PAGE No.
DATE / /

Q) C++ program to add 2 numbers

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;
    cout<<"Enter the two numbers:";
    cin >> a >> b;
    c= a+b;
    cout<<"Sum of two numbers ="<< c;
    return 0;
}
```

Q) C++ program to check whether entered no is even or odd.

```
#include <stdio.h> <iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Enter the number:";
    cin>>n;
    if(n%2==0)
    {
        cout<<"number is even".
    }
else
{
    cout<<"It is odd";
}
    return 0;
}
```

PAGE No.
DATE / /

Q) Write a program to print 1 to 10 numbers using for loop

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1; i<=10; i++)
    {
        cout<<" "<<i;
    }
    return 0;
}
```

Q) Write a program to print nos 1-20 using while loop.

```
#include <iostream>
using namespace std;
int main()
{
    int i = 1
    while (i<=10)
    {
        cout<<" "<<i;
        i++;
    }
    return 0;
}
```

31/7/25

PAGE No. / DATE / /

Experiment – 1

Q1) Write a C program to declare class student having data members as roll no and name accept and display data for one object.

```
#include <iostream>
using namespace std;
Class student
{
  int roll;
  string name;
  public:
  void accept()
  {
    cout << "Enter value of roll number:";
    cin >> roll;
    cout << "Enter the student name:";
    cin >> name;
  }
  void display()
  {
    cout << "Roll number =" << roll;
    cout << "Name =" << name;
  }
};
int main()
{
  Student S1;
  S1.accept();
  S1.display();
  return 0;
}
```

Q2) Write a program to declare a class 'book' having data members as book name, price and no of pages. Accept this data for 2 objects and display the name of book having greater price.

```
#include <iostream>
using namespace std;
class book
{
    String name;
    int price;
    int pages;
  public:
    void accept()
    {
      cout << " Enter book name : ";
      cin >> name;
      cout << " Enter price of book : ";
      cin >> price;
      cout << " Enter no. of pages : ";
      cin >> pages;
    }
    void display()
    {
      cout << " Name of book = " << name;
      cout << " price of book = " << price;
      cout << " pages = " << pages;
    }
};
int main()
{
    book b1, b2;
    b1.accept();
```

```
    b2.accept();
cout << "\n book with greater price = \n";
{ if (b1.price > b2.price)
  {
    b1.display();
  }
  else if (b2.price > b2.price)
  {
    b2.display();
  }
else
{
  cout << "both books have same price";
    b1.display();
    b2.display();
}
return 0;
}
```

(Q3) Write a C program to declare class 'time'. Accept a time in HH:MM:SS format, convert it in seconds and display them.

```
#include <iostream>
{using namespace std;
Class time
{
  int H;
  Plint M;
  int S;
```

```
public :
  void accept();
{
   cout<<"Enter the time:";
   cin>>H>>M>>S;
}
  void display()
{
   int total;
   total=((H*3600)+(M*60)+S);
   cout<<"Time in seconds="<<total;
}
};
int main()
{
  time t1;
  t1.accept();
  t2.accept();
  t1.display();
}
return 0;
}
```

31/7/25

PAGE No. DATE / /

Experiment – 2

Q1) 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.

```
#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 having highest population:"<<nam
    }
};
int main()
{
    city c[5];
    int i, max;
    for(i=0; i<5; i++)
    {
        c[i].accept();
    }
    max = 0;
```

```
{for(i=0; i<5; i++)
{
    if(c[i].population > c[max].population)
    {
        max = i;
    }
c[max].display();
return 0;
}
```

Q2) WAP to declare a class 'Account' having data members as account number and balance. Accept this data for 5 accounts and give interest of 10% where account balance is equal to or greater than 5000 and display them.

```
#include <iostream>
using namespace std;
class account
{
public:
int acc_no;
int bal;
void accept()
{
cout<<" Enter the account no: ";
cin>> acc_no;
cout<<" Enter the balance: ";
cin>> bal;
```

```
};
};
int main()
{
    account a[5];
    int i;
    for(i=0; i<5; i++)
    {
        a[i].accept();
    }
    for(i=0; i<5; i++)
    {
        if (a[i].bal>=5000)
        {
            a[i].bal = a[i].bal + (a[i].bal * 0.1);
            cout<<"account number:"<<a[i].acc_no<<"\n";
            updated "balance:"<< a[i].bal;
        }
    }
    return 0;
}
```

Q3) WAP To declare a class 'Staff' having data members as name and post. Accept this data for 5 Staff's and display the name of staff whose post is 'HOD'.

