Anushka Hardikar

@hardikaranushka7

Personal Information

hardikaranushka7@gmail.com

Add your mobile number

India

My Resume

+ Add Resume

Add your resume here

EEO settings

Complete your profile

Add your missing details →

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

0%

My Badges

Problem Solving ★★★★★

C++ CPP ★★★★★

Python ★★★★★

My Certifications

You have not earned any certificates yet. Get Certified

Work Experience

+ Add Work Experience

Add your work experience. Don't forget to add those internships as well.

Education

+ Add Education

We believe in skills over pedigree; but go ahead add your education for the recruiters who don't.

Experiment 1

1.) Write a C++ program to declare class student having data members roll_number and name. Accept & display for single student.

```
#include <string>
class student
{
    int Roll_no;
    string Name;
    Public:
    void accept()
{
    cout << " Enter name & Roll no:";
    cin >> Name >> Roll_no;
}   getline(cin, name);
    void display(){
    cout << "Name:" << Name;
    cout << "Roll Number:" << Roll_no;
};

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

2.) Write C++ code to create a class book having data members book_name, b_price & b_pages. Accept data for two books & display name of book having greater price.

```
  class book
{  Public:
     string b_name;
     int b_price;
     int b_pages;
     Public:
     void accept()
{
    cout << "Enter book name, price
              & number 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()
{
    book B1, B2;
     B1.accept();
     B2.accept();
```

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

O/P - Enter name & Roll No: Anushka
7
Name: Anushka
Roll No : 7

O/P - Enter book name, price & number of pages : Bookone
150
200
Enter book name, price & number of pages : Booktwo
120
300
Book name : Bookone
Book Price: 150
No. of pages : 200

3) Write a program to declare class 'Time', accept time in HH:MM:SS format, convert into seconds & display them.
Keyword

```
class timesec
{
    int h,m,s;
    char c;
    public:
    void accept()
{
    cout << "Enter time in HH:MM:SS
            format:";
    cin >> h >> c >> m >>c >>s;
}

    void display(){

    int totalseconds = h*3600 + m*60
                       + s;
    cout << "Time is seconds:" <<
        totalseconds << endl;
    }
};

int main()
    timesec t;
    t.accept();
```

```
t.display();

return 0;

}
```

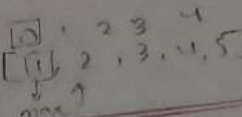
O/P – enter time in HH:MM:SS format:
20:44:21
Time in seconds : 74661

157

* Experiment 2 -

Q1> WAP to declare a class city having data members name & population. Accept data for 5 cities & display name of city having highest population.

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

class city {

public:
int population;
string name;
public:

void accept() {

cout << " Enter City name & population :";
cin << name << population;
}

void display() {
                                    name << endl;
cout << " Name: " << end
cout << " population: " population << endl;

}
};
```




```
int main(){

city c[5];
int i;
int max_pop;

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

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

cout<<" City with maximum population
c[max_pop].display();

return 0;

}
```

O/P - Enter name of city & population : Pune 300
" : Mumbai 500
" : Chennai 700
" : Banglore 900
" : Delhi 40000

City with maximum population:
Name : Banglore
Population: 9000.

Q2) WAP to declare class 'Account' having data members as Account no. & balance. Accept data for 10 accounts & give 10% interest where balance is equal to or greater than 5000.

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

class account {
  public:
  int acc_no;
  int balance;
  public:

  void accept() {

  cout<<" Enter Account No. & balance:";
  cin << acc_no << balance;
  }
    void display() {

    cout <<"Account No.:" << acc_no<<endl;
    cout <<" Balance:" << balance <<endl;
  }};
```

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

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

cout<<"Accounts with balance
  greater than 5000: "<< endl;

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

return 0;

}
```

O/P – Enter account number & Balance:

```
1  4000
"
2  5000
"
3  9000
"
4  6000
"
5  6000
"
6  4000
"
7  9000
"
8  5001
"
9  6800
"
10 10,000
Accounts with balance greater than 5000
Account number : 3
Balance : 9900
Account number : 4
Balance : 6600
" : 5
" : 6600
" : 7
" : 9900
" : 8
" : 5501
" : 9
" : 7480
" : 10
" : 111000
```

