EXPERIMENT: 1

A. Write a C++ program to declare a class student having data members as Roll_No & Name. Accept and display data for single ~~content~~ student.

→
```
# include <iostream>
Class student
{
 int Roll_No;
 str name;
 Public:
 void accept()
 {
  cout << " Enter name and rollno: ";
   cin >> name >> Roll_No;

 void display()
 {
  cout << "Name: " << name;
  cout << "Rollno:" << Roll_No;
  };

int main()
{
 student S1;

 S1. accept();
 S1. display();
return 0;
}
```

OUTPUT

Enter name and rollno of student:
ADWITA 27

name of student: ADWITA
Rollno : 27

B. Write C++ code to create a class book having data members as book_name, b_price, b_pages. Accept the data for two books and display the name of book having greater price.

→
```
# include <iostream>
Class book
{ Public:
    str b_name;
    int b_price;
    int b_pages;

 Public:
 void accept()
 { cout << " Enter book name, book price & no of pages: ";
   cin >> b_name >> b_price >> b_pages;
 }

 void display()
{ cout << " Book name : " << b_name;
   cout << " Book price: " << b_price;
   cout << " No of pages: " << b_pages;
};

 int main()
{ bBook B1, B2;
    B1. accept();
    B2. accept();
```

```
if ( B1.b_price > B2.b_price )
{
   B1. display ();
}
else
{
   B2. display ();
}
return 0;
}
```

OUTPUT

```
Enter book name, book price and book pages : Life
250
500.

Enter book name, book price and book pages : Tree
300
700

Name of book: Tree
Book Price : 300
No of pages : 700.
```

c. Write a program to declare class "time", accept time in HH:MM:SS format, convert it into total seconds and display them.

→

```
# include <iostream>

class Time {
public:
int hrs;
int mins;
int secs;
public:
void accept () {
 cout << "Enter time in hrs: mins: secs";
 cin >> hrs >> mins >> secs;
}

void display () {
int total_time = (hrs * 3600) + (mins * 60) + secs;
 cout << "Total time in seconds :" << total_time;
}
};

int main() {
 Time t1;

 t1. accept ();
 t1. display();
 return 0;
}
```

OUTPUT

```
Enter time in hrs: mins: secs 4
                              40
                              35
Total time in seconds: 16835.
```

15/7/25

Page No.
Date

EXPERIMENT : 2

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

```
# include <iostream>
# include <string>
 class city
 {
  public:
   string name;
   int population;

  public:
  void accept()
  {
    cout << "Enter the name of city: ";
    cin >> name;

    cout << "Enter population";
    cin >> population;
  }
  void display()
  {
    cout << "\n Name of city: " << name;
    Cout << "\n Population : " << population;
  }
  };
```

```
int main() {
  city c[5];
  int i;

 for( i=0; i<5; i++)
{
   c[i].accept();
}

int max = 0;
for(i=0 ; i<5 ; i++){
   if (c[max].population < c[i].population)
  {
     max = i;
   }
 }

 cout << "In city with maximum population:";

 c[max].display();
 return 0;
  }
```

OUTPUT

```
Enter name of city: A
Enter population : 789.
" - " : B
656

C
7979

D
879867

E
878966

City with maximum population: D
Population of city: 879867.
```

B. Write a program to declare a class

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

class Account {
  public:
    int acc_no;
    float bal;

  void getdata()
{
  cout << "Enter account number: ";
   cin >> acc_no;
  cout << "Enter balance;
   cin >> bal;
}

  void display()
{
   cout << "\n Account Number: " << acc_no;
   cout << "Balance: " << bal;
}
};
```

```
int main()
{
  account a[10];

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

 for (int i=0; i<10 ; i++)
 {
    if (a[i].bal >= 5000) {
        a[i].bal = a[i].bal + a[i].bal * 0.1;
        a[i].display();
    }

return 0;
}
```

C.

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

class staff {
public:
    string name;
    string post;

public:
    void accept() {

        cout << "Enter Staff name:";
        cin >> name;
        cout << "Enter post:";
        cin >> post;
    }

    void display() {
        cout << "\n Staff name:" << name;
    }
};
```

```
int main()
{
    staff s[5];
    for (int i=0; i<5; i++)
    {
        s[i].accept();
    }
    for (int i=0; i<5; i++)
    {
        if (s[i].post == "HOD") {
            s[i].display();
        }
    }
    return 0;
}
```

24/7/25

EXPERIMENT: 3

A.

