(int i=0; i<5; i++){
```

```
    s[i].display();
  }
  return 0;
}
```

```
Enter the details for staff 1
        Enter staff Name: Juam
        Eenter staff Post : HOD

Enter details for staff 2
        Enter staff name : Maria
        Enter staff Post : Head

Enter the details for staff 3
        Enter staff name : Jane
        Enter staff Post : HOD

Enter details for staff 4
        Enter staff name : Mark
        Enter staff Post : HR

Enter details for staff 5
        Enter staff name: JHB
        Enter staff Post : Head

Staff who are HOD:
        Name of staff : Juan
        Name of staff : Jane
```

29/7/25

Experiment - 3

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

→
```
#include <iostream>
using namespace std;
class book
{
  public:
      string book-title;
      int price;
      string author_name;

  void accept ()
  {
    cout << "enter book name:";
    cin >> book_title;
    cout << "enter the name of author:";
    cin >> author_name;
    cout << "price of the book:";
    cin >> price;
  }
 void display ()
 {
   cout << "book name is:" << book_title;
   cout << "\n Author name:" << author_name;
   cout << "\n Price is:" << price;
 }
};
```

```
int main()
{
    book bl;
    book *p;
    bl.accept();
    bl.display();

    return 0;
}
```

book bl;
book *bl;
bl → accept();
bl → display();

- Output :

```
Enter book name    : Python
Enter name of author: Rusk
Price of the book : 999
  Book name is    : Python
  Author name is  : Rusk
    Price is      : 999.
```

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

→
```
#include <iostream>
using namespace std;
class student
{
  int roll_no;
  float percentage;
  public:
```

```
void accept()
{
cout<<"enter studen roll-no:";
cin>>this->roll_no;
cout<<"enter student percentage :";
cin>>this->percentage;
}
  void display()
{
cout<<"Roll number of student is:"<<roll_no;
cout<<"\n percentage is:"<<percentage;
}
};
 int main()
{
   student s1;
   student *p;
   p=&s1;
 s1.accept();
 s1.display();

   return 0;
}
```

■ Output :

```
Enter the student roll_number : 39
Enter the student percentage : 7.93
Roll number of studen is = 39
 percentage is = 7.93
```

3) WAP to demonstrate use of nested class.

->
```
#include<iostream>
using namespace std;
class number
{
 private:
    int num;
    class operations
    {
    public:
     int square(int n)
     {
      return n*n;
     }
     int cube(int n)
     {
      return n*n*n;
     }
};
    public:
   void accept()
{
  cout<<" enter an number:";
    cin>>num;
}
 void display()
{
 operations op;
```

```
cout<<" square of :"<<" ="<< op.square(num)<<endl;
cout<<" cube of :"<<"<<" ="<< op.cube(num)<<endl;
}
};
 int main()
{
 Number n1;
 n1.accept();
 n1.display();
 return 0;
}
```

- Output:

```
Enter an number : 2
 square of : 4
 cube of : 8
```

29/7/25

EXPERIMENT - 4

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

→

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

class number {
    int a, b;
Public:
        void input() {
        cout << "enter two numbers:";
        cin >> a >>b;
        }
        void display() {
        cout << "a:" << a<< "b:" <<b << endl;
        }

        void swapnumbers (Number & obj) {
        int temp = obj.a ;
        obj.a = obj.b ;
        obj.b = temp ;
        }
};
int main() {
    number n1;
    n1. input();
    cout << "Before swapping:";
    n1.display();
    n1. swapnumbers (n1);
    cout << "After swapping:";
```

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

- Output :

Enter two numbers : 2 4

Before swapping : a = 2, b = 4
After swapping : a = 4, b = 2.