Q3) WAP to declare a class 'Staff' having members name & post. Accept data for 5 staff & display names of HOD:

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

class staff {

string name;
string post;
public:

void accept(){

cout << "Enter name :" << endl;
getline(cin, name);
cout << "Enter post :" << endl;
getline(cin, post);
}

void display(){

cout << "In Name :" << name;
cout << "In 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 == "HOD"){
        s[i].display;
  }
}

return 0;
}
```

O/P - Enter Name : A
Post : HOD
Enter Name : B
Post : GM
" : C
" : Finance
" : D
" : IT
" : E
" : HOD
Name : A
Post : HOD
Name : E
Post : HOD

29/7/25

29.7.25

Experiment 3

Q1) Write a program to declare a class book having data members as book-title, author_name and price. Accept & display information for an object using pointer.

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

class book {
 String book-title;
 String author_name;
 float price;
 public:
{
  void accept () {

  cout << Enter book title, author & price
  cin >> book-title >> author_name >>
      price;

 void display () {
  cout << "Book Title:" << book-title;
  cout << "Author name:" << author_name;
  cout << "Price:" << price;

 }
};
```

```
int main() {
  Book b1, b2;
  book * ptr;

  ptr = &b1;
  cout << "Enter details for book:"
  ptr -> accept();

  cout << "Book details:";

  ptr = &b1;                    b1.accept()
  cout << "Book:";              ptr-> accep
  ptr -> display();

  return 0;

}
```

(27) WAP to declare class 'student' having data members roll no & percentage. Using this pointer invoke member function to accept & display this data for one object

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

class student {

int roll_no;
int percentage;
public:

void accept() {

cout << "Enter Roll numer:";
cin >> this -> roll_no;
cout << "Enter percentage:")
cin >> this -> percentage;
}

void display() {
  this -> accept()
  cout << "roll number:" << roll_no;
  cout << "percentage:" percentage;
  }
};
```

```
int main() {

student s1;
 s1. display();

return 0;

}
```

Q3) WAP to demostrate use of nested class.

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

Class Student {
 public:
 Class marks {
  public:
  int m1, m2, m3;

 void getInput () {

  cout << " Enter marks for 3 subjects
  cin >> m1 >> m2 >> m3;
 }

 float getpercentage () {
   int total = m1+m2+m3
   return total / 3.0;
  }
 };
};

int main ()
{     class          nested
  Student :: marks marks;
   marks. getInput ();

float percentage = marks.get
                      percentage();

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

return 0;
}
```

O/P Enter Marks for 3 subjects : 9
98
98
Percentage : 97%.

5/8/25

$\frac{m_1+m_2+m_3}{300} \times 100$

8.8.25

Experiment 4

Q1) WAP to swap two numbers from same class using object as function argument write swap function as member function.

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

class Number{

  int value;
pubic:
void getvalue(){
cout << "Enter value:";
cin >> value;
}

void display(){
cout << value << endl;
}

void swapValues(Number &obj){

   int temp = value;
    valeue = obj.value;
    Obj.value = temp;
  }
};

int main(){

Number n1, n2;

cout << "Enter first value for
        Object:";
n1.getValue();

cout << "Enter value for second
        Object:";
n2.getValue();

cout << "\n Before swapping" << end
cout << "n1 = "; n1.display();
cout << "n2 = "; n2.display();

n1.SwapValue(n2);

cout << "\n After Swapping:" << end
cout << "n1 = "; n1.display()
cout << "n2 = "; n2.display()

return 0;
}
```

O/P -

```
Enter value for first object:
Enter value : 5
Enter value for second object:
Enter value : 8
Before :  n1 = 5   After   n1 = 8
Swapping  n2 = 8   swaping  n2 = 5
```

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

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

class Number {
 private:
 int n1, n2;
 public: void accept() {
 cout >> << " Enter two number ";
 cin >> n1 >> n2;
 }

 void display() {
 cout << " n1 = " << n1 << endl;
 cout << "n2 = " << n2;
 } void swap
 friend void swapNumber (Number &n)
 { int temp = n.n1;
 n.n1 = n.n2;
 n.n2 = temp;
 }
 };

int main() {

 Number m;

 m.accept();
 cout << "Before swap :",
 m.display();

 swap (m);

 (out << " After swap : ";
 m.display();

 return 0;

 }
```

O/P-

```
Enter 2 numbers : 2
3 B4
After swap : n1 = 2
n2 = 3
After swap : n1 = 3
n2 = 2
```

Q3) WAP to swap 2 numbers from different classes using friend function.