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

class book {
   string book_name;
   string author_name;
   int book_price;

   public :
   void accept () {
    cout << "Enter book name :";
    getline (cin, this -> book_name);

    cout << "Enter author name :";
    getline (cin, this -> author_name);

    cout << "Enter book price :";
    getline (cin,
    cin >> book_price ;
}
```

```
void display () {
    cout << "\n Name of book : " << book_name;
    cout << "\n Name of author: " << author_name;
    cout << "\n Book price : " << book_price;
}
};

int main () {
   book * b1 = new book;
   b1 -> accept ();
   b1 -> display ();

   return 0;
}
```

B.

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

class STUDENT {

    int roll_no;
    float percentage;

public:
 void accept() {
    cout << "Enter student roll_no:";
     cin >> this -> rollno;
    cout << "Enter student percentage:";
     cin >> this -> percentage;
 }

 void display() {
     cout << "\n student roll no:" << roll_no;
     cout << "\n student percentage:" << percentage;
 }
};
```

```
int main() {
  STUDENT S1;
     S1.accept();
     S1.display();

  return 0;
  }
```

C.

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

class student {
 public:
  class marks {
     public:
      int m1, m2;

      void getinput() {
        cout << "Enter marks:";
        cin >> m1 >> m2;
      }
```

```
Float percent(){
        int total = m1 + m2;
        return total/2.0;
    }
};

int main()
{
  student::marks marks;
  marks.getinput();

  Float percentage = marks.getpercent();

  cout << "Percentage = " << Percentage << "%";

  return 0;
}
```

Enter marks: 50
60
Percentage = 55%

Practice Questions.

1) Create two classes, class A and class B, each with a 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 classA {
private:
   int numA;
public:
   classA (int a) : numA(a) {}
   friend int sum (const classA &, const classB &);
};

class classB {
private:
  int numB;
public:
  classB (int b) : numB(b) {}
  friend int sum (const classA &, const classB &);
};

int sum (const classA & a, const classB & b){
    return a.numA + b.numB;
}
```

```
int main()
{
  ClassA objA(30);
  ClassB objB(70);

  cout << "Sum = " << sum(objA, objB) << endl;
  return 0;
}
```

(2)

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

class Number {
private:
  int value;
public:
  Number(int val) : value(val) {}
  void show() {
    cout << "Value: " << value << endl;
  }
  friend void swapNumbers(Number&, Number&);
};
```

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

int main() {
  Number num1(60);
  Number num2(40);

  cout << "Before swap:" << endl;
  num1.show();
  num2.show();

  swapNumbers(num1, num2);

  cout << "After swap:" << endl;
  num1.show();
  num2.show();
  return 0;
}
```

3)

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

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

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

void findGreater(const Box& b, const Cube& c)
{
  if (b.volume > c.volume)
    cout << "Box has greater volume" << endl;
  else if (b.volume < c.volume)
    cout << "Cube has greater volume" << endl;
  else
    cout << "Both have equal volume";
}
```

```
int main() {
  Box box1(20);
  Cube cube1(80);
  findGreater(box1, cube1);
  return 0;
}
```

(4)

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

class complex {
private:
  int real;
  int imag;
public:
  complex(int r=0, int i=0): real(r), imag(i) {}
  void show() const {
    cout << real << " + " << imag << "i" << endl;
  }
  friend complex add(const complex&, const complex&);
};

complex add(const complex& c1, const complex& c2)
{ return complex(c1.real + c2.real, c1.imag
                 + c2.imag);
}
```

```
int main()
{ complex a(7, 8);
  complex b(2, 3);

  complex c = add(a, b);
  cout << "Sum of complex numbers: ";
   c.show();
  return 0;
}
```