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

→
```
#include <iostream>
using namespace std;
class Number {
  int a, b;
Public:
  void input() {
   cout << "enter two numbers :";
   cin >> a >> b;
  }
  void display() {
   cout << " a:" << a << "b:" << b << endl;
  }
friend void swapnumbers (number & obj);
};
void swapnumbers (Number & obj) {
 int temp = obj.a;
```

```
    obj.a = obj.b;
    obj.b = temp;
}
int main() {
    Number n1;
    n1.input();
    cout << "before swapping:";
    n1.display();
    swapnumbers(n1);
    cout << "After swapping:";
    n1.display();
    return 0;
}
```

- Output :

Enter two numbers : 1 3

Before swapping :

a : 1 , b : 3

After swapping :

a : 3 , b : 1

3 WAP to swap two numbers from different class using friend function

->
```
# include <iostream>
using namespace std;
class class1;
class class2 {
int b;
Public:
    void input() {
    cout<<" enter number for class 2:";
     cin>>b;
     }
    void display() {
     cout <<" class b =" <<b<< endl;
    }
    friend void swapnumbers (class1 &x, class &y);
    };
    class class1 {
     int a;
  Public:
    void input(){
    cout<<" enter number for class1:";
    cin>>a;
  }
  void display () {
  cout<<" class1 a =" << a<< end1;
  }
  friend void swapnumbers (class1 &x, class &y);
  };
```

```
void swapnumbers ( class1 &x , class2 &y)
{
   int temp = x.a;
   x.a = y.b;
   y.b = temp;
}
int main() {
   class1 obj1;
   class2 obj2;

   obj1.input();
   obj2.input();

   cout<<" Before swapping:" <<endl;
   obj1.display();
   obj2.display();

   swapnumbers (obj1, obj2);

   cout<<" After swapping:"<<endl;
   obj1.display();
   obj2.display();

   return 0;
}
```

Page No.
Date

- Output:

```
Enter number for class 1 : 5
Enter number for class 2 : 2

Before swapping:
class1 a = 5
class2 b = 2

After swapping :
class1 a = 2
class2 b = 5.
```

4. WAP to create two classes Result1 & Result2 which stores the marks of the students. Read the value of a mars for both class objects & compute the average of two results.

→

```
#include <stdiciostream>
using namespace std;
class Results2;

class Results1 {
    int marks;
Public:
    void input(){
    cout << "enter marks for Results:";
    cin >> marks;
    }
    friend void average (Result r1, Result r2);
};
```

```
Class results2 {
  int marks;
Public:
  Void input () {
    cout<<" enter marks for results 2:";
    cin >> marks;
    }
  friend void average (result r1, result r2);
};

void average (result r1, result r2) {
     float avg = (r1.marks + r2.marks) / 2.0;
     cout<<" Avg marks =" << Avg << endl;
   }
int main() {
   Result1 obj1;
   Result 2 obj2;

   obj1.input();
   obj2.input();

   average (obj1, obj2);
    return 0;
}.
```

- Output :

```
Enter marks for result1 : 80
Enter marks for result 2: 90

Avg marks = 85
```

Page No.
Date

Experiment -5

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

-> • using default constructor:

```
#include <iostream>
using namespace std;
{
 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 :" <<s;
 }
};
int main ()
{
sum s1;
s1. calculate()
s1. display ()
```

```
return 0;
}
```

- Output :

The sum from 1 to n is : 55

- by using parameterized contructor :

```
#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

c) using copy constructor

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

```
};

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

- Output:

the sum from 1 to n is : 55

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

-> • by default constructor :

```
# include <iostream>
using namespace std;
class student{
 string name;
 float percentage;
 Public:

student()
{
 name = " ";
 percentage = 0.0;
}
```

```
void accept()
 {
cout << "enter name:";
cin >> name;
cout << " enter percentage:";
cin >>percentage;
}
void display()
{
cout <<"\n student Name:" <<name;
cout <<"\n Percentage:" << percentage << "%.";
}
};
int main()
{
student s;
s.accept();
s.display();
 return 0;
}
```

- output:

```
Enter name: Aseeda
Enter percentage: 85.6

Student name: Aseeda
 percentage : 85.6%.
```

- Parameterized constructor:

```
#include <iostream>
using namespace std;
class student{
 string name;
 float percentage;

Public :
     student (string n, float p)
  {
     name = n;
     percentage = p;
  }
 void display()
{
cout<<" \n student Name :" << name;
cout<<" \n Percentage  :" << percentage <<"%.";
}
};

int main()
{
 student s ("Aseeda", 85.6);
 s.display();

 return 0;
}
```

