PAGE NO
DATE / /

Experiment - 01

1) Write a 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 {
  string name;
  int roll no;

public :
    void accept () {
    cout << "Enter student Name : ";
    cin >> name;
    cout << "Enter Roll No : ";
    cin >> roll no;
  }

  void display () {
    cout << "student Name : " << name << endl;
    cout << "Roll no:" << roll no << endl;
  }
};
int main () {
  Student student;

  student.accept ();
  student.display ();
    return 0;
}
```

Output :-

Enter student Name - Rudra.
Enter student Roll no = 77

Student Details :
Name : Rudra.
Roll no : 77.

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

=> #Include <iostream>
using namespace std;

class Book {
int id;
string name;
float price;

public :
void accept () {
cout << "Enter book ID : ";
cin >> id;
cout << "Enter book name : ";
cin >> name;
cout << "Enter book price : ";
cin >> price;
}

```
void display () {
cout << "Book ID : " << id << endl ;
cout << "Book Name : " << name << endl ;
cout << "Book Price : " << price << endl ;
}

float getPrice () {
    return price ;
  }
};

int main () {
  int n ;
  cout << "Enter number of books : " ;
  cin >> n ;

  Books books [n] ;

  for (int i = 0 ; i < n ; i++) {
    cout << "\n Enter details of book " << i+1 << " : "
          << endl ;
    books [i] . accept () ;
  }
  int maxIndex = 0 ;
  for (int i = 1 ; i < n ; i++) {
    if (book [i] . getPrice () > books [maxIndex] .
      getPrice () ) {
      maxIndex = i ;
    }
  }
  cout << "\nBook with the greatest price : " << endl ;
  books [maxIndex] . display () ;
  return 0 ;
}
```

Output :

Enter Number of books : 2
Enter details of book 1 :
Book ID : 5565
Book Name : kkii
Book price : 987
Enter details of book 2 :
Book ID : 45
Book Name :- tdhied
Book Price : 65

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

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

   class Time {
       int H, M, S;

   public :
       void accept()
        cout << "Enter hours : ";
        cin >> H;
        cout << "Enter minutes : ";
        cin >> M;
        cout << "Enter seconds : ";
        cin >> S;
   }
```

```
void displayTotalSeconds () {
    int totalseconds = H*3600+M*60 +S;
    cout << "Total time in seconds: "<< totalSeconds
    << endl ;
  }
};
int main () {
    Time t ;
    t.accept ();
    t.displayTotalSeconds ();

    return 0;
}
```

Output :-

Enter hours : 4
Enter minutes :- 23
Enter seconds : 54
Total time in seconds :- 15834.

Q
15/7

Experiment - 02.

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

```
~~#include<stdio~~
#include<iostream>
using namespace std;
class city
{
    public:
    string name;
    int population;
    void accept ()
{
    cout << "Enter city Name:";
    cin >> name;
    cout << "Enter city Population:";
    cin >> population;
}
    void disp()
{   cout << "city having highest population:"<<name;}}
int main()
{   city c[5];
    int i, max;
    for(i=0; i<5; i++)
{   c[i].accept();}
    max = c[0].population;
    for (i=0; i<5; i++)
{   if( c[i].population > max)
```

```
{max = i ; }}
c[max] . disp()
  return 0;
}
```

Output:

```
Enter city Name : Pune
Enter city Population : 600
Enter city Name : Mumbai
Enter city Poupulation : ~~Sat~~ 900
Enter city Name : Satara
Enter city Population : ~~Mahabaleshwar~~ 1000
Enter city Name :- Mahabaleshwar
Enter city Population : 1100
Enter city Name : Yevla
Enter city population : 200
```

6) WAP to declare a class 'Account' having data Members as Account no. and balance. Take data for 10 accounts.

```
#include<iostream>
using namespace std;
class Account {
public:
    int account_no;
    float balance;

void accept(){
cout << "Enter account number and balance:";
cin >> account_no >> balance;
```

```
}
    void apply Internet () {
        if(balance >= 5000) {
            balance += (balance * 0.10);
        }
    }
    void display () {
        cout << "Account Number : " << account_no << ";
                Balance after Interest : " << balance << endl;
    }

};
int main ()
    Account accounts[10];

    for (int i=0 ; i<10 ; i++) {
        account[i].accept ();
    }

    cout << "\n Account with balance >= 5000 after
            applying interest : \n ";
    for(int i=0 ; i<10 ; i++) {
        accounts[i] .apply Interest ();
        if(accounts[i] .balance >=5000) {
            account[i] . display ();
        }
    }

    return 0;
}
```

PAGE NO.
DATE / /

c) Take details of 2 employees and stimate them with each other.

```
#include <iostream>
using namespace std;
class staff
{ string name;
 public:
  string post;
  void accept ()
  { cout << "Enter the staff name:";
   getline (cin, name);
   cout << "Enter the staff post:";
   getline (cin, post); }
  void disp ()
  {cout << "\n Staff Name : " << name;
   cout << "\n Staff Post : " << post; }};
int main ()
{ staff s[5];
 int i;
 for (i=0; i<5; i++)
  {s[i].accept (); }
  for (i=0; i<5; i++)
 { if (s[i].post == "Hop")
  { s[i].disp(); }}
  return 0;
}
```

9/9

PAGE No.
DATE / /

Experiment - 03.