(5)

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

class student {
 private:
   string name;
   int mark1, mark2, mark3;
 public:
   student(string n, int m1, int m2, int m3) :
          name(n), mark1(m1), mark2(m2), mark3(m3)
          {}
  friend void calculateAverage(const student &);
};

void CalculateAverage(const student& s) {
```

```
float avg = (s.mark1 + s.mark2 + s.mark3) / 3.0f;
   cout << "Student Name: " << s.name << endl;
   cout << "Average Marks: " << avg << endl;
}
int main()
{ Student s1("Adwita", 90, 98, 97);
   calculateAverage(s1);
   return 0;
}
```

(6)

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

class Alpha {
private:
   int a;
public:
   Alpha(int val) : a(val) {}
   friend void totalSum(const Alpha&, const Beta&,
                        const Gama&);

};

class Beta {
private:
   int b;
public:
```

```
    Beta(int val): b(val) {}
    friend void totalSum(const Alpha&, const Beta&,
                         const Gama& );
};

class Gama {
private:
    int c;
public:
    Gama(int val): c(val) {}
    friend void totalSum(const Alpha&, const Beta&,
                         const Gama& );
};

void totalSum(const Alpha& x, const Beta& y,
              const Gama& z)
{   int sum = x.a + y.b + z.c;
    cout << "Sum of all values: " << sum << endl;
}

int main()
{   Alpha obj1(50);
    Beta obj2(30);
    Gama obj3(20);

totalSum(obj1, obj2, obj3);
return 0;
}
```

(7)

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

class Point {
private:
    int x, y;
public:
    Point(int xVal, int yVal): x(xVal), y(yVal) {}

    friend double calculateDistance(const Point&,
                                    const Point&);
};

double calculateDistance(const Point& p1, const Point& p2){
    int dx = p2.x - p1.x;
    int dy = p2.y - p1.y;
    return sqrt(dx * dx + dy * dy);
}

int main() {
    Point p1(3,4);
    Point p2(7,1);

double distance = calculateDistance(p1, p2);
    cout << "Distance between points: " << distance << endl;
    return 0;
}
```

(8)

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

class BankAccount {
  private:
    int balance;
  public:
    BankAccount(int b): balance(b) {}
    friend void auditBalance(const BankAccount&,
                             const Audit&);
};

class Audit {
public:
    void startAudit(const BankAccount& acc) const {
        auditBalance(acc, *this);
    }
    friend void auditBalance(const BackAccount&, const
                             Audit&);
};

void auditBalance(const BankAccount& acc, const
                  Audit&)
{   cout << "Auditing Account....\n";
    cout << "Balance: $ " << acc.balance;
}
```

```
int main() {
    BankAccount myAcc(1200000);
    Audit auditor;
    auditor.startAudit(myAcc);
    return 0;
}
```

EXPERIMENT: 4

(1)

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

class number {
  int a,b;
public:
  void Data (int x, int y) {
        a = x ;
        b = y ;
  }
  void display ()
  {
   cout<<"a = "<<a<<"b="<<b<<endl;
  }
void swap (number &obj)
  int temp ;
  temp = a ;
   a = obj.a ;
  obj.a = temp;

  temp=b ;
  b = obj.b;
  obj.b = temp;
  }
};
```

```
int main()
{ Number obj1 , obj2 ;
  obj1.Data (10, 20);
  obj2.Data (30,40);

  cout << "obj1 :";
  obj1.display();
  cout << "obj2 :";
  obj2.display();

  obj1.swap(obj2);

  cout << "\n After swap :"<<endl;
  cout << "obj1 :";
  obj1.display();
  cout << "obj2 :";
  obj2.display();
  return 0;
}
```

(2)

```
#include <iostream>

class number {
  int value;
public:
  void value ( int v)
  {   value = v;
    }
  void display () {
    cout << "Value: " << value << endl;
    }
  friend void swapvalue (number & n1, number & n2);