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

class Class2;
class Class1{
 private:
int num1;

 Public:
 void accept(){
 cout<<" Enter first number:";
  cin >> num1;
 }
 void display(){
 cout <<"Class1 value:"<< num1<<endl;
}

friend void swapNumbers( Class1 &a,
                         Class2 &b);
};

void swapNumbers (Class1 &a, Class2 &b)
{ class Class2{
 private:
int num2;
public:
void accept(){
cout <<" Enter second number:";
 cin >> num2;
}
void display(){
 cout <<" Class 2 value :" << num2 <<
  endl;
}

friend void swapNumbers ( Class1 &a,
                          Class2 &b);
};

void swapNumbers (Class1 &a, Class2
                          &b){
   int temp = a.num1;
   a.num1 = b.num2;
   b.num2 = temp;
}

int main(){
 Class1 A;
 Class2 B;

A.accept();
B.display(),
  accept

cout <<" Before swaping:";
```

```
A.display();
B.display();

SwapNumbers(A,B);

cout << "\n After swap: \n";
A.display();
B.display();

return 0;

}
```

```
Enter first number:5
Enter second number:6

Before swap:
Class1 value: 5
Class2 value: 6

After swap:
Class1 value:6
Class2 Value:5
```

Q4) Create class Result1 & 2 to store student marks. Read value of marks from both class objects & compute average of 2 results.

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

class Result2;
class Result1{
 private:
 float mark1;
 public:
 void accept(){
 cout << "Enter marks for first student:";
  cin >> mark1;
 }
friend float Average(Result1, Result2
};

class Result2{
 private:
 float mark2;
 public:
 void accept(){
 cout << "Enter marks for second
        student:";
  cin >> mark2;
}
```

```
friend float Average (Result1, Result2)
};

float Average (Result1 a, Result2 b){
  float average = (a.mark1 + b.mark2)/
                  2.0;
  return (lverage;
}

int main(){

  Result1 r1;
  Result2 r2;
  float avg;

  r1.accept();
  r2.accept();

  avg = Average (r1, r2);
  cout << "Average of two results is :" <<
    averag << endl;

  return 0;

}
```

O/P
```
Enter marks for first student : 70
Enter marks for second student : 60
Average of two results is : 65
```

Q5) WAP to find greatest number among two numbers from diff classes using friend function.

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

class Number2;
class Number1 {
private:
  int num1;
pubic:
  void accept (){
  cout << "Enter number :";
  cin >> num1;
}
  friend void Find Greatest (Number1, Numbe
};

Class Number 2{
private:
  int num2;
public:
  void accept(){
  Cout << "Enter number:" ;
    cin >> num2;
}                                    be2
  friend void  find Greatest ( Number1, Num
};
```

```
void findGreatest (Number1 a, Number2 b)
{
    if (a.num1 > b.num2){
    cout << "Greatest number is:" << a.num1 <<endl;
    } else is (b.num2 > a.num1){
    cout << "Greatest number is:" <<
    b.num2 <<endl;
    } else {
    cout << "Both are equal" << endl;
}
}

int main() {

Number1 obj1;
Number2 obj2;

obj1.accept();
obj2.accept();

findGreatest (obj1, obj2);

return 0;