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

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

PAGE No.
DATE / /

```
int main ()
{
    book bi;
    book * p;
    p = &bi;
    p -> accept ();
    p -> display ();
}
```

Output :

Enter book title : Ugylite
Enter Author Name : Rudra.
Enter Price of Book : 1500r
Book Title : Ugylite.
Author Name : Rudra.
Book Price : 1500r

(2) WAP to delare 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 class.

```
#include <iostream>
using namespace std;
class Student
{
    int roll_no;
    float p;
    public :
    void accept ()
{
    cout << "Enter Roll no";
```

```
    cin >> this -> roll no ;
    cout << "Enter percentage : " ;
     cin >> this -> p ;
  }
    void display ()
  {
     this -> accept ()
 cout << "Roll no" << roll no ;
  cout << " Percentage" << p ;
  }
   };
   int main ()
  {
    student S1 ;
   S1 display () ;
   return 0 ;
  }
```

Output:

```
Enter Roll no : 77
Enter Percentage : 80
Roll no : 77
 Percentage : 80.
```

PAGE No.
DATE / /

Q.

```
#include <iostream>
using namespace std ;
class student
{
public :
    string name ;
    int roll ;
void accept ()
{
    cout << "Enter the name of student \n" ;
    cin >> name ;
    cout << "Enter the roll number \n" ;
    cin >> roll no ;
}
void display ()
{
    cout << "The name of student" << name ;
    cout << "The roll no \n" << roll ;
}
class marks
{
Public :
int m1 , m2 ;
int m1 , m2 ;
void accept ()
{
cout << "Enter marks of two subject" ;
cin >> m1 >> m2 ;
}
void display1()
{
```

```
float per = ((m1+m2)/100.0)* 100;
    cout << " The percentage of 2 subjects is << per << endl";
}
};
int main ()
{
    student s1;
    student :: marks m;
    s1 accept ();
    m.accept1 ();
    s1.display ();
    m.display1 ();
    return 0;
}
```

Output :

```
Enter the name : Rudra.
Enter Roll no : 77.

Name : Rudra
Roll no : 77
Enter two marks
    32
    16
Percentage : 48.
```

9/9

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

```
#include <iostream>
using namespace std;
class class B;
class class A;
    int A;
public:
    void accept () {
    cout <<"Enter value for A:" ;
    cin >> A ;
}
friend int sum (class A oA, class B oB);
} na;
class class B {
    int B;
public:
    void accept () {
    cout <<"Enter value for B:" ;
    cin >> B ;
}
friend int sum (class A oA, class B oB);
} nb;
int sum (class A oA , class B oB);
    return oA.A + oB.B ;
}
int main () {
    na.accept ();
    nb. accept ();
```

```
cout << "Sum : " << sum(na, nb) << endl;
    return 0;
}
```

Output:

Enter value for A : 3
Enter value for B : 6
Sum : 9.

2. Write a program with a class Number that contains a private integer. Use a friend function swapNumbers (Number &, Number &) to swap the private values of two Number.

```
#include <~~stdio.h~~iostream>
using namespace std;
class Number {
    int num;
public:
        void accept () {
        cout << "Enter value : " ;
        cin >> num;
}
    void display () {
        cout << "Value :" << num << endl;
}
friend void swapNumbers (Number & num1, Number &
num2);
{ n1, n2 ;
void swapNumbers (Number &, Number &) {
    int temp = n1. num;
    n1. num = n2. num;
    n2. num = temp;
```

```
{
int main () {
    n1.accept ();
    n2. accept ();
cout << "Before swap : " << endl ;
    n1. display ();
    n2. display ();
swapNumbers (n1, n2);
cout << "After swap :" << endl ;
    n1. display ();
    n2. display ();
    return 0 ;
}
```

Output:

```
Enter value : 3
Enter value : 6
Before swap :
value : 3
value : 6
After swap.
value : 6
value : 3
```