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

int main()
{
    staff s[5];
int i;
for(i=0; i<5; i++)
{
    s[i].accept();
}
for cout<<"\n Staff members with post 'HOD':\n";
{for(i=0; i<5; i++)
```

PAGE No.
DATE / /

```
if(s[i].post == "HOD"){
    cout<<s[i].name<<endl;
}
return 0;
}
```

31/7/25

Experiment - 3

Q1) WAP to declare a class 'book' containing data members as book_title, author_name, and price. Accept and display the information for one object using pointer to that object.

```
#include <iostream>
using namespace std;
class book
{
  public:
  string book_title;
  string author_name;
   int price;
   void accept()
  {
     cout<<"Enter book title";
     cin>>book_title;
     cout<<"Enter author name";
     cin>>author_name;
     cout<<"Enter price";
     cin>>price;
  }
   void display()
  {
     cout<<"Book title="<<book_title;
     cout<<"Author name="<<author_name;
     cout<<"Price="<<price;
  }
};

int main()
{
```

```
book b1;
book *p;
p = &b1;
p -> accept();
p -> display();

return 0;

}
```

Q2) WAP to declare a class 'student' having data members roll no, and percentage. Using 'this' pointer invoke member function to accept and display this data for one object of the class.

```
#include <iostream>
using namespace std;
class student
{
    int roll;
    float percentage;
public:
    void accept()
    {
        cout << "Enter roll no :";
        cin >> this -> roll no;
        cout << "Enter percentage :";
        cin >> this -> percentage;
    }
    void display()
    {
        cout << "Student Details : \n";
```

```
    cout<<" Roll Number :" << this→ roll <<endL;
    cout<<" Percentage :" << this→ percentage <<"%"<<endL;
  }
};

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

Q3) WAP to demonstrate use of nested ~~loc~~ class.

```
#include <iostream>
using namespace std;
class student {
 int roll;
  string name;
  public:
   void accept(){
    cout<<" Enter roll no:" ;
    cin >>roll;
    cout<<" Enter name :";
     cin>> name;
  }
     void display() {
     cout<<" Roll no :" << roll;
     cout<<" Name :" << name;
  }
     class Marks {
     int m1, m2;
```

```
public :
    void accept1() {
    cout << " Enter m1 and m2";
    cin >> m1 >> m2
}

    void calculate()
{
    int m3;
    m3 = m1 + m2 / 2;
}

    void display1()
{
        cout << " average marks =" << m3;
}
};
    int main()
{
    Student S1;
    S1. accept();
    Student :: Marks Sm;
    Sm. accept1();
    Sm. calculate();
    S1. display();
    Sm. display1();

return 0;

}
```

Q1

1418

Experiment — 4

Q1) WAP to swap two numbers from same class using ~~concept of friend function~~ object as function argument. Write swap function as member function.

```
#include <iostream>
using namespace std;
class number
{
  int n1, n2;
  public:
    void accept()
  {
    cout<<" Enter values of n1 and n2 ";
    cin>>n1>>n2;
  }
    void display()
  {
    cout<<" Swapped value of n1:"<<n1;<<endl;
    cout<<" Swapped value of n2:"<<n2;<<endl;
  }
  void swap(number &obj)
{
    int temp = obj.n1;
    obj.n1 = obj.n2;
    obj.n2 = temp;
}

};
  int main()
{
  number nm;
```

```
nm.accept();
nm.swap(nm);
nm.display();
return 0;
}
```

Q2) WAP to swap two numbers from same class using concept of friend function.

```
#include <iostream>
using namespace std;
class number
{
    int n1, n2;
    public:
    void accept()
    {
        cout<<"Enter values of n1 and n2";
        cin>>n1>>n2;
    }
    void display()
    {
        cout<<"Swapped value of n1:"<<n1<<endl;
        cout<<"Swapped value of n2:"<<n2<<endl;
    }
    friend void swap(number &nm);
};
void swap(number &nm)
{
    int temp;
    temp = nm.n1;
    nm.n1 = nm.n2;
    nm.n2 = temp;
}
```

```
int main()
{
    number p;
    p.accept;
    swap (p);
    p.display;
    return 0;
}
```

Q3) WAP to ~~swap~~ find greatest of two numbers from different class using friend function.

```
#include <iostream>
using namespace std;
class BB;
class AA {
  int a;
  public:
    void info ()
  {
    cout << " Enter value of a:" ;
    cin >> a ;
  }
  friend void avg (AA a1 , BB b1);
};
  class BB
{
  int b;
  public:
  void info1 ()
{
  Cout << " Enter value of b" ;
  cin >> b1;
}
```

```
};friend void avg (AA a1, BB b1);

{ void avg (AA a1, BB b1)
    { if (a1.a > b1.b)
    }
        cout <<"\n a is the greater value";
    else {
        cout <<"\n b is greater";
    }
}
int main() {
    AA a;
    BB b;
        a.info();
        b.info1();
        avg(a,b);
    }
        return 0;
}
```