- Output:

```
student Name: Aseeda
 Percentage : 85.6%.
```

- Copy constructor:

```
# include <iostream>
using namespace std;
Class student
{
string name;
float Percentage;
Public:

student (string n, float p)
{
 name = n;
 Percentage = p;
}
 student (const student &s)
 {
   name = s.name;
   Percentage = s.Percentage;
 }
 Void display()
 {
 cout << " student Name:" <<name;
 cout << " Percentage:" <<Percentage <<"%.";
}
};
 int main()
{
 student s1 ("Aseeda", 65.6);
 student s2 ( s1);

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

```
return 0;
}
```

- output:

```
student Name : Aseeda
percentage : 85.6%.

student Name : Aseeda
Percentage : 85.6%.
```

c) Define a class 'college' variables as roll-no, name, course. WP using constructor with default value as "computer engineering" for course. Accept this data for 2 objects of class & display the data.

→ • default constructor:

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

class college {
  string name;
  string course;

Public:
  college() {
   course = "computer Engineering:"

void accept()
 {
```

```
cout << " enter student name:";
cin >> name;
}

void display()
{
cout << " Name:" << name;
cout << " course:" << course << endl;
}
};
int main ()
{
college c1, c2;

c1. accept();
c2. accept();

c1. display();
c2. display();
return 0;
}
```

output:

enter student name: Aseeda
enter student name: Tehsin

Name: Aseeda
course: computer Engineering

Name: Tehsin
course: computer Engineering.

- Parameterized

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

class college {
    string name;
    string course;

public:
    college( string n, string c = "computer Engineering")
    {
    name = n;
    course = c;
}
void display()
{
cout << " Name: " << name;
 cout << " course : " << course << endl;
}
};
int main() {
 college c1 ("Aseeda")
 college c2("Tehsin");

 c1.display();
 c2.display();
}
```

Output:

Name: Aseeda
course: computer engineering

Name: tehsin
course: computer engineering.

copy constructor:

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

class college
{
 string name;
 string course;

public:
 college (string n, string c = "computer Engineering")
 {
  name = n;
  course = c;
 }
 college (const college &cl)
 {
  name = cl.name;
  course = cl.course;
 }
 void display()
 {
 cout<<" name :"<<name;
 cout << " course:"<<course<< endl;
 }
};
int main()
{
```

```
college c1("Aseeda");
college c2(c1); (Tehsin) = (c1);

c1.display();
c2.display();

}
```

- Output:

```
Name: Aseeda
course: computer Engineering

Name: Tehsin Aseeda
course; computer engineering
```

d Write a program to demonstrate constructor overloading.

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

class student
{
  string name;
  string course;

Public:
    Student()
    {
      name = " Tehsin";
      course = "computer Engineering";
    }
```

```
student (string n, string c)
{
 name=n;
 course=c;
}

student (const student &s)
{
 name=s.name;
 course = s.course;
}

Void display()
{
cout << "Name:" <<name;
cout<<" course:" << course <<endl;
}
};
int main()
{
student s1;
student s2 ("Aseeda", "IT");
student s3 (s2);

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

- Output :

Name : tehsin
course: computer Engineering

Name : Aseeda
course: IT

Name : Aseeda
course: IT

17/10

Experiment - 6

a) WAP to implement multilevel inheritance. Assume suitable

→

```
#include <iostream>
using namespace std;
class person {
public:
  string name;
void getName() {
    cout<<" enter name:";
    cin>>name;
  }
};
class student: public person {
public:
  int RollNo;
void getRollNo() {
  cout<<" enter roll number:";
  cin>>roll no;
  }
};
class Marks : public student {
  public:
    int marks;
    void getmarks() {
    cout << " enter marks:";
    cin>>marks;
  }
  void display() {
  cout<<" \n Name :" <<name;
  cout<<" \n Roll no:"<<roll No;
  cout<<" \n marks:" <<marks<<" %." <<endl;
  }
```

```
};

int main () {
    marks m;
    m.getName ();
    m.getRollNo ();
    m.getMarks();
    m.disPby();
}
```

- Output :

```
Enter name :  Aseeda
Enter Roll number : 32
Enter marks  : 85

Name  : Aseeda
Roll No  :  32
MarKS :  85%.
```

b) WAP to implement multiple inheritance. Assume suitable data.