3. Define two classes, class Box and Cube, each having a private volume. Write a friend function findGreater (Box, Cube) that determines which object has larger volume.

```
#include <iostream>
using namespace std;
class cube;
class Box {
    double v;
public:
    void accept () {
        double l, w, h;
    cout << "Enter length, width and height for Box:";
    cin >> l >> w >> h;
    v = l * w * h;
  } friend void findGreater (Box b, Cube c);
};
class Cube {
    double v;
public:
    void accept () {
    double s;
    cout << "Enter ~~value~~ side length for the cube:";
    cin >> s;
    v = s * s * s;
  } friend void findGreater (Box b, Cube c);
} c;
void findGreater (Box, Cube) {
  string res = (b.v > c.v) ? "Box" : ((c.v > b.v) ? "Cube" :
"Both equal.");
  cout << "Greater volume : " << res << endl;
}
int main () {
  b.accept ();
  c.accept ();
  findGreater (b, c);
  return 0;
}
```

Output:
Enter length, width, and height for the box:
3 4 3
Enter side length for the cube: 3
Greater volume: Box.

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

```
#include <iostream>
using namespace std;
class complex {
  double r, i;
public:
    void accept () {
  cout << "Enter real and imaginary parts: ";
    cin >> r >> i;
}

    void display () {
    cout << r << "+" << i << "i" << endl;
} friend Complex add(Complex c1, complex c2);
} c1, c2;
complex add( complex , complex) {
complex sum;
sum.r = c1.r + c2.r;
sum.i = c1.i + c2.i;
  return sum;
}

int main () {
  c1. accept ();
  c2. accept ();
```

```
Complex sum = add(c1, c2);
  cout << "The sum is : ";
    sum.display();
    return 0;
}
```

Output:
Enter real and imaginary parts : 3 4
Enter real and imaginary parts : 1 2.
The sum is : 4 + 6i

5. Create 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 n;
  int m[3];
public:
    void accept () {
  cout << "Enter student name : ");
    cin >> n;
  cout << "Enter marks for three subjects : ";
  cin >> m[0] >> m[1] >> m[2];
} friend void calculateAverage (student s);
} s;
void calculateAverage (student) {
  double avg = (s.m[0] + s.m[1] + s.m[2]) / 3.0;
  cout << " Student : " << avg << endl;
}
```

```
int main () {
  s accept ();
  calculateAverage(s);
  return 0;
}
```

Output:
Enter student Name : Rudra
Enter marks for three subjects: 39 41 51
Student : Rudra
Average marks: 44.

Create a class Point with Private members x and y. Write a friend function that calculates and returns the distance between two Point objects.

```
#include <iostream>
#include <cmath>
using namespace std;
class Point {
  double x, y;
public:
    void accept () {
  cout << "Enter x and y coordinates : ";
  cin >> x >> y;
  } friend double dist (Point p1, point p2);
  } p1, p2;
  double dist (Point, Point) {
  double dx = p1.x - p2.x;
  double dy = p1.y - p2.y;
  return sqrt((dx * dx) + (dy * dy));
  }

int main () {
p1.accept ();
p2.accept ();
```

```
cout << "Distance : " << dist (p1, p2) << endl ;
return 0;
}
```

Output:
Enter x and y coordinates : 2 5
Enter x and y coordinates : 5 2
Distance : 4.24264.

8. Create two classes: Bank Acc. and Audit. Bank acc. holds private balance information. Write friend function in Audit.

```
#include <iostream>
using namespace std;
class BankAccount;
class Audit {
public:
    void pBal (Bank Account);
} aud;
class BankAccount {
double b; public:
  void acept () {
    cout << "Enter bank account balance: ";
    cin >> b;
} friend function void Audit :: pBal (Bank Account);
} ba;
void Audit :: pBal (Bank Account) {
  cout << "Auditing ... Bank Account Balance : ₹"
  << ba.b << endl;
}
int main () {
ba.accept ();
```

```
acel.pBal(ba);
return 0;
}
```

Output:

Enter bank Acc. Balance : 3000

Auditing ... Bank Account Balance : ₹ 3000

```
ml(n);
```

₹ "

Exp. 04.

Q. WAP to solve the following:
1. Swap 2 no. from the same class using object as function arguments. Write swap function as member function.

→
```
#include <iostream>
using namespace std;
class num {
public:
    int n1, n2;
    void accept () {
    cout << "Enter first no. : " << endl;
    cin >> n1;
    cout << "Enter second no. : " << endl;
    cin >> n2;
}
    void display () {
    cout << "Num1 : " << n1 << endl;
    cout << "Num2 : " << n2 << endl;
}
    void swap (num) {
    int temp = n1; n1 = n2; n2 = temp;
    cout << "Swapped : \n First number : " << n1
    Second number : " << endl
}
}
int main () {
    nu.accept();
    nu.display();
    nu.swap(nn);
    return 0;
}
```

O/P:
Enter first number:
5

Enter second number:
9
Num1 : 5
Num2 : 9
Swapped:
First number : 9
Second number : 5

2. Swap two numbers from same class using concept of friend function.

→