Q4) WAP to create two classes Result1, and Result2 which stores the marks of the students Read the value of a student for both the class objects and compute the average of two results.

```
#include <stdio.h>
using namespace std;
class result2;
class result1 {
    int a;
```

```
public :
    void accept() {
    cout << " Enter marks";
     cin >> a;
}
    friend void avg R(result1.r1, result2 r2);
};
  class result 2
{
     int b;
    public:
      void acceptH() {
     cout << " Enter the marks";
      cin >> b;
    friend void avg (result r1, result r2);
};
     void avg ( result r1, result r2)
  {
     float avg = (float) (r1.a + r1.b)/2;
      cout << " avg";
  }
    int main() {
    result1 x;
    result 2 y;
     x. accept();
     y. accept1();
     avg (x,y);
    return 0;
}
```

Experiment – 5

Q1. WAP to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the Constructor.

① Using Concept of Default Constructor

```
#include <iostream>
using namespace std;
class number {
 int n;
 public:
   number()
   {
      n = 5;
   }
   void calculate()
   {
      int sum=0, i;
      for(i=0; i<=n; i++)
      {
         sum = sum + i;
      }
      cout<<"Sum = "<<sum;
   }
};
int main() {
   number n1;
   n1.calculate();
   return 0;
}
```

Output :

Sum = 15

PAGE No.
DATE / /

② Using Concept of Parameterized Constructor

```
#include <iostream>
using namespace std;
class number {
    int num;
public:
    number (int n)
    {
        int sum=0;
        for(int i=0; i<=num; i++)
        {
            sum = sum+i;
        }
    }
    void display()
    {
        cout << " sum = " << sum;
    }
};
int main()
{
    number s1(5);
    s1.display();
    return 0;
}
```

Output

Sum = 15

PAGE No.
DATE / /

③ Using Concept of Copy Constructor

```
#include <iostream>
using namespace std;
class sum
{
 int n;
 int total;
public:
 sum (int num)
 {
    n = num;
    total = 0;
 for (int i = 1; i <= n; i++)
 {
        total = i++;
 }
}
 sum (const sum &obj)
 {
    n = obj.n;
    total = obj.total;
 }
 void display () {
 cout << "sum = " << total;
 }
};
int main () {
int num;
cout << " Enter a number: ";
cin >> num;
sum S1 (num);
sum S2 = S1;
```

```
s2.display();
return 0;
}
```

Output

Enter a number : 5

sum = 5

Q2) WAP to declare class student having 2 data members as name and percentage. Write constructor to initialise these data members.

① Using Default Constructor Concept

```
#include <iostream>
using namespace std;
class student
{
  int per;
  string name;
  public:
    student()
    {
      per = 70;
      name = "Prathamesh";
    }
    void display()
    {
      cout<<"Name = "<<name<<endl;
      cout<<"Percentage = "<<per<<endl;
    }
};
```

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

Output
Name = Prathamesh
Percentage = 70

② Using Parameterized Constructor Concept

```
#include <iostream>
using namespace std;
class Student {
int per;
string name;
 public:
  Student (int p , string n)
{
   per = p;
   name = n;
}
 void display () {
cout <<"Percentage = "<<per;
 cout<<" Name = "<< name;
}
};
 int main(){
 Student S1 (90, "Prathamesh");
 S1.display();
 return 0;
}
```

Output
Percentage = 90
Name = Prathamesh

③ Using Copy Constructor Concept

```
#include <iostream>
using namespace std;
class student {
 int per;
 String name;
 public:
   student ()
{
    per = 370;
    name = "abc";
}
  Student (const Student &S)
{
    per = S.per;
    name = S.name;
}
  void display ()
{
  cout<<" Percentage ="<<per<<endl;
  cout<<" Name ="<<name<<endl;
}
};
 int main () {
 Student S1;
 Student S2 = S1;
 cout<<" Student 1 :"<<endl;
  S1.display ();
```

```
cout<<" Student 2:" <<endl;
s2.display();
return 0;
}
```