→

```
# include <iostream>
using namespace std;
class student {
Public:
  void getstudentInfo ()
{
 cout<<" Name: ";
 String name;
int rollno;
Public:
```

```
void getstudentinfo()
{
 cout << " enter name:";
 cin >> name;
 cout << " enter roll number:";
 cin >> rollno;
 }
};
class marks {
 Protected:
  int m1, m2, m3;
 Public:
  void getmarks()
 {
 cout << " enter marks in 3 subjects:";
 cin >> m1 >> m2 >> m3;
 }
};
class Result : Public student, Public marks {
  Public:
   void displayResults() {
    int total = m1 + m2 + m3;
    float Percentage = total / 3.0;
     cout << "\n --- Result ";
     cout << " name:" << name << endl;
     cout << " RollNo :" << rollno << endl;
     cout << " total marks:" << total << endl;
     cout << "percentage:" << percentage << "%." << endl;
    }
 };
int main() {
 Result r;
```

r. getstudentinfo();
r. get marks();
r. displayResult();
return 0;
}.

- Output:

Enter name: Aseeda
Enter roll number: 101
Enter marks in 3 subjects: 85 90 88

Name : Aseeda
Roll No : 101
total marks: 263
Percentage: 87.667%.

c) by heirarchical inheritance.

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

Class student {
 Public:
 string name;
 int roll;
 void getdata () {
  cout << "enter name and roll no:";
  cin >> name >> roll;
  }
};
```

```
class Marks : Public student {
Public:
  int m1, m2;
  void getMarks() {
  cout << "enter 2 subject marks:";
  cin >> m1 >> m2;
  cout << "total marks =" m1 + m2 << endl;
  }
};
class sports : Public student {
Public :
  int score;
  void getscore() {
  cout << "enter sports score:";
  cin >> score;
  cout << "sports score =" << score << endl;
  }
};
int main () {
 marks m;
 sports s;

 m.getData();
 m.getMarks();

 s.getData();
 s.getMarks();
}
```

Page No.
Date

- Output :

```
Enter name and roll no  :  Aseeda  32
Enter 2 subject marks   :  87  65
Total Marks             :  152
Enter name and roll no  :  abc  56
Enter sports score      :  87
sports score            :  87.
```

d. By hybrid inheritance.

→

```
#include <iostream>
using namespace std;
class student{
 Public:
   int roll;
   void input (){
      cout << " enter Roll NO:";
      cin >> roll;
     }
};
class Marks: Public student{
 Public:
   int m1, m2;
   void getmarks(){
     cout<<" enter two subject marks:";
     cin >> m1 >>m2;
    }
};
class sports {
Public:
   int score;
```

```
void getscore() {
    cout << "enter sports score:";
    cin >> score;
    }
};

class Result : public Marks, public sports {
public:
    void display() {
    cout << "\n Roll No:" << roll;
    cout << "\n total =" << (m1 + m2 + score) << endl;
    }
};
int main() {
 Result r;
 r.input();
 r.getMarks();
 r.getscore();
 r.display();
}
```

- output:

```
Enter roll No          : 5
Enter two subject marks : 78  85
Enter sports score     : 10

Roll no                : 5
total                  : 173
```

e WAP to demonstrate virtual base class. Assume suitable data with figures.

→

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

class person {
 Public:
    string name;
};

class student : virtual Public person {
   Public:
     int roll;
};

class Teacher : virtual Public person {
 Public:
    string subjects;
};

class TA : public student, Public teacher { };
int main() {
  TA ta;
  ta.name = "Aseeda";
  ta.roll = 32;
  ta.subject = "Maths";

  cout << " TA Details: \n";
  cout << "Name:" << ta.name << endl;
  cout << "Roll no :" << ta.roll << endl;
  cout << "subject :" << ta.subject << endl;
}
```

TA details :

Name : Aseeda

Roll NO : 32

Subject : Maths.