```
~~#include <stdio.h>~~
#include<iostream>
using namespace std;
class num {
public: int n1, n2;
    void accept () {
    cout << "Enter first number : " << endl ;
    cin >> n1 ;
    cout << "Enter second number : " << endl ;
    cin >> n2 ;
}
    void display () {
    cout << " Num1 : " << n1 << endl << " Num 2 : " <<
      n2 << endl ;
}
} num ;
void swap (num) {
```

```
int temp = nu.n1;
nu.n1 = nu.n2;
nu.n2 = ~~nu.n1~~; temp;
cout << "Swapped: \n First: " << nu.n1 << "\n
    Second: " nu << endl;
}
int main() {
nu.accept();
nu.display();
swap(nu);
return 0;
}
```

O/P:

```
Enter first number:
5
Enter second number:
9
Num 1: 5
Num 2: 9
Swapped:
  first: 9
  second: 5.
```

8. Swap two number from different class using friend function.

```
#include <iostream>
using namespace std;
class num1 {
Public : int n1;
    void accept () {
  cout << "Enter first number : " << endl ;
   cin >> n1 ; }
 void display () { cout << "Num1 :" << n1 << endl; }
} nu1 ;
class num2 {
public : int n2 ;
    void accept () {
    cout << " Enter second number ! " << endl ;
    cin >> n2 ; }
  void display () { cout << "Num2 :" << n2 << endl ;
} nu2 ;
void swap (num1 , num2 ) {
    int temp = nu1 . n1 ;
    nu1 . n1 = nu2 . n2 ;
    nu2 . n2 = temp ;
    cout << "Swapped :\n First : " << nu1 . n1 << "\n Second : " << nu2 . n2 << endl ;
}

int main ()
   nu1 . accept ();
   nu2 . accept ();
   nu1 . display ();
   nu2 . display ();
   Swap (nu1 , nu2 ) ;
    return 0 ;
}
```

O/P:

Enter first number:
5
Enter second number:
9
Num 1 : 5
Num 2 : 9
Swapped:
First : 9
Second : 5.

4. Create 2 classes Result1 and Result2 which states the marks of the students. Read the value of marks for both the class object and compute avg.

→
```
#include <iostream>
using namespace std;
class Result {
public:
    int r1;
    void accept () {
    cout << "Enter first result: "<<endl;
    cin >> r1; }
} res1;
class Result2 {
public:
    int r2;
    void accept () {
    cout << "Enter second result : " <<endl;
    cin >> n2; }
} res2;
```

```
float avg(Result1, Result2){
    return (res1.r1 + res2.r2)/2;
}
int main(){
    res1.accept();
    res2.accept();
    float avg = avg(res1, res2);
    cout<<"Average: "<<avg;
    return 0;
}
```

O/P:
Enter first result:
95
Enter second result:
99
Average: 97.

5. Find the greatest number among 2 numbers from 2 different classes using friend function.

→

```
#include <iostream>
using namespace std;
class num1{
public: int n1;
    void accept(){
    cout<<"Enter first number: "<<endl;
    cin>>n1;}
};
class num2{
public: int n2;
    void accept(){
    cout<<"Enter second number: "<<endl;
    cin>>n2;}
```

```
f nu2;
int gnum (int n1, int n2) {return(n1>n2?n1:n2)
;}
int main() {
    nu1.accept();
    nu2.accept();
    int gr= gnum (nu1.n1, nu2.n2);
  cout <<"Greatest : " <<gr <<endl;
    return 0;
}
```

O/P:

Enter first number:
5

Enter second number:
9

Greatest : 9.

Q
+19

PAGE No.
DATE / /

Experiment - 05

n!=n2

a) WAP to find the summ of no. betn 1 to n using a constructor where the value of n will be passed to the constructor.

⇒
```
#include <iostream>
using namespace std;
class sum {
int n , i, sum =0;
public:
 sum ()
{
  n = 20;
}
 void calculate ()
{
 for(i=1; i<=n ;i++)
 {
     sum = sum + i ;
}
    void display ()
{
     cout <<"sum = " << sum;
}
};
int main()
{
 sum n;
 n.calculate ();
 n.display ()
 return0;
}
```

O/P:

sum = 210

(ii) using parametrized constructor ;

```
#include <iostream>
using namespace std;
class sum
{
    int n, i, sum=0;
  public:
      sum(int num)
  {
      n = num;
  }
    void calculate ()
  {
    for (i=1 ; i<=n ; i++)
  {
    sum = sum+i ;
  }
  }
    void display ()
  {
    cout <<"sum = " << sum ;
  }
};
  int main ()
  {
    sum n (20) ;
    n. calculate ;
    n. display ;
    return 0;
```

O/P :

Sum = 210

6) WAP is declare a class 'student' having data members as name and percentage. Write a constructor to initialise these data members.