Output

Student 1:
Percentage = 70
Name = abc
Student 2:
Percentage = 70
Name = abc

Q3) Define a class 'College' members as roll_no, name, course. WAP using constructor with default value as "Computer Engineering" for course. Accept and display data for 2 objects.

```
#include <iostream>
using namespace std;
class college {
int roll;
String name, course;
public:
 college ( int r, String n, String c = "computer
          Engineering")
{
 roll = r;
 name = n;
 course = c;
}
 void display (){
cout<<" name, roll no and course is :" <<name
<<roll<< course ; } };
```

```
int main()
{
    college c1(1, "Prathamesh", "CSE");
    college c2(1, "Prathamesh", "CSE");
    c1.display();
    c2.display();
    return 0;
}
```

Output

name, roll and course is : Prathamesh
1
CSE
name, roll and course is : Prathamesh
1
CSE

Q4) WAP to demonstrate constructor overloading.

```
#include <iostream>
using namespace std;
class rectangle {
  int l, w;
  public:
   rectangle ()
{
    l = 2;                    // default Constructor
    w = 1;
}
 rectangle (int a, int b)
{
    l = a;
   w = b;
}
 void calculate ()
{
    int a;
    a = l * w;
    cout << " Area = " << a;
}
};
  int main ()
{
    rectangle r1;
    rectangle r2 (5);
    rectangle r3 (4,5);
    r1. calculate ();
    r2. calculate ();
    r3. calculate ();
  return 0;
}
```

Output

Area = 2 Area = 25 Area = 20

Experiment – 6

Q1. Single Inheritance

Problem:

Create a base class called person with attributes name and age. Derive a class student with person that adds an attribute roll_no. Write functions to display all details of the Student.

```
#include <iostream>
#include <string.h>
using namespace std;
class person {
  protected:
    string name;
    int age;
};
class student : protected person
{
  private:
    int roll;
  public:
    void accept()
{
    cout << " Name:";
    cin >> name;
    cout << " Age :";
    cin >> age;
    cout << " roll no :";
    cin >> roll;
}
    void display () {
cout << "Name :" << name << endl;
cout << " Age :" << age << endl;
cout << " Roll no :" << roll << endl;
```

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

Output

```
Enter name: Prathamesh
Enter age: 17
Enter roll number: 71
name: Prathamesh
age: 17
roll number: 71
```

Q2 Multiple Inheritance

Problem:
Create two base classes, Academic and Sports
- Academic class contains marks of a student
- Sports class contains sports score.

Create a derived class result that inherits from both academic and sports.
Write a function to calculate the total score and to display details.

```
#include <iostream>
#include <string.h>
using namespace std;
class academic {
protected:
  int marks;
};
class sports {
protected:
  int spmarks;
};
class result: protected academic : protected sports
{
  public:
  int total;
  void accept() {
  cout<<" Enter the academic score:";
   cin >> marks;
  cout<<" Enter the sports marks:";
   cin >> spmarks;
}
 void calculate()
{
```

```
    total = marks + spmarks;
    cout<<"Academic marks:"<<marks<<endl;
    cout<<"Sports Score:"<<spmarks<<endl;
    cout<<"Total marks:"<<total<<endl;
  }
};
int main(){
  result r;
  r.accept();
  r.calculate();
  return 0;
}
```

Output

```
Enter the academic score: 50
Enter the sports score: 30
Academic scoremarks: 50
Sports score: 30
Total marks: 80
```

PAGE No.
DATE / /

Q3) Multilevel Inheritance

Problem:
Create a class vehicle with attributes like brand and model. Derive a class Car from vehicle which adds an attribute type (eg suv, sedan). Further derive a class electric car which adds battery capacity. Write functions to display all the details.

```
#include <iostream>
using namespace std;
class vehicle {
public:
  string brand;
  int model;
};
class car : public vehicle {
public:
  string type;
};
class electriccar : protected car {
private:
  int battery;
public:
void accept() {
cout<<"Enter Car brand:";
 cin >> brand;
 cout<<" Enter model the model:";
  cin >> model;
  cout<<" Enter the type:";
  cin >> type;
cout<<" Enter the ELECTRIC CAR'S BATTERY
      CAPACITY:";
 cin >> battery;
}
```

```
void display (){
cout<<" CAR BRAND :" << brand <<endl;
cout<<" MODEL :" << model <<endl;
cout<<" BODY TYPE :" << type <<endl;
cout<<" BATTERY CAPACITY :" <<battery <<endl;
}
};
int main ()
  electriccar e;
  e. accept ();
  e. display ();
  return 0;
}
```

Output

enter the car brand: Range rover
enter the model: 2016
enter the type: SUV
enter the electriccar's battery capacity : 90
CAR BRAND: Range rover
Model : 2016
BODY TYPE: SUV
BATTERY CAPACITY : 90