17/10

Page No.
Date

Experiment 7.

a. WAP using function overloading to calculate the area of a laboratory & area of classroom.

→
```
#include <iostream>
using namespace std;
class Area {
Public:
   int calculate (int length, int width) {
      return length * length;
   }
int calculate (int side) {
      return side * side;
   }
};
int main () {
   Area a;

cout << " Area of labotory (rectangle 10x20): " << a.
        calculate (10,20) << endl;

cout << " Area of classroom (square 15x15): " << a.
        calculate(15) << endl;

      return 0;
}
```

- Output:

```
Area of laboratory (Rectangle 10x20) : 200
Area of classroom (square 15x15)  : 225
```

b) WAP using function overloading to calculate sum of 5 float values & sum of 10 integer values.

→

```
#include <iostream>
using namespace std;
class calculator {
 int num;
 public:
 calculator (int n=0) { num = n };
 Calculator operator + ()
 {
  return calculator (-num);
 }
 Void showNum () {

  cout << "value:" << num << endl;
 }
 float add (float a, b, c, d, e) {
     return a+b+c+d+e;
 }
int add (int a, b, c, d, e, f, g, h, i, j ) {
     return a+b+c+d+e+f+g+h+i+j ;
   }
};
int main () {
   calculator c1 (10);
   c1. showNum();

   calculator c2 = + c1;
   c2. showNum();

   cout << " sum of 5 floats:" << c1. add(1.1, 2.2, 3.3, 4.4, 5.5) << endl;
   cout << " sum of 10 int :" << c1. add (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) << endl;
```

```
return 0;
}
```

- Output:

```
Value          :  10
value          :  -10
sum of 5 floats :  16.5
sum of 10 int  :  55
```

c) WAP to implement unary -operator when used with the object so that the numeric data members of class is negated.

→
```
# include <iostream>
using namespace std;
class Number {
    int value;
    public:
    Number(int v) {
    value = v;
}
void display() {
 cout << " value : " << value << endl;
 } }
void display (string msg) {
 cout << msg << value << endl;
}
void operator -() {
    value = -value ;
 }
};
```

```
int main()
{
  number n(10);

  n.display();
  n.display("current value:");
  -n;
  n.display("After negation:");
    return 0;
}
```

- Output:

```
value          : 10
current value  : 10
After negation : -10
```

d) WAP to implement unary ++ operator (for pre increment & post increment) when used with the object so that the numeric data members of the class is incremented.

→

```
#include <iostream>
using namespace std;
class Demo{
   int num;
   public:
     Demo(int n=0){
     num=n;
   }
  void show(){
  cout<<"Number:"<<num<<endl;
```

```
void show ( string msg) {
 cout << msg << num << endl ;
 }
Number operator ++ () {
     ++ num;
     return *this;
  }
Number operator ++(int) {
   number temp= *this;
   num++;
    return temp;
  }
};
int main () {
  number n1(5);

  n1.show();
  n1.show("value:");
  ++ n1;
  n1.show("After Pre-increment:");
  n1++;
  n1.show("After Post-increament:");
}
```

- output:

```
Number          : 5
value           : 5
After Pre-increment : 6
After Post-increment : 7
```

10/10

Experiment - 8

* WAP to demonstrate compile time & Run time Polymorphism (operator overloading binary).

a) WAP to overload the '+' operator so that two strings can be concatenated.

→
```
#include <iostream>
#include <string>
using namespace std;
class Mystring {
    string str;
Public:
    Mystring (string s) : str(s) {
    }
Mystring operator + (const Mystring &s) {
    return mystring (string + s.string);
 }
Void display() {
 cout << str << endl;
  }
};
int main() {
 Mystring s1 ("Hello") , s2("world!");
 Mystring s3 = s1 + s2;
 s3. display();
}.
```

■ Output :

Hello , World !

b) WAP to create a base class Login having data members as name & password, declare accept() function virtual. Derive Email login & Membershiplogin classes from login. Display Email login details & membership login details of emp.