};

void swapvalues (number & n1 , number & n2 ) {
      int temp  = n1. value;
      n1. value  = n2. value;
      n2. value  = temp;
    }
int main () {
  number A, B;
      A. value ( 10);
      B. value (20);
      A. display ();
      B. display ();
      swapvalues (A, B);
      cout << " New values: " << endl;
      A. display ();
      B. display ();
      return 0;
    }
```

(3) #

```
class classA {
   int numA;
public:
   void values (int a) {
        numA = a;
    }
   void display () {
   cout << " classA value = " << numA << endl;
   }
friend void swapvalues (classA & , classB & );
};

class classB {
   int numB;
public:
   void values ( int b) {
        numB = b;
}
```

```
    void display() {
        cout << " class B value = " << numB << endl;
    }
    friend void swapvalues(classA &, classB &);
};

void swapvalues(classA &a, classB &b) {
    int temp = a.numA;
    a.numA = b.numB;
    b.numB = temp;
}

int main() {
    classA A;
    classB B;

    A.value(20);
    B.value(30);
    cout << "Values: ";
    A.display();
    B.display();
    swapvalues(A, B);
    cout << "swapped values: ";
    A.display();
    B.display();
    return 0;
}
```

(3)

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

class result1 {
    float marks1;
public:
    void readmarks() {
        cout << "Enter marks for result 1: ";
        cin >> marks1;
    }
    friend float computeAverage(result1 r1, result2 r2);
};

class result2 {
    float marks2;
public:
    void readmarks() {
        cout << "Enter marks for result 2: ";
        cin >> marks2;
    }
    friend float computeAverage(result1 r1, result2 r2);
};

float computeAverage(result1 r1, result2 r2) {
    return (r1.marks1 + r2.marks2) / 2;
}
```

```
int main () {
    Result 1   A1 ;
    Result 2   A2 ;

    A1 . readmarks () ;
    A 2 . readmarks () ;

float avag = computeAverage (A1, A2) ;
cout << "\n Average of two results : " << avg ;
 return 0 ;
 }
```

5)

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

class classA {
  int numA ;
  public:
  void accept (int a) {
    numA = a ;
  }
  void display () {
  cout << " ClassA Number : " << numA ;
}
friend void findgreatest (classA , classB) ;
};
```

```
class classB {
  int numB ;
  public:
   void accept (int b) {
      numB = b ;
   }
   void display () {
    cout << " ClassB number : " << numB ;
   }
friend void find greatest (classA , classB) ;
};

void findgreatest ( classA a , classB b) {
   if (a.numA > b.numB) {
  cout << " Greatest number is from class A " << a.numA ;
   }
else if ( b.numB >> a.numA) {
 cout << " Greatest number is from class B " << b.numB ;
  }
else {
cout << " Both are equal " ;
  } }
int main () {
   class A  A ;
   class B  B ;

   A . accept (35) ;
   B . accept (40) ;
   A . display () ;
   B . display () ;
```

```
findgreatest (A, B);
return 0;
}
```

12/8/25

* EXPERIMENT

Page No.
Date

EXPERIMENT : 5

1) Write a program to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

default constructor

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

class addition {
    int n ;
    int sum ;
    public :
    addition () {
      n = 10 ;
      sum = 0 ;
     for ( int i=1 ; i<=to n ; i++)
      {  sum += i ;
        }
    }
  void display () {
     cout <<" Sum of first "<< n <<" numbers :" <<
          sum ;
    }
};
int main () {
    addition a ;
    a. display() ;
  return 0;
}
```

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

class addition {
   int n ;
   int sum ;

   public :
   addition (int no) {
      n = no ;
      sum = 0 ;
      for (int i = 1 ; i <= n ; i++)
      {     sum += i ;
        }
   }
   void display () {
     cout << "Sum of first " << n << " numbers is: " <<
          sum ;
     }
};
int main () {
   addition a (5) ;
    a. display () ;

   return 0 ;
   }
```

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

class sum
{    int n ;
     int s = 0 ;
     public :
     sum()
   {  n = 4 ;
    }
   sum (sum & obj)
{
     n = obj.n ;
     int i ;
     for(i = 1 ; i <= n ; i++)
     {
        s += i ;
      }
     cout << "Sum = " << s ;
   }
};
 int main ()
{    sum s1 ;
     sum s2 (s1) ;
     return 0 ;
   }
```

2) WAP to declare a 'class student" having data members name & percentage. Write a constructor to initialize these data members. Accept and display data for one student.