PAGE No.
DATE / /

Q4) Hierarchial Inheritance:

Problem -

Create a base class Employee with attributes empId and name. Derive two classes Manager and Developer from employee. Manager has an attribute department and developer has an attribute programming language. Write all functions to display details for both.

```
#include <iostream>
using namespace std;
class employee {
protected:
  int emp_id;
    string name;
};
class manager : protected employee {
private:
  String dname;
  public:
    void accept () {
  cout<<" Enter the employee id :";
  cin>> emp_id;
  cout<<" enter the employee name :";
  cin >> name;
  cout<<" enter the department name :";
    cin >> dname;
}
  void display () {
cout<<" EMPLOYEE ID :"<<emp_id<<endl;
cout<<" EMPLOYEE NAME :"<<name<<endl;
cout<<" DEPARTMENT :"<<dname<<endl;
  }
};
```

```
class developer : protected employee {
private :
string lan;
public:
void accept1() {
cout<<" enter the developer's language :";
cin>>lan;
}
void display1() {
cout<<" DEVELOPER'S LANGUAGE :" <<lan<<endl;
}
};

int main() {
    manager m;
    developer d;
    m.accept();
    d.accept();
    m.display();
    d.display();
    return 0;
}
```

Output

```
enter the employee id : 1
enter the employee name : PRATHAMESH
enter the department name : CSE
enter the developer's language : PYTHON
EMPLOYEE ID : 1
EMPLOYEE NAME : PRATHAMESH
DEPARTMENT : CSE
DEVELOPER'S LANGUAGE : PYTHON
```

Q5) Hybrid Inheritance

Problem -

Combine multilevel and multiple Inheritance.
Create a base class person with attributes name and age
Derive a class student from person. roll no.
Creat two Classes Sports and Academics
Derive class Result ~~es~~ from Student and sports. Write all functions and display total marks, sports score, academic score with student details.

```
#include <iostream>
using namespace std;
class person {
protected:
  String name;
  int age;
};
Class Student : protected person {
privicte:
  int roll;
public:
  void accept () {
cout << " enter the name :";
  cin >> name;
cout << " enter the age :";
  cin >> age;
cout << " enter the roll number :";
  cin >> roll;
}
  void display () {
cout << " NAME :" << name << endl;
cout << " AGE :" << age << endl;
cout << " ROLL NUMBER :" << roll << endl;
```

```
    }
};
class Sports {
protected:
    int score;
};
class academics {
protected:
    int grade;
};
class result : protected Sports, protected academics {
private:
    int total;
public:
    void accept1() {
    cout << "enter the sports score:";
    cin >> score;
    cout << "enter the academics score:";
    cin >> grade;
}
    void calculate() {
    total = score + grade;
}
    void display1() {
    cout << "SPORTS SCORE:" << score << endl;
    cout << "ACADENIC GRADE:" << grade << endl;
    cout << "TOTAL SCORE:" << total << endl;
}
};
```

```
int main() {
  Student s1;
  result r1;
  s1.accept();
  r1.accept1();
  r1.calculate();
  s1.display();
  r1.display1();
  return 0;
}
```

Output

```
enter the name : Prathamesh
enter the age : 17
enter the roll number : 71
enter the sports score : 50
enter the academic score : 50
NAME : Prathamesh
AGE : 17
ROLL : 71
SPORTS SCORE : 50
ACADEMIC SCORE : 50
TOTAL SCORE = 100
```

Experiment – 7

Q1) WAP using function overloading to calculate the area of a laboratory (which is rectangular) and classroom (which is square)

```
#include <iostream>
using namespace std;
class areacalculator {
  public:
    float area(float length, float width) {
      return length * width;
    }
    float area(float side) {
      return side * side;
    }
};
int main() {
  areacalculator calc;
  float lablength, labwidth, classside;
  cout << "Enter length:";
  cin >> lablength;
  cout << "Enter width:";
  cin >> labwidth;
  cout << "Enter side:";
  cin >> classside;
cout << "Area of laboratory (Rectangle):" << calc.area
(lablength, labwidth) << endl;

cout << "Area of classroom (Square):" << calc.area
(classside) << endl;

  return 0;
}
```

Output
Enter length : 5
Enter Width : 7
Enter Side : 2
Area of Laboratory (Rectangle) : 35
Area of Classroom (Square) : 4