→

```
#include <iostream>
using namespace std;
class Number {
  int value;
 Public:
  Number (int v=0) {
    value = v;
  }
  Number operator +(Number n) {
   return Number( value +n. value);
  }
  void display() {
   cout << " value :" << value << endl;
  }
};
class login {
 Protected : string name, Password;
  Public :
     virtual void accept() {
       cin >> name >> Password;
     }
    virtual void accept() {
    }
};
class Emaillogin : Public login {
     string email;
 Public :
```

```
Void accept() {
    login :: accept();
    cin >> email;
}
Void display() {
    cout << " email login: " << name << " " << Password << " " << email
        << endl;
}
};
class membershiplogin : Public login {
    string memId;
Public:
    Void accept() {
        login:: accept();
        cin >> mem ID;
    }
    void display() {
    cout << " membership login: " << name << " " << Password << " "
                                    memId << endl;
    }
};
int main() {
    number n1(10), n2(20), n3;
    n3 = n1 + n2;
    n3.display();
    login ob;
}
```

17/10

■ output:

value : 30

Email login : Alice Pass123 alice@gmail.com

membership login : bob Pass456 M123.

a) WAP

→ #in
#in
usin
int n
if
of

if (
co
}
strin
whi
o
}
cou
in
ou
}

■ output

Experiment - 9

a) WAP to copy contents of one file into another.

→

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
ifstream infile("first.txt");
ofstream outfile("second.txt");

if(!infile){
  cout<<"first.txt not found!"<<endl;
  return 1;
}
string line;
while(getline(infile,line)){
  outfile<<line<<endl;
}
cout<<"context copied succesfully!"<<endl;
infile.close();
outfile.close();
}
```

- output:

contents copied succesfully!

b) WAP to count digits & spaces using file handling.

→
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
  ifstream file("input.txt");
  char ch;
  int digits=0; spaces=0;
 while (file>> noskipws>>ch) {
  if (ch>='0' && ch<='9') digits++;
  if (ch==' ') spaces++;
 }
 cout<<" Digits:" <<digits<<" \nspaces:" << spaces <<endl;
 file.close();
}
```

- output:

```
digits=0
spaces=0
```

c) WAP to count words using file handling.

→
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
ifstream file ("input.txt");
string word;
int count=0;
```

```
while (file >> word)
    count++;
cout << " total words : " << count << endl;
 file.close();
}
```

- Output:

Total Words : 0

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

→

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
 ifstream file("sample.txt");
string word, target;
int count = 0;
cout << " enter word to count:";
cin >> target;
while (file >> word) {
 if (word == target) {
    count++;
 }
 cout << " the word " << target << " occurs " << count << " times".
 file.close();
 return 0;
}
```

Output :

Enter words to count : file
the word file occurs 0 times.

Q2

Experiment 10

Page No.
Date

a) WAP to find sum of Array elements using function template.

→
```
#include <iostream>
using namespace std;
template <typename T>
T arraysum (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};
  double doubleArr[] = {1.5, 2.5, 3.5};

cout << "sum of int array:" << arraysum (intArr, 5) << endl;
cout << "sum of double array:" << arraysum(doubleArr, 3) << endl;
}
```

Output:

```
sum of int array: 15
sum of double array: 7.5
```

b. WAP of square function using template specialization.
- calculate square of integer no. & a string.
(square of string is nothing but concatenation of a string with itself.

- Write a specialized function for the square of a string.

→
```
#include <iostream>
using namespace std;
template <class T>
T square (T n) {
    return n*n;
}
template < >
string square (string s) {
    return s+s;
}
int main() {
    int n;
    string str;
    cout << " enter an integer: ";
    cin >> n;
    cout << " square of integer: " << square (n) << endl;

    cout << " enter a string: ";
    cin >> str;
    cout << " square of strings: " << square (str) << endl;

    return 0;
}
```

Page No.
Date

1. output :

```
Enter an integer :  2
Square of integer :  4
Enter a string :  abc
square of string:  abcabc.
```

c WAP to build simple calculator using class template.
(using 10 operations & switch case in main())

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

template <class T>
class calc {
  T a, b;