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

class student {
  int percentage;
  string name;

  public:
  student () {
     name = "ADWITA";
     percentage = 90;
  }

  void display () {
   cout << "Name of student :" << name << endl;
   cout << "Percentage of student:" << percentage;
  }
};
int main() {
    student S1;
    S1. display();
    return 0;
}
```

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

class student {
   float percentage;
   string name;

   public:
   student (string n, float p) {
       name = n;
       percentage = p;
   }

   void display(){
    cout << " Name of student :" << name << endl;
    cout << "Percentage of student :" << percentage;
   }
};
int main(){
  student S1 ("ADWITA", 97.6);
   S1. display();

   return 0;
}
```

```
#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 << "Name:" << name << endl;
  cout << "Percentage:" << percentage << endl;
 }
};
int main()
{
    string n;
    float p;
    cout << "Enter Student name:";
    getline (cin,n);
    cout << "Enter Percentage:";
    cin >> p;
    student s1(n,p);
    student s2(s1);
    cout << "\n Student data" << endl;
    s2.display();
    return 0;
}
```

Output

```
Student Data.
Name: ABC
Percentage: 98%.
```

3)

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

class college {
   int rollno;
   string name;
   string course;
public:
   college (int r, string n, string c = "Computer Science")
   {
      rollno = r;
      name = n;
      course = c;
   }
void display()
   {
      cout << "Roll No : " << rollno << endl;
      cout << "Name : " << name << endl;
      cout << "Course : " << course << endl;
   }
};
int main()
{ student s1, s2;
  int r1, r2;
  string n1, n2;
  cout << " Enter rollno, name for first student";
  cin >> r1 >> n1;
  cout << " Enter rollno, name for second student";
  cin >> r2 >> n2;

  college (r1, n1);
  college (r2, n2);

  cout << " Details of students :";
  s1. display ();
  s2. display ();
  return 0;
}
```

4)

```
#include <iostream>
using namespace std;
class number {
    int x, y;
    public:
    number (int a)
    {   x = a;
        y = 0;
    }
    number (int a, int b)
    {   x = a;
        y = b;
    }
    void display()
    {
    cout << "x = " << x << ", y = " << y << endl;
    }
};
int main()
{   number n1(5);
    number n2(5,10);
    n1. display();
    n2. display();
    return 0;
}
```

OUTPUT

x = 5 , y = 0
x = 5 , y = 10

16/9/25

# EXPERIMENT - 6.

1) Single inheritance.

```
#include <iostream>
using namespace std;
class Person{
 public:
 string name;
  int age;
};
class student : public Person
{  int rollno;
   public:
   void accept(){
    cout << "Enter name, age, rollno: "<<endl;
    cin >> name >> age >> rollno;
   }
   void display(){
    cout << "Name : " << name << endl;
    cout << " Age : " << age << endl;
    cout << " ROLLNO : " << rollno << endl;
   }
};
 int main() {
   student s1;
   s1.accept();
```

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

OUTPUT

```
Enter name, age, rollno :
   ABC   17   27
Name : ABC
Age : 17
ROLLNO : 27
```

2) Multiple Inheritance.

```
#include <iostream>
using namespace std;
class academic
{ public :
   int marks;
  };

class Sports
{ public:
   int score;
  };

class Result : public academic, public Sports
{
  int result;
  public:
  void getdata()
  { cout << " Enter Academic marks: ";
    cin >> marks;
    cout << " Enter Sports score: ";
    cin >> score;
  }

  void calculate()
  {
     result = marks + score;
     cout << " Result : " << result << endl;
   }
};
int main()
{ Result r1;
    r1.getdata();
    r1.calculate();
    return 0;
  }
```

OUTPUT

```
Enter Academic marks: 10
Enter Sports score: 8
Result = 18
```

3) Multilevel inheritance.