Q2) WAP using function overloading to calculate the sum of 5 float values and sum of 10 integer values.

```
#include <iostream>
using namespace std;
float Sum(float a, float b, float c, float d, float e)
{
    return a+b+c+d+e;
}
int Sum(int a1, int a2, int a3, int a4, int a5, int a6,
        int a7, int a8, int a9, int a10)
{
    return a1+a2+a3+a4+a5+a6+a7+a8+a9;
}
int main(){
float f1, f2, f3, f4, f5;
cout<<"Enter 5 float values:";
cin>>f1>>f2>>f3>>f4>>f5;
int i1, i2, i3, i4, i5, i6, i7, i8, i9, i10;
cout<<"Enter 10 integer values:";
cin>>i1>>i2>>i3>>i4>>i5>>i6>>i7>>i8>>
    i9>>i10>>endl;
cout<<"Sum of integer values="<<Sum
(i1, i2, i3, i4, i5, i6, i7, i8, i9, i10)<<endl;
return 0; }
```

PAGE No.
DATE / /

Q3) WAP to implement Unary operator when used with the object so that the number data member of it is negated

```
#include <iostream>
using namespace std;
class number {
private:
int value;
public:
number(int v=0): value(v) { }
number operator -() const {
return number(-value);
}
void display() const {
cout<<"Value :" << value <<endl;
}
};
int main() {
number num1(10);
number num2;
cout<<"Original no:" <<endl;
num1.display();
num2 = -num1;
cout<<"After applying unary-operator" <<endl;
num2.display();
return 0;
}
```

Output:
Original number:
Value : 10
After applying unary-operator:
Value : -10

Q4) WAP to implement unary ++ operator (for pre- and post increment) when used with object so that numeric data member of the class is Incremented.

```
#include <iostream>
using namespace std;
class number{
  int value;
public:
number (int v=0) : value (v) {}
number &operator ++() {
    ++ value;
  return *this;
}
number operator ++(int){
 number temp = *this;
  value ++;
  return temp;
}
  void display () const {
  cout << "Value :" << value << endl;
}
};
  int main () {
number n (5);
cout << "Initial Value :" << endl;
n.display ();
cout << "Pre-Increment Value :" << endl;
n.display ();
cout << "Post Increment Value :" << endl;
n.display ();
return 0;
}
```

Output

Initial Value:
5

Pre-Increment Value:
6

Post Increment Value:
7

Experiment – 8

PAGE No. / DATE / /

(Q1) WAP to overload '+' operator so that two strings can be concatenated

```
#include <iostream>
using namespace std;
class mystring {
private:
  string str;
public:
mystring (string s = "") : str(s) { }
mystring operator + (const mystring &other)
  const {
  return mystring (str + other.str);
}
  void display() const {
  cout << str << endl;
}
};
  int main() {
  mystring s1("xyz");
  mystring s2("abc");
  mystring s3 = s1 + s2;
  cout << "Concatenated string:";
  s3.display();
  return 0;
}
```

Output :

Concatenated String = xyzabc

Q2) WAP to create a base class Ilogin having data members name and password. Declare accept() function virtual. Derive Email login and membership login class from Ilogin. Display Email login details and membership login details of the employee.

```
#include <iostream>
#include <string>
using namespace std;
class Ilogin {
protected:
    string name;
    string 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;
    }
    virtual ~Ilogin() { }
};
    class Emaillogin : public Ilogin {
    string email_id;
public:
void accept() override {
cout << "Email login details :" << endl;
cout << "Enter name:";
cin >> name;
cout << "Enter Password:";
```

```
cin >> password;
cout << " Enter email id :";
cin >> email_id;
}

    void display() override {
cout << " Email login details :" << endl;
cout << " Name :" << name << endl;
cout << " Password :" << password << endl;
cout << " Email id :" << email_id << endl;
}
};

class membershiplogin : public ILogin {
    String memberId;
public:
void accept() override {
cout << " Membership login details :" << endl;
cout << " Enter Name :";
cin >> name;
cout << " Enter Password :";
cin >> password;
cout << " Enter membership Id :";
cin >> memberId;
}
void display() override {
cout << " Membership login details :" << endl;
cout << " Name :" << name << endl;
cout << " Password :" << password << endl;
cout << " Membership Id :" << memberId << endl;
}
};
```

```
int main () {
Emaillogin emailuser;
emailuser. accept ();
emailuser. display ();

membershiplogin memberuser;
memberuser. accept ();
memberuser. display ();
return 0;
}
```

Output

Enter name: Prathamesh
Enter password: pratham08
Enter Email Id: pratham@gmail.com

Email login details
Name: Prathamesh
password: pratham08
Email Id: pratham@gmail.com

Membership login details
Name: Prathamesh
password: pratham08
Membership Id: 71

Experiment - 9

Q1) WAP to copy the contents of one file into another. Open "First.text" in read mode (ios.in) and second.text in write mode. Copy the contents of 'first.text' into 'second.text'. Assume 'first.text' is already created.