```
#include <iostream>
using namespace std;
class student
{
int per;
string per;
public:
student ()
{
name = "Rudra";
per = "93";
}
void display ()
{
cout << "Name : " << name;
cout << "Percentage : " << per;
}
};
int main () {
student s;
s. display ()
return 0;
}
```

O/P:

Name : Rudra
Percentage : 93.

(ii) Parametised Constructor.

```
#include <~~stdi~~iostream>
    using namespace std;
    student
{
    int per;
    string name;
    public:
      student (string n, int percent)
{
      name = n;
      per = percent;
}
    void display () {
    cout << "Name = " << name;
    cout << "Percentage = " << per;
}
};
  int main()
{
  student s ("Rudra", 77)
   s. display ();
    return 0;
}
```

O/P:

Name: Rudra
Percentage = 77.

c) Define a class 'college' members variables as roll-no, name, course. WAP using constructor with default value as "Computer Engineering" for course. Accept this data for two object of a class and display the data.

```
= #include <iostream>
    using namespace std;
  class college
  {
    private:
    string course_name, stud_name;
    int roll_no;
    public:
    college ()
  {
      roll_no = 57;
    stud_name = "Rudra";
    course_name = "AIDS";
  }
    void display ()
  {
    cout << "Roll no:" << roll_no << endl;
    cout << " Student name = " << stud_name << endl;
    cout << "Course Name :" << course_name << endl;
  }
};
    int main ()
  {
      college u;
      u.display();
      return 0;
  }
```

O/P:
Roll no: 77.
Student Name: Rudra.
Course Name: AIDS.

Pu
11111

PAGE No.
DATE / /

Experiment - 08.

1) Single inheritance.
Create a base class called person with attributes name and age. Derive a class student with attributes from person that adds an attribute roll no.

```
#include<iostream>
using namespace std;
class person
{
    protected:
    string name = "Rudra";
    int age = 17;
};
class student : protected person.
{
    int roll = 77;
    public:
      void display ()
{
  cout <<" Roll no: " << roll <<"\n name: " << name <<"\nage"
  << age;
}
};
int main()
{
    student s;
    s1.display();
      return 0;
  }
```

O/P:

2) Multiple inheritance.
Create 2 class as academics and sports. Write a function to calculate the.

```
#include <iostream>
using namespace std;
class academics.
{
 Protected :
   int marks = 90 ;
};
class sports
{
   protected:
   int sports score = 60;
};
class result : protected academics , protected sports
{
   int total score = marks + sports score ;
   public :
   void calculate ()
   {
   cout << "total score :" << totalscore ;
   }
};
int main()
{
   result r1;
   r1. calculate.
```

O/P :
total score : 150

3) Multilevel inheritance.

Create a class vehicle with attributes like brand and model. Derive a class car from vehicle which adds an attributes type.

```
#include <iostream>
using namespace std;
class vehicle
{
    protected:
      string brand = "bmw";
      string model = "x";
};
  protected: class car : protected vehicle
{
    protected:
    string type = "SUV";
};
class electricar : protected car
{
    int battery capacity = 50;
    public:
      void display ()
{
    cout <<"\nbrand is : " << brand;
    cout <<"\n model is : " << model;
    cout <<"\n type is : " << type;
    cout <<"\n battery capacity : " << battery capacity;
}
};
```

```
int main ()
{
  electricar e1;
  e1.display ()
  return 0;
}
```

O/P:

brand : BMW.
mode : x1
type is: SUV
battery capacity : 80.

4) Hierachial Inheritance.

```
#include <iostream>
using namespace std;
class employee
{
  protected :
    int emp ID = 5;
    string name = "Rudra";
};
class manager : protected emp
{
  String dept = 'poly';
  public:
   void display ()
  {
    cout << "\nempid : " << empID ;
    cout << " \n name : " << name ;
    cout << " \n dept name : " << dept;
  }
};
```

```
class developer : protected employee
{
    string programming language = "CPP";
    public:
     void display ()
{
    cout << "\n empID : " << empID;
    cout << "\n name: " << name;
    cout << "\n programming language : " << programing.
}
};
int main()
{
    manager m1;
    developer d1;
    m1.display ();
    d1.display ();
    return 0;
}.
```

O/P:

empID is 5
name is Rudra.
dept name is poly.
programming lang. is CPP.

- virtual base class.

WAP to Implement following Inheritance

class: clg. student.
data mem: student id.

class: test.
data mem: percentage.

class: sports.
data mem: grade.

class: result.

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

class college student {
protected:
    int student_id;
    int clg_id;
public:
    void collect data(){
    cout << "enter student ID:" ;
    cin >> student id;
    cout << "enter college id :" ;
    cin >> college_code ;
}

    void display () {
    cout << "\n student id :" << student_id ;
    cout << "\n college id :" << college_code ;
}
};
```

```
class Test : virtual public collegestudent {
protected:
        float percentage;
public:
        void getdata() {
        cout<<" enter percentage: ";
        cin >> percentage;
    }
    Void display test data() {
    cout<<"\n percentage : "<<percentage;
    }
};
class sports : virtual public collegestudent {
protected:
        char grade;
public:
        ~~void getsports data.~~

        void displaysports data () {
        cout<<"\n sports grade : "<<grade;
    }
};
Class Result: public test, public sports {
public:
        void getdata () {
            collect data();
            get test data ();
            get sports data ();
    }
    void display data () {
        cout << "result: \n";
        display ();
```