}
```

O/P- Enter number:8
Enter number:7
Greatest number is :8

11.8.25

Practice Questions (for Friends)

Create two classes, ClassA and ClassB each with private integer. Write a friend function sum() that can access private data from both classes & return the sum.

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

class ClassB;

class ClassA {
private:
  int valueA;
public:
void input() {
  cout << "Enter value for ClassA:"
  cin >> valueA;
}

friend int sum ( ClassA a, ClassB b
};

class ClassB {
private:
  int valueB;
```

```
public:
void input() {
  cout << " Enter value for ClassB :";
  cin >> valueB;
}

friend int sum (ClassA a, ClassB b);
};

int sum (ClassA a, ClassB b){
return a.valueA + b.valueB;
}

int main() {
  ClassA objA;
  ClassB objB;

  objA.input();
  objB.input();

  cout << " Sum: " << sum (objA,
      objB) << endl;

return 0;

}
```

Op - Enter value for ClassA : 2
Enter value for ClassB : 5
Sum : 7

Q27 Write a program with a class Number that contains a private integer. Use a friend function swapNumber (Number&, Number&) to swap private values of 2 Number objects.

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

class Number {
  private:
  int value;

  public:
  void accept() {
  cout << "\n Enter a value: ";
   cin >> value;
  }

  void display() {
    cout << value << endl;
  }

  friend void swapNumbers (Number& n1,
                           Number& n2)
};

void swapNumbers (Number& n1, Number&
                  n2)
```

```
// {
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp;
}

int main() {

Number num1, num2;
cout << "Enter first number:";
num1.access();

cout << "Enter second number:";
num2.access();

cout << "\n Before swapping:\n";
 cout << " Number 1:";
  num1.display();
  cout << "Number 2:";
  num2.display();

Swap Number(num1, num2);

cout << "\n After swapping:\n";
cout << "Number 1:";
num1.display();
cout << "Number 2:";
num2.display();
return 0;
}
```

O/P -

```
Enter first number:
Enter a value: 2
Enter second number:
Enter a value: 3

Before swapping:
Number 1: 2
Number 2: 3

After swapping:
Number 1: 3
Number 2: 2
```

Q3.) Define two classes Box and Cube each having a private volume. Write a friend function find Greater(Box, Cube) that determi which object has a larger volu

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

class Cube;
class Box {
  private:
  double volume;
  public:
  void input() {
   double length, width, height;
```

```
cout << "Enter length, width,
        height of box: ";
cin >> lenght > width >> height;
volume = length * width * height;
}

friend void findGreater (box b, cube
};

class cube {
 private:
 double volume;
 pubic:
 void input(){
  double length, width, height;
  cout << "Enter length, wiath,
     height of cube: ";
  cin >> length >> width >> height;
  volume = length * width * height;
}
friend void findGreater(box b, cube c);
};

void findGreater (Box b, Cube c){
 if (b.volume > c.volume)
    cout<<" Box has greater volume.\n";
else if ( c.volume > b.volume)
    cout << " Cube has greater
        volume.\n";
else
  cout << " Equal volume.\n" ;
}

int main() {

 Box box();
 Cube cube();

 findGreater (box, cube);

 return 0;

}
```

O/P- Enter length, width and height of the box : 30 40 50
Enter length, width, and height of cube : 30 30 30
Box has greater volume.

Q4) Create a class complex with real and imaginary parts as private members. Use a friend function to add two complex numbers and return as a new complex object.

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

Class Complex {
  private:
  float real;
  float imag;

  public:
  void accept () {
  cout << " Enter real and imaginary
          part :";
    cin >> real >> imag;
  }

void display (){
  cout << real<< " + " << imag << "i"
  << endl;
}

friend Complex addComplex
        (Complex c1, Complex c2);
};

Complex addComplex (Complex c1,
                    Complex c2);
{

Complex result;
result.real = c1.real + c2.real;
result.imag = c1.imag + c2.imag;
return result;
}

int main (){
  Complex num1, num2, sum;

cout << "Enter first complex number: \n";
num1.accept();

cout << "Enter second complex number:
        \n ";
num2.accept();

sum = addComplex (num1, num2);

cout << " The sum of 2 complex
        number is : ";
sum.display();