Public:
  calc (Tx, Ty) {

      a = x;
      b = y;
   }
Void add() {
   cout<<" sum = " << a+b <<endl;
  }

Void subtract() {

  cout <<" difference = " << a-b <<endl;
  }
```

```
void mul() {
    cout << " Product = " << a*b << endl;
}

void div() {
    cout << " Quotient = " << a/b << endl;
}

void mod() {
    cout << " Remainder = " << (int) a % (int) b << endl;
}

void sinV() {
    cout << " sin(a) = " << sin(a) << ", sin(b) = " << sin(b) << endl;
}

void cosV() {
    cout << " cos(a) = " << cos(a) << ", cos(b) = " << cos(b) << endl;
}

void tanV() {
    cout << " tan(a) = " << tan(a) << ", tan(b) = " << tan(b) << endl;
}

void logV() {
    cout << " log(a) = " << log(a) << ", log(b) = " << log(b) << endl;
}

void root() {
    cout << " root(a) = " << sqrt(a) << ", root(b) = " << sqrt(b)
    << endl;
}
```

```
};

int main () {
    int ch;
    double x, y;
    cout << " enter two numbers: ";
    cin >> x >> y;
    calc <double> c (x, y);

cout << " \n1. add \n2. subtract \n3. multiply \n4.
        divide \n5. modulus \n6. sin \n7. cos \n8. Tan
        \n9. log \n10. Root \n Enter choice: ";

    cin >> ch;

switch (ch) {

 case 1 :
      c.add ();
      break;

 case 2 :
      c.subtract ();
      break;

 case 3:
      c.mul ();
      break;

 case 4 :
      c.div ();
      break;
```

```
case 5:
    c.modC();
    break;

case 6:
    c.sinvC();
    break;

case 7:
    c.cosvC();
    break;

case 8:
    c.tanvC();
    break;

case 9:
    c.logvC();
    break;

case 10:
    c.rootC();
    break;

default:
    cout<<" Invalid choice:";
}
    return 0;
}.
```

Page No.
Date

. Output :

Enter two numbers : 2 4

1. Add
2. substract
3. Multiply
4. Divide
5. Modulus
6. sin
7. cos
8. Tan
9. Log
10. Root

Enter choice : 2
Difference : -2.

d WAP to implement Push & Pop from stack using class template.

→
```
#include <iostream>
using namespace std;
template <class T>
class stack {
    Ts[10];
    int top;
Public:
    stack() {
      top = -1;
    }
```

```
void Push (T x) {
    if (top == 9)
    cout << "overflow\n";
    else s[++top] = x;
    }

void Pop() {
    if (top == -1)
    cout << "underflow\n";
    else
     cout << "Popped: " << s[top--] << endl;
    }

void display() {
    for(int i=top; i>=0; i--)
    cout << s[i] << " ";
    cout << endl;
    }
};
int main() {
    stack<int> st;
    int ch, x;
    do {
     cout << "\n1. Push 2. Pop 3. display 4. exit \n Enter choice:";
     cin >> ch;
     switch (ch) {
        case 1: cout << "enter value:";
                cin >> x;
                st.push(x);
                break;
```

```
case 2:
    st.pop();
    break;

case 3:
    st.display();
    break;
}
}
while (ch!=4);
}
```

- Output :

1. Push 2. Pop 3. display 4. exit

enter choice : 1
enter choice : 10

1. Push 2. Pop 3. display 4. exit.

enter choice : 1
enter choice : 20.

1. Push 2. Pop 3. display 4. exit

enter choice : 3
enter choice : 20 10

Page No.
Date

Experiment 11

* WAP to implement generic vectors. Include following member functions.

- To create the vector
  a) to modify the value of a given element.
  b) to multiply by scalar value.
  c) to display the vector in form (10, 20, 30).

→

```
#include <iostream>
using namespace std;
template <class T>
class vector {
    T a[10];
    int n;
Public:
  Void create(){
    cout<<"enter size:";
    cin>>n;
    cout<<" enter elements:";
    for(int i=0; i<n; i++)
    cin>>a[i];
  }
Void modifyElement (int Pos, T val) {
    if (Pos>=0 && pos<n)
      a[pos]=val;
      else,
        cout<<" invalid Position\n";
  }
Void multiplyscalar (T s){
for(int i=0; i<n; i++)
  a[i]*=s;
}
```

eg (10, 25, 30) X 2
⇒ (20, 50, 60)