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

int main() {
    ifstream fin("First.txt", ios::in);
    ofstream fout("Second.txt", ios::out);

    if (!fin) {
        cout << "Error: Could not open file";
        return 1;
    }

    char ch;
    while (fin.get(ch)) {
        fout.put(ch);
    }

    cout << "File copied Successfully!";
    fin.close();
    fout.close();
    return 0;
}
```

Output

File Copied Successfully.

Q2) WAP to count digits and spaces while file handling.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ifstream fin("First.text", ios::in);
    if (!fin) {
        cout<<"File creation failed";
    }
    else {
        char ch;
        int digit = 0, spaces = 0;
        while (fin.get(ch)) {
            if (isdigit(ch)) {
                digits++;
            }
            if (isspace(ch)) {
                spaces++;
            }
        }
        cout<<"Total Digits :"<<digits<<endl;
        cout<<"Total Spaces :"<<spaces<<endl;

        fin.close();
        return 0;
    }
}
```

Output

Total Digits : 0

Total Spaces = 0

PAGE No.
DATE / /

Q3) WAP to Count words using file handling.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {

ifstream fin ("First.text");
  if (!fin) {
   cout<<"File creation failed:";
}

String word;
int count =0;
while( fin >> word) {
    count ++;
}

cout<<"Total Words :"<< cocent <<endl;

fin. close ();
  return 0;
}
```

Output

Total Words : 0

Q4) WAP to count occurence of a given word using file handling.

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

int main(){

  ifstream fin("FIRST.txt");
  if(!fin){
  cout<<"File could not open";
    return 1;
  }

  string word, target;
  int count=0;

  cout<<"Enter the word to search:";
  cin>>target;

  while(fin>>word){
    if(word==target)
  {
      count++;
  }
  }

  cout<<"The word"<<target<<"occured"<<count<<
    times";
  fin.close();
    return 0;
}
```

12/11

Experiment – 10

Q1) WAP to find sum of array elements using function template (eg. pass integer, float and double array of 10 elements)

```
#include <iostream>
using namespace std;
template <class t> t func(t a[], int n){
    t sum=0;
    int i;
    for(i=0;i<n;i++){
        sum=sum+a[i];
    }
    return sum;
}
int main(){
int a1[3]={2,0,2};
float a2[3]={1.2,2.2,2.4};
double a3[3]={10.5,20.5,30.5};
cout<<"Sum of integer array is:"<<func(a1,3)<<endl;
cout<<"Sum of float array is:"<<func(a2,3)<<endl;
cout<<"Sum of double array is:"<<func(a3,3)<<endl;
return 0;
}
```

Output:

Sum of integer array is: 4
Sum of float array is: 5.8
Sum of double array is: 61.5

Q2) WAP of Square function using template Specialization:

1. Calculate the square of integer no and a string
2. Write a specialized function for the square of a string.

```
#include <iostream>
using namespace std;
template <class T> T Square (T x){
    T result;
    result = x*x;
    return result;
}

template <> String Square <String> (String ss){
    return (ss+ss);
}

int main (){
int i = 2; ii;
String www ("MITWPU");
ii = Square <int> (i);
cout << i << ":" << ii << endl;
cout << Square <String> (www) << endl;
return 0;
}
```

Output

2:4

MITWPUMITWPU

Q3) WAP to build a simple calculator using class template.

```
#include <iostream>
using namespace std;
template <class T>
class calc{
public:
  T num1=8;
  T num2=8;
 void add(){
cout<<"Addition ="<<num1+num2<<endl;
}
  void sub(){
cout<<"Substraction ="<<num1-num2<<endl;
}
  void mul(){
cout<<"Multiplication ="<<num1*num2<<endl;
}
  void div(){
  cout<<"Division ="<<num1/num2<<endl;
}
};
int main(){
 calc <int> d;
 d.add();
 d.sub();
 d.mul();
 d.div();
 return 0;
}
```

Output :
Addition = 816
Substraction = 0
Multiplication = 64
Division = 1

Q4) WAP to implement push and pop methods from Stack using class template

```
#include <iostream>
#include <vector>
using namespace std;
template <typename T>
class Stack {
private:
    vector <T> elements;
public:
void push (T item) {
    elements.push_back(item);
cout<<item<<" pushed to stack\n";
}
    void pop(){
    if (elements.empty()) {
    cout<<" Stack is empty cannot pop\n";
    return;
}
    cout << elements.back()<<" popped\n";
    elements.pop_back();
}
    void display(){
cout<<" Stack elements:";
    for (auto item : elements)
{   cout <<item<<" ";
} cout<<endl; }
```

```
};

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

Output