Experiment - 7

a) WAP using function overloading to calculate the area of a laboratory and area of class room (rectangular).

→
```
#include<iostream>
using namespace std;
class Area
{
    private:
    float l, b;
    public:
    void area (float l)
{
    float area;
    area = l * l
cout << "Area of square :" << area << endl;
}
void area (float l, float b)
{
    float area;
    area = l * b;
cout << "Area of rectangles : " << area;
}
};
int main ()
{
    Area a1;
    a1.area(8);
    a1.area(4, 12);
    return 0;
```

O/P : Area of Square : 64.
Area of rectangles : 48.

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

→
```
# include <iostream>
using namespace std;
{
    private :
int a, b, c, d, e, f, g, h, i, j;
float k, l, m, n, o;
public :
void sum (float k, float l, float m, float n,
  float o)
{
    float sum ;
    sum = k + l + m + n + o;
cout << "sum of 5 floating no. " << sum << endl;
}
void sum (int a, int b, int c, int d, int e, int f,
 int g, int h, int i, int j)
{
    int sum;
    sum = a + b + c + d + e + f + g + h + i + j;
    cout << " sum of 10 integers " << sum;
}
};
int main()
{
```

```
int main()
{
    sum s1;
    s1.sum(~~1,2,3,~~ (1.2, 3.5, 5.6, 8.4, 1.5);
    s1.sum(1, 2, 3, 4, 5, 6, 7, 8, 9, 10); 97
    return 0;
}
```

o/p: sum of 5 float numbers : 20.2.
sum of 10 integers : 55.

c. WAP to implement unary-operator when used with the obj so that the numeric data member of the class

```
#include <iostream>
using namespace std;
class Number
{
    private int x;
    public void accept ()
    {
        cout << "Enter a number : ";
        cin >> x;

    void operator - ()
    {
        x = -x;
    }
    void display ()
    {
        cout << "Negated number : " << x << endl;
```

```
};
int main()
{
    Number n1;
    n1.accept();
    -n1;
    n1.display();
    return 0;
}
```

O/P:

Enter a number : 5
Negated number : -5

d.

```
#include <iostream>
using namespace std;
class Number
{
    private int x;
    public void accept()
    {
        cout << "Enter a number";
        cin >> x;
    }
    temp = x;
    void operator ++()
    {
        x = temp;
    }
    void operator ++(int)
    {
        x = x++;
    }
    void display()
    {
        cout << " The number is : " << x << endl;
    }
    void display ()
    {
        cout << "(post) The number is : " << x;
    }
};
int main()
{
    Number n1;
    n1.accept();
    ++n1;
    n1.display();
    n1.reset();
    n++;
    n1.display2();
    return 0;
}
```

O/P: Enter a number : 5

4/11

O/P:

Concatenated string:
xyzpqr

6)

```
#Include<string>
#Include <iostream>
using namespace std;
Class Login {
protected : string name password;
Public:
        virtual void accept(){
        cout << "Enter name:";
        cin >> name;
        cout << "password:";
        cin >> password:";
    }
};
class EmailLogin : Public Login {
        string email;
    public:
        void accept () override {
        Login : !accept ()
        cout << "Enter email:";
        cin >> email;
    }

    void display () {
        cout << "\n Membership Login Details \n";
        cout << "Name << \n password " <<password"<<
"Membership ID"<<membership ID<<endl;
    }
};
int main()
{
    EmailLogin e;
    Membership m;
    e.accept();
    m.accept();
    e.display();
    return 0;
}
```

```
cout<<" Name:"<<name<<"password:"<<
password <<"Email" <<email <<endl;
}
};
class MembershipLogin: Public Login {
    string membership ID;
  public:
        void accept() override {
        login : accept();
        cout<<"Enter membership ID:";
        cin >> membership ID";
    }
    void display () {
        cout << "\n Membership Login Details \n";
        cout << "Name << \n password " <<password"<<
"Membership ID"<<membership ID<<endl;
    }
};
int main()
{
```

4/11

O/P:

Enter Name: Rudra.
Enter password: 123
Enter email: abc@gmail.com.
Enter name: Rudra.
password: 745
Enter membership ID: 5745
--- Email login Details ---
Name: Rudra
Password: 123
email: abc@gmail.com.
--- Membership Login Details ---
Name: Rudra.
password: 745
membership ID: 5745.