```
#include <iostream>
using namespace std;
class Vehicle {
 public:
string brand;
string model;
};

class Car: public Vehicle {
 public:
string type;
};

class ElectriCar: public Car {
 public:
 int batteryCapacity;

void accept(){
cout<<"Enter Car Details:"<<endl;
cout<<"Car Brand:";
cin >> brand;
cout<<"Car model:"<<endl;
cin >> model;
cout << "Car type:" << endl;
cin >> type;
cout<<"Battery Capacity:"<<endl;
cin>> batteryCapacity;
}

void display()
{ cout
   cout
   cout
   cout
}
};
int main()
{ ElectricCar E1;
   E1.accept();
   E1.display();
   return 0;
}
```

~~OUTPUT~~

4. Hierarchical

```
#include <iostream>
using namespace std;
class Employee {
  public
  int emp_id;
  string e_name;
};

class manager: public Employee
{ string public:
   string department;

void accept(){
{ cout<< "Enter department id: ";
   cin>> emp_id;
   cout<< "Enter name: ";
   cin>> e_name;
   cout<< "Enter department: ";
   cin>> department;
}
void display()
{ cout << " Manager details " << endl;
cout << " Employee ID : " << emp_id << endl;
cout << " Employee Name: " e_name << endl;
cout << " Department :" << department << endl;
 }
};

class Developer : public Employee
{ public:
   string p_lang;

 void getdata()
{ cout << " Enter employee name, employee id and
    programming language: ";
  cin >> e_name >> emp_id >> p_lang;
 }
 void showdata()
{ cout << " Employee ID :" << emp_id << endl;
  cout << " Employee Name:" << e_name << endl;
  cout << " Programming Language:" << p_lang << endl;
 }
};

int main()
{ Manager M1;
   M1.accept();
   M1.display();
   Developer D1;
   D1.getdata();
   D1.showdata();
   return 0;
}
```

~~OUTPUT~~

5. Hybrid inheritance.

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

class Person {
  public:
    string name;
    int age;
void displayPerson()
{
   cout << "Name: " << name << endl;
   cout << "Age: " << age << endl;
   }
};
class student : public Person {
 public:
  int student_id;
void displaystudent()
{
   displayPerson();
  cout << " student ID: " << student-id << endl;
  }
};
```

```
class sports
{ public:
   int sports_score;
   void setscore (int score) {
      sports_score = score;
   }
   void display_score ()
   {
    cout << " Sports score: " << sports_score << endl;
   }
};
class result : public student, public sports
{ public:
  int marks;
 void setmarks(int m) { mark = m; }
 int calculate () {
    return marks + sports_score;
  }
 void displayResult()
{ display Person();
  display student();
  display sports();
  cout << " Marks: " << marks << endl;
  cout << " Total (marks+sports score): " << calculate << endl;
}};
int main(){
  Result r;
  r. setperson (" Bob", 35);
  r. setstudent (102); r. setscore (45);
r. setmarks (90); r. displayResult();
return 0;
}
```

7/10

# EXPERIMENT : 7

1)

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

class area
{ public:
  int calculate (int l, int b)
  {
     return l * b;
  }

  int calculate (int side)
  {
    return side * side;
  }
};
int main()
{
 area a;
 cout << "Area of laboratory (Rectangle 10 x 5) "
      << a.calculate (10, 5) << endl;
 cout << "Area of classroom (square 6 x 6) " <<
      a.calculate (6) << endl;
 return 0;
}
```

2)

```
#include <iostream>
using namespace std;
class sum
{ public:
   float add (float a, float b, float c, float d,
              float e)
   {
     return a + b + c + d + e;
   }
   int add (int a1, int b1, int c1, int d1, int e1,
            int f1, int g1, int h1, int i1, int j)
   {
    return a1 + b1 + c1 + d1 + e1 + f1 + g1 + h1 + i1 + j1;
   }
};
int main()
{ sum s;
 cout << "Sum of 5 float no: " << s.add (1.1,
         2.2, 3.3, 4.4, 5.5) << endl;
 cout << "Sum of 10 integer nos: " << s.add (1,
         2, 3, 5, 4, 6, 7, 8, 9, 10) << endl;
 return 0;
}
```

3.