```
10 pushed to Stack
20 pushed to Stack
30 pushed to Stack
Stack elements: 10 20 30
30 popped
Stack elements: 10 20
```

Experiment – 11

~~int main(){~~

Q1) WAP to implement generic vectors, and include following member functions

a) To modify the value of a given element,
*Display, Print the size of vector

```
#include <iostream>
#include <vector>
#include <cctype>
using namespace std;
int main() {
vector <char> v(10);                    //* Creation *//
  int i;
cout<<" Size =" << v.size()<<endl;      //Display Size//

for(i=0; i<10; i++) {
  v[i] = i + "a";
}
cout<<" Current Contents:\n";           //* Display Contents *//
for (i=0; i<10; i++) {
   cout<<v[i]<<" ";
   cout<<"\n\n";

cout<<" Expanding vector:\n";
for (i=0; i<10; i++){
  v.push_back (i + 10 + " a");

Cout<<"Current size =" << v.size() <<endl;

for (i=0; i<10; i++) {
   v[i]= toupper(v[i]);
cout<<" Modified Contents=:\n";
  for (i=0; i<10; i++) {
     cout<<v[i] <<endl; }
```

```
return 0;
}
```

Output:

```
Size : 10
Current Contents:
a
b
c
d
e
f
g
h
i
j
expanding vector:
Current size : 20
Modified Contents:
A
B
C
D
E
F
G
H
I
J
```

PAGE No.
DATE 11/

b) To multiply by a scalar value.

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
vector <int> vec = {1,2,3,4,5};
 int Scalar = 3;
 for (int &val : vec) {
     val = val * Scalar;
 }
 for (int val : vec) {
    cout << val << " ";
 }
 cout << endl;
 return 0;
}
```

Output:
3 6 9 12 15

12/11

Experiment – 12

Q1) WAP using STL

a) Implement Stack

```
#include <iostream>
#include <stack>
using namespace std;
int main () {
Stack <int> S;
  S. push (10);
  S. push (20);
  S. push (30);
cout << " Top element :" << S. top () << endl;
 S. pop ();
 cout << " Top element after pop :" << S. top () << endl;
cout << " Stack size = " << S. size () << endl;
if (S. empty ()) {
    cout << " Stack is empty" << endl;
}
else {
    cout << " Stack is not empty" << endl;
}
return 0;
}
```

Output

Top element : 30
Top element after pop : 20
Stack size : 2
Stack is not empty

b) Implementation of queue

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
 queue <int> q;
  q.push(10);
  q.push(20);
  q.push(30);
cout << "Front elements: " << q.front() << endl;
cout << "Back elements: " << q.back() << endl;
q.pop();
cout << "after pop operation: " << endl;
cout << "Front element: " << q.front() << endl;
cout << "Queue Size: " << q.size() << endl;
if (q.empty()) {
   cout << "Queue is empty" << endl;
}
else {
   cout << "Queue is not empty" << endl;
}
return 0;
}
```

Output

Front elements : 10
Back element : 30
after pop operation
front element : 20
Queue Size : 2
Queue is not empty

c) Implement Sorting and Searching with user defined records such as personal records (name, birth-date, telephone-no), item record (item code, item name, quantity and cost)

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

Struct Person {
    String name;
    int age;
Person (String n, int a) : name(n), age(a) {}
};

int compare_age(const Person &p1, const Person &p2)
{
    return (p1.age < p2.age);
}

int main() {
vector<Person> people = {
    Person ("Alice", 30);
    Person ("Bob", 25);
    Person ("David", 28);
    Person ("Charlie", 35); };

Sort (people.begin(), people.end(), [
    (const person &a, const person &b) {
    return compare_age (a,b); }
cout << " Sorted Records by age: \n";
for (auto &p : people) {
    cout << p.name << " " << p.age << "\n";
}
```

```
int search_age = 28;
int found index = -1;
for(int i=0; i<(int) people.size(); i++){
  if (people[i].age == search_age ? 1 : 0) {
      foundindex = i;
break;
}
}
if (foundindex != -1){
 cout << " person with age " << search_age << " found: " <<
 people[foundindex].name << "\n"; }
else
{
  cout << " person with age " << search age << " not
  found " << endl;
return 0;
}
```

Output:

Sorted records by age

Bob - 25

David - 28

Alice - 30

Charlie - 35

Person with age 28 found: David

12/11