return 0;
}
```

Op - Enter first complex number:
Enter real and imaginary part: 2
3

Enter second complex number:
Enter real and imaginry part: 4
5

The sum of 2 complex number
6+8i

Q5) Create a class Student with private data members. name and three subject marks. write a friend function calculateAverage (Student) that calculates and displays the average marks.

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

class Student {
  private:
  string name;
  float m1, m2, m3;

  pubic:
  void accept() {
  cout << " Enter name of student: ";
   cin >> name;

   cout << " Enter marks for all three
          subjects: ";
   cin >> m1 >> m2 >> m3;
  }

friend void calculateAverage (Student s)
};
void calculateAverage (student s) {
  float avg = (s.m1 + s.m2 + s.m3) / 3.0;
```

```
cout << "In Student Name: " <<
    s.name << endl;
cout << "Average Marks: " <<
    avg << endl;

}

int main() {

  Student std;
  std.accept();
  calculateAverage(std);
  return 0;

}
```

OP - Enter name of student : Anvesha
Enter marks for all three subjects:
100
99
98

Student Name : Anvesha
Average Marks : 99

Q6) Create three classes Alpha, Beta, and Gamma each with a private data member. Write a single friend function that can access all three and print their sum.

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

class Beta;
class Gamma;
class Alpha {

 private:
  int a;
 public:
 void accept() {
 cout << "Enter value for Alpha: ";
  cin >> a;

 friend void Sum ( Alpha, Beta, Gamma);
};

class Beta {
  private:
  int b;
  public:
 void accept () {
 cout << "Enter value for Beta ";
```

```
    cin >> b;
}
friend void sum( Alpha, Beta, Gamma);
};

class Gamma {
  private:
  int g;
  public:
  void accept(){
  cout << "Enter value for Gamma :"
   cin >> g;
  }

  friend void sum (Alpha, Beta, Gamma);
};
```

O/P - Enter value for Alpha : 30
Enter value for Beta : 40
Enter value for Gamma : 50
The sum of alpha beta and gamma is : 120

07) Create a class Point with private members x and y. Write a friend function that calculates distance between two point objects.

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

class Point {
  private:
    float x;
    float y;

  public:
    void accept () {
    cout << " Enter coordinates (x y)";
    cin >> x >> y;
  }

  friend float calcDistance (Point, Point);
};

float calcDistance ( Point p1, Point p2)
{
    float dx = p2.x - p1.x;
    float dy = p2.y - p1.y;
    return sqrt (dx * dx + dy * dy);
}
```

```
int main() {

Point A, B;
A.accept();
B.accept();
float distance = calcDistance(A,B);
Cout << "The distance between
    the two points is: " <<
    distance << endl;

return 0;

}
```

O/P - Enter coordinates (x y): 3.
4

Enter coordinates (x y) : 5
5

The distance between the two points is : 2.23607

Q8) Create two classes: BankAccount and Audit. BankAccount holds private balance information. Write a friend functions in Audit that accesses and prints balance information for auditing

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

class Audit;

class BankAccount {
private:
    double balance;
public:
    void getBalanceFromUser() {
        cout >> "Enter account balance
            : ₹ ";
        cin >> balance;
    }

    friend void AuditReport(BankAccount,
        Audit);
};

class Audit {
public:
    friend void AuditReport(BankAccount,
                            Audit)
```

```
};
void AuditReport (BankAccount
        account, Audit) {

cout << "Audit Report : Account
    balance is ₹ " << account.
    balance << endl;
}

int main() {

BankAccount acc;
Audit auditor;

acc.getBalanceFromUser();
AuditReport(acc, auditor);

return 0;

}
```

O/P: Enter account balance : ₹ 10000
Audit Report : Account balance is ₹ 10000.

2/9/25

Experiment 5

Q1.) Write a program to find sum of numbers between 1 to n where value of n will be passed to the constructor.

* Default :

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

class sum {
    int n;
    int result = 0;

public:
    Sum() {
n=5 for (int i=1; i<=n; i++)
        result += i;
    }

void display() {
    cout << "sum of numbers from
1 to" "n" << "is" << result << endl;
    }
};

int main() {
```

```
Sum s;
s.display();
return 0;
}
```

O/P - Sum of numbers from 1 to 10 is : 55

* Parameterized :