Experiment - 09.

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

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    fstream first_file;
    fstream second_file;
    first_file.open("First.text", ios::in);
    if(!first_file)
    {
    cout << "Error opening second.text!" << endl;
    return 1;
    }
    char ch;
    while(first.file.getch)
    {
        second_file put(ch); }
        first_file.close();
        second_file.close();
    cout << "file copied successful!" << endl;
    return 0;
}
```

6) WAP to count digits and spaces using file handling.

```
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;
int main()
{
fstream new_file;
new_file.open("First.txt", ios::in);
if(!new_file)
{
cout << "Error Occurred opening!"
<<endl;
return 1;
}
int digit_count = 0;
int space_count = 0;
char ch;
while(new_file.get(ch)){
if(digit(ch))
{
digits.count++;
}
if(kspace(ch)){
space_count++;
}
new_file.close();
cout << "Number of digits :" <<
```

2) WAP to count words using file handling

```
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;
{
fstream new_file;
new_file.open("First.txt", ios::in);
(!new_file)
{
cout << "Error Occurred!" <<endl;
return 0;
}
char ch;
int words_count = 0;
bool inWord = false;
while(new_file.get(ch))
{
if(isspace(ch))
{
inWord = false; }
else if(!inWord)
{
words_count++;
inword = true;
}
}
new_file.close()
cout << "Number of words " << words_count
<<endl;
```

d) WAP to count occurrance of given word

```
#include <iostream>
# include <fstream>
# include < string>
using namespace std;
int main()
{
    fstream new_file;
  new_file open ("First.txt", ios::in);
   if(!new_file)
   {
     cout<<"Error in opening First.txt!"
     <<endl;
     return 1;
   }
  string target = "Heaven";
   string word;
   int count = 0;
   while (new_file >>word)
  {
     if( word = target)
     {
       count ++ ;
     }
     new.file.close();
  cout<<"The word ";<<target << "\"Ocurred"
  << count<<"times" <<endl;
  return 0;
  }
```

Exp. 10.

Q1. WAP to find sum of array elements using function template

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

template <class T> T find_sum (T arr[ ], int n)
{
    T sum = 0;
    for(int i = 0; i < n; i++)
   {
      sum += arr[i];
   }
    return sum; }
  int main ()
{
    int size = 10;
    int intArr[size] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
float float Arr[size] = { 1.1, 2.2, 3.2, 4.4, 5.5,
   6.6, 7.7, 8.8, 9.9, 10.10 }
double double Arr[size] = { 1.11, 2.22, 3.33, 4.44,
   5.55, 6.66, 7.77, 8.88, 9.99, 10.10};
cout << "Sum of integer array = " << findsum
    (int Arr, size) << endl;
cout << "sum of float array = " << findsum
    (float Arr size);
cout << " sum of double Arr = " << findsum(double
   Arr size) << endl;
```

O/P:

sum of integer array = 55
sum of float array = 59.6
sum of double array = 60.05

Q.2 WAP to square function using template specialization

```
#include <bits/stdc++h>
using namespace std;
template <class T>
T square (T num){
    return num * num;
}
template < >
string square <string> (string str){
    return str + str; }
int main ()
{
    int intNum = 5;
    string str = "Hello";

cout << "Square of integer "<<intNum<<"="
<<square (intNum)<<endl;
cout << "square of string |" <<str << "|" ="
<< square(str) <<endl;
return 0;
}
```

O/P:

Square of integer = 25
square of string = HelloHello

3. WAP to build a simple calculator using class template

```
#include <bits/stdc++h>
using namespace std;

template <class>
class Calculator {
  private:
    T num1, T num 2;
  public:
    Calculator (T n1, T n2){
      num1 = n1;
      num 2 = n2;
    }
  T add() { return num1 + num 2; }
  T subtract () { return num1 - num2; }
  T multiply () { return num1 * num2; }
  T divide () {
      if (num2) = 0)
      return num1 / num 2;
    else {
    cout<< "Error! division by zero"<<endl;
    return 0;
    }
}

  void displayResult () {
  cout<< "Numbers:"<<num1 <<"and"<<num
2<<endl;
```

```
cout << "Numbers : "<<num1 << "and" <<num2 << endl
cout << "Addition = " << add() << endl ;
cout << " Substraction " << subtract () <<endl;
cout << " multiplication" << multiply() << endl;
cout << " Division = " << divide()<<endl;
}
};
int main () {
cout<<"simple Calculator using class template \n";
calculator <int > int Cal c(10, 5);
cout << "\n Integer calculator --- \n" ;
int calc. display Results();
calculator <float> float calc (10.5, 2.5);
Cout << "\n --- float Calculator --- \n";
float calc. displayResult ();
return 0;
}
```