```
#include <iostream>
using namespace std;
class Number {
    int x;
    public:
     Number (int a)
     {
        x = a;
     }
     void display()
     {
       cout << " Value : " << x << endl;
     }
     void operator - ()
     {
       x = - x;
     }
};
int main ()
{
   Number n(10);
    n. display();
     -n;
    n.display();
    return 0;
}
```

4.

```
#include <iostream>
using namespace std;
class number
{  int x;
   public:
    number (int a)
   {  x = a;
    }
   void display()
   {
    cout << " Value : " << x << endl;
   }
   void operator ++ ()
   {
      ++x;
    }
};
int main()
{  number n(5);
    n. display();
     ++n;
    n. display();
     n++;
    n.display();
    return 0;
}
```

17/16

# EXPERIMENT - 8

1)

```
#include <iostream>
using namespace std;
class addstring {
    string str;
    public:
    addstring (string s)
    {  str = s ;
    }
    addstring operator + (addstring obj)
    {
        return Addstring (str + obj.str) ;
    }
    void display()
    {
     cout << *str <<endl;
    }
};
int main()
{
    addstring s1 ("xyz");
    addstring s2 ("pqr");
    addstring s3 = s1 + s2 ;
    cout<< "concatenated strings :" << s3.display();
    return 0;
}
```

2)

```
#include <iostream>
using namespace std;
class I-login
{ protected :
    string name,
    public :
    virtual void accept()
  { cout << "Enter Name : " ;
     cin >> name;
     cout << "Enter Password : " ;
     cin >> password;
  }
    virtual void display()
  { cout << "Name : " << name << endl;
    cout << "Password : " << password << endl;
  }
};
class email_login : public I-login
{   string email;
    public:
     void accept()

{ cout << " Email login details : " << endl;
   cout << "Name : " << name << endl;
   cout << "Password : " << password << endl;
  }
};
```

```
class membershiplogin : public, Ilogn
{   int member_id;
    public:
    void accept()
    {
     cout << " Membership login : " ;
     I_logn : : accept();
     cout << " Enter member ID : " ;
      cin >> member_id;
    }

    void display()

{  cout<< "Membership login Details : " ;
    cout<< " Name : "<< name << endl;
    cout<< " Password : " << password << endl;
    cout << " Member ID : " member_id << endl;
    }
};
int main()
{   I_login * login ;
    emailLogin e ;
    membershiplogin m ;

    login = & e ;
    login → accept();
    login → display();
    login = & m ;
    login → accept();
    login → display();
    return 0;
}
```

17110

EXPERIMENT - 9

a)

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

int main()
{ fstream new_file ;
new_file . open ("new_file txt ", ios : : out);
if (! new_file)
{ cout << " File creation failed ! " << endl;
}
else
{ cout << " New file created " << endl;
  new - file . close ();
}
return 0;
}
```

b)

```
#include <iostream>
#include < fstream>
using namespace std;
int main()
{
  ofstream fuel (" fuel .txt ", ios :: app);
    if (! File i)
  {
```

```
cout << "Error opening file1.txt " << endl;
 return 1;
 }

File1 << "Welcome to MIT" << endl;
File1.close();
ifstream file1.read("File1.txt");
ofstream file2("file2.txt");

if (!File1_read)
{ cout<< "Error opening File1.txt for
          reading");
   return 1;
}

if (!file2)
{
 cout<< "Error opening file2.txt for
       writing"<<endl;
  return 1;
}
 char ch;
while (file1_read.get(ch))
{
    file2.put(ch);
}
cout << "content copied successfully from
       file1.txt to file2.txt" << endl;
       file1_read.close();
       file2.close();
  return 0;
}
```

## EXPERIMENT-10

1)