```
void display() {
    cout << "(";
    for (int i=0; i<n; i++) {
        cout << a[i];
        if (i<n-1)
        cout << ", ";
    }
cout << ")\n";
}
};
int main() {
  vector <int> v; - create a vector of integers
  int ch, pos, val, s;
  v.create();
  do {
    cout << "\n1.modify element 2.multiply by scalar 3.
          display 4. Exit \n Enter choice:";
    cin >> ch;
    switch (ch) {
      case 1:
        cout << " enter position and new value:";
        cin >> pos >> val;
        v.modifyElement (pos, val);
        break;

      case 2:
        cout << " enter scalar value:";
        cin >> s;
        v.multiplyscalar (s);
        break;

      case 3: v.display();
```

```
        break;
    }
    while (ch != 4);
}
```

- Output:

```
Enter size: 3
enter elements: 10 20 30

1. Modify element  2. Multiply by scalar  3. display  4. exit
  enter choice: 3
    (10, 20, 30)
```

b With Iterators :

```
#include <iostream>
#include <vector>
using namespace std;
template <class T>
class MyVector {
    vector <T> v = {10, 20, 30};
    public:
    void modify (int i, T val)
    {
        v[i] = val;
    }
    void multiply (T s) {
    for (typename vector <T> :: iterator it = v.begin ();
            it != v.end ();
            ++ it)
            *it = (*it) * s;
```

```
}
void display() {
    cout << "(";
    for (typename vector<T>::iterator it = v.begin(); it != v.end();
        ++it) {
            cout << *it;
            if (it != v.end() - 1)
            cout << ", ";
            }
            cout << ")\n";
        }
    };
int main() {
    MyVector<int> v;
    v.display();
    v.modify(1, 50);
    v.display();
    v.multiply(2);
    v.display();
}
```

output :

(10, 20, 30)

(10, 50, 30)

(20, 100, 60)

Page No.
Date

Experiment - 12

* WAP using STL

a) Implement Stack
b) Implement Queue
c) Implement sorting & searching with user-defined records such as Person Record (Name, birth date, telephone no), item Record (item name, code, Quantity & cost)

a)

```
#include <iostream>
#include <stack>
using namespace std;
int main () {
stack <int> s;
s.push(10);
s.push(20);
s.push (30);
cout<<" stack elements (top to bottom):";
while ( !s.empty ()) {
    cout<<s.top() <<" ";
    s.pop();
}
}
```

- Output:

Stack elements (top to bottom): 30 20 10

a)

```
#include <iostream>
#include <queue>
using namespace std;
int main () {
    queue <int> q;
    q.Push (10);
    q.Push (20);
    q.Push (30);
cout << "Queue elements (front to rear): ";
while (!q.empty()) {
    cout << q.front () << " ";
    q.POP();
  }
}
```

* Output:

Queue elements (front to rear): 10 20 30

b)

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
struct person {
  string name;
  string birthDate;
  string Phone;
};
bool compareByName (const person &a, const person &b) {
    return a.name < b.name;
}
int main () {
```

```
vector<Person> people = {{"Alice", "01-01-2000", "111"},
{"bob", "02-02-1999", "222"}, {"charlie", "03-03-2001", "333"}};
sort(people.begin(), people.end(), compareByName);
cout << "sorted list by name:\n";
for (auto &p : people)
    cout << p.name << " " << p.birthDate << " " << p.phone
    << "\n";
    string searchName = "Bob";
    auto it = find_if (people.begin(), people.end(), [&](
    Person &p) { return p.name == searchName;
    }
    if (it != people.end())
    cout << "\nfound: " << it->name << " " << it->birthDate << " "<<it->
    phone << "\n";
else
    cout << "\nperson not found\n";
}
```

- Output:

```
sorted list by name:
Alice 01-01-2000 111
Bob 02-02-1999 222
charlie 03-03-2001 333

Found: Bob 02-02-1999 222
```