O/P :

```
--- Integer Calculator: ---
Numbers: 10 and 5
Addition = 15
Subtraction = 5
Multiplication = 50
Division = 2
--- float Calculator ---
Numbers: 10.5 and 2.5
Addition = 13
Subtraction = 8
Multiplication = 26.25
Division = 4.3.
```

4. WAP to implement push and pop methods from stack using class template.

```
#include<bits/stdc++h>
using namespace std;
#define size 5
template <class T>
class stack {
private :
  T arr [size];
  int top ;
public :
  stack () {
  top = -1 ;
  }
void push (T value) {
  if (top == size - 1) {
    cout << "stack overflow / cannot push" <<
      value << endl;
  }
  else {
    arr [++top] = value ;
  cout << value << " Pushed into stack" << endl;
  }
}
void pop () {
  if (top = = -1) {
  cout << "stack is empty" << endl; }
  else {
  cout << arr [top--] << "popped from stack" <<
```

```
}
}
void display () {
 if (top == -1) {
cout << "stack is empty " << endl; }
 else {
 cout << "stack is empty " << endl; }
else {
cout << "stack elements : " ;
 for (int i = top; i >= 0; i--)
  cout << arr [i] << " " ; }
  cout << endl ;
  }
 }
};
 int main () {
 cout << "stack implementation using class template
   \n" ;
stack < int > int stack ;
cout << "stack implementation using class template"

int stack . push (10) ;
int stack . push (20) ;
int stack . push (30) ;
int stack . display () ;
int stack . pop () ;
int stack . display () ;

stack <string> str stack ;
cout << "\n --- string stack operations --- \n" ;
str stack . push ("Apple") ;
str stack display () ;
str . stack pop () ;
str . stack display () ;
 return 0 ;
}
```

O/P:

```
--- Integer stack operations ---
10 pushed into stack.
20 pushed into stack.
30 pushed int stack.
stack elements : 30  20  10.
```

Exp. 11

1) Without itrator :-

→
```
#include <iostream>
# include < vector >
# include <cctype>
using namespace std;
int main () {
  ~~vec int main()~~
  vector <char> v(10);
  unsigned int i ;
  cout << "size : " << vsize () << " " << endl;
  for (i=0; i<10 ; i++) { v[i] = i + 'i'; }
cout << "Current content : \n";
for (i=0 ; i < v . size () ; i++) { cout << v [i]
<< " " ; }
cout << "\n\n"; cout << "expanding vector:\n";
for(i = 0 ; i<10 ; i++){ v . push_back
 (i+10 + 'a');
cout << "size now = " << v . size () << endl;
cout << "Current content : \n";
for (i=0; i< v . size () ; i++) { cout <<
v[i] << " ";
for(i=0 ; i < v . size () ; i++) { v[i] << " ";
cout << "\n\n";
for (i=0 ; i < v size () ; i++) { v[i] = toupper
(v[i]); }
cout << "Modified contents :\n";
for (i=0; i < v . size () ; i++ ) { cout << v
[i] << " " ; }
cout << endl;
return 0;
}
```

Q.2. With iterator.

→
```
#include <iostream>
# include <vector>
using namespace std;
int main () {
  vector <char> v(10);
  vector <char> :: iterator p;
  int i ;
  p = v . begin ()
    i = 0 ;
while (p != v . end() ) {
  *p = i + 'a';
    p++ ;
    i++ ;
}
cout << " original content :\n";
  p = v . begin ()
while (p != v . end () ) {
cout << * p << " ";
    p++ ;
}
```

Ph
11/11

Exp. 18.

1) Implement Stack.

→
```
#include <bits/stdc++h>
using namespace std;
int main() {
stack <char> cars;
cars.push("BMW"); cars.push("Audi");
cars.push("Mercedes"); car.push("Ferrari");
cout << " Top element is : " << cars.top() << endl;
cout << "size of stack is : " << cars.size() << endl;
cars.pop(); cars.pop();
while (!cars.empty()) {
cout << " Element in stack are : " << cars.top() << endl;
car.pop();
}
return 0;
}
```

o/p :-

Top element : Ferrari
size of stack is : 4
Element in stack are : Audi
Element in stack are : BMW.

2. Implement Queue.

```
#include <bits/stdc++h>
using namespace std;
int main() {
queue <int> ;
age.push(21); age.push(22); age.push(23); age(24);
cout << " front element is : " << age.front() << endl;
cout << " Back element is : " << age.back() << endl;
age.pop(); age.pop();
while (!age.empty())
{
cout << " Element in queue are : " << front() << endl;
age.pop(); }
return 0;
}
```

o/p :

front Element is : 21
Back element is : 24
Element are : 23
Elements are : 24.