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

template <class T>
T sumArray (T arr[], int n){
    T sum = 0;
  for (int i=0; i<n ;i++)
    sum += arr[i];
   return sum;
}

int main(){
 int int_arr[5] = {1,2,3,4,5};
 float flo_arr[5] = {1.1, 2.2, 3.3, 4.4, 5.5};
 double dec_arr[5] = {0.5, 1.5, 2.5, 3.5, 4.5};

 cout << "Sum of Integer array: " << sumArray(
    int_arr,5) <<endl;
 cout << "Sum of float array: " << sumArray(flo_arr,5)
   <<endl;
 cout << "Sum of double array: " << sumArray(dec_arr,5);

 return 0;

}
```

2)

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

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

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

int main(){
 int num = 5;
 double dec_num = 2.5;
 string str = "Apple";

cout << " Square of integer : "<< square (num)<<endl;
cout<< "Square of double: "<< square (dec_num)<<endl;
cout<< "Square of string: "<< square (str)<< endl;

return 0;
}
```

3)

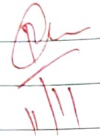
```
#include <iostream>
#include <math.h>
using namespace std;

template <class T1, class T2> class
calc {
  public:
   T1 x;
   T2 y;

 calc (T1 n, T2 m){
    x = n;
    y = m;
 }
 void sum(){
    cout <<"Sum: " << x + y << endl;
 }
 void diff(){
    cout << " Difference: " << x - y << endl;
 }
 void pro(){
  cout << "Product: " << x * y << endl;
 }
 void quo(){
  cout << "Quotient: " << x / y << endl;
 }
 void rem(){
   cout<< "Remainder: " << x % y << endl;
}
```

```
};

int main(){
  calc <int,int> n(100,200);
  n.sum();
  n.diff();
  n.pro();
  n.quo();
  n.rem();
  return 0;
}
```



## EXPERIMENT - 11

1)

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

int main(){
  vector <int> v = {1,2,3,4,5,6,7,8,9,10};
  cout << "Initial Vector:" << endl;
  for(int i=0; i<10; i++){
    cout << v[i] << " " << endl;
  }
  cout << "Multiply by 10" << endl;
  for(int i=0; i<10; i++){
    v[i] = v[i] * 10;
  }
  cout << "New Vector:" << endl;
  for(int i=0; i<10; i++){
    cout << v[i] << end " " << endl;
  }
  return 0;
}
```

```
int n;
cout << "Enter scalar value:" << endl;
cin >> n;
```

using iterator.

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

int main()
{
 int n;
 cout<<"Enter scalar value:"<<endl;
 cin>>n;

 vector<int> v={1,2,3,4,5};
 cout<<"Initial vector:"<<endl;
 for(int i=0;i<5;i++)
 {
  cout<<v[i]<<" "<<endl;
 }
 for(int i=0;i<5;i++)
 {
  v[i]=v[i]*n;
 }
 cout<<"New Vector:"<<endl;
 for(int i=0;i<n;i++)
 {
  cout<<v[i]<<" "<<endl;
 }
 return 0;
}
```

EXPERIMENT -12

a) WAP using STL to implement stack.

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

int main()
{ stack <int> s;
  cout<<"Stack:";

  s.push(10);
  s.push(20);
  s.push(30);

cout<<s.top()<<endl;

for(int i=0;i<s.size();i++)
 {
    cout<<s.pop();
 }
 return 0;

}
```

b) WAP to implement queue using STL.

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

int main(){
  queue <int> q = { 20, 25, 30 };

  cout << q.size();
  q.push(40);
  q.push(50);

  cout << q.front() << endl;
  cout << q.back();


  if (q.empty())
  {
    cout << "queue empty";
  }
  else
  { cout << "Elements being removed: ";
    for(int i=0; i< q.size(); i++)
    {
      cout << q.pop() << endl;
    }
  }
  return 0;
}
```

# ADWITA KURLE

@adwitakurle21

Complete your profile

Add your missing details →

This data will be helpful to auto-fill your job applications

0%

## Personal Information

adwitakurle21@gmail.com

Add your mobile number

India

## My Badges


## My Resume

+ Add Resume

Add your resume here

## My Certifications


## EEO settings