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

class Sum {
 int n;
 int sum = 0;
 public:
 Sum (int num){
   n = num;
  for ( int i=0; i<=n; i++) {
     sum = sum+i;
   }
 }

void display(){
  cout << " sum = " << sum;
 }
};
```

```
int main() {

  Sum s1(5);
  s1 . display();

return 0;

}
```

O/P - Sum = 15

* Copy :

```
#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 = s+i;
}
cout << "sum =" << s;
 }
};
int main(){

 Sum s1;
 Sum s2(s1);

return 0;
}
```

O/P Sum = 10

Q2) Write a program to declare class 'student' having dm name & %. Write a constructor to initialize these data members. Accept & display for one student.

* Default :

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

class student{
  string name;
  float percentage;

Public:
 student()
{
    name = "anushka";
    percentage = 90.0;
}
 void display()
{
   cout << " Name & Percentage: " <<
    name << "  " << percentage;
}
};

int main()
```

```
{
    Student S1;
    S1.display();

return 0;

}
```

O/P - Name & Percentage : anushka 90.0

* Parameterized:

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

class student {
    float per;
    String name;

 public:
 Student ( float p, string n)
{
      per = p;
      name= n;
}
void display {

   cout <<"Name:" << name <<endl
   cout << "Percentage:" << per;
}
```

```
n=i.x
A a()
A a(a)
};

int main()
{
    student S1 (90, "anushka");
    si.display();

return 0;

}
```

O/P - Name : anushka
Percentage : 90

* Copy:

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

class student {
 string name;
 float percentage;

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

student (student &s) {
```

Name: Anushka
Percentage: 90

```
    name = s.name;
    percentage = s.percentage;
}

void display(){
    cout "Name :" << name << endl;
    cout << "Percentage :" << percentage
    << "%." << endl;
}
};

int main(){
    string n;
    float p;

cout << "Enter name :";
getline (cin, n);

cout << "Enter Percentage :";
cin >> p;

Student S1 (n,p);
Student S2 (S1);

cout << "In Student Data :" << endl;
S2.display();
return 0;
```

Enter name: Anushka.
O/P - Enter Percentge : 90

Q3) 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 2 objects of class & display data.

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

class college {

  int roll_no;
  String name;
  String course;
public:
  college ( int r, Sring n, string c =
            "computer Science")
  {
     roll_no = r
     name = n;
     course = c;
  }
void display ()
{
   cout << " Roll No:" << roll_no << end
   cout << " Name:" << name << endl
   cout << " course:" << course << endl;
}};
```

```
int main()
{
  int r1, r2;
  string n1, n2;
  cout << "Enter roll no & name fir
          student:";
  cin >> r1 >> r2;

  college c1 (r1, n1);
  college c2 (r2, n2);

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

O/P - Enter roll no & name for first st
54 abc
Enter roll no & name for second
22 xyz

Details for student:
Roll No: 54
Name: abc
Course: Computer Engineering
Roll No: 22
Name: xyz
Course: Computer Engineering

Q4) WAP to demonstrate constructor overloading:

```
#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 (10,20);
  n1.display ();
  n2.display ();
  return 0;
}
```

Qn 16/9/25

O/P- x = 5 y = 0
x = 10 y = 20

Experiment 6

Q1) Single Inheritance:
Create a base class called Person with attributes name and age. Derive a class Student from Person that adds an attribute rollNumber. Write functions to display all details of the student.

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

class Person {
protected:
   string name;
   int age;
public:
void acceptpersondetails () {
   cout << "Enter name :" ;
   cin >> name;
   cout << "Enter age:";
    cin >> age;
}

void displaypersondetails() {
   cout << "Name:" << name<<endl;
   cout << "Age:" << age <<endl;
 }
};

class Student : public Person {
private:
int rollno;

public:
void acceptstudentdetails() {
   acceptpersondetails();
    cout << "Enter Roll No:";
    cin >> rollno;
}

void displaystudentdetails() {
  cout << "Roll No:" << rollno <<endl;
 }
};

int main() {
  Student s;

  s.acceptstudentdetails();
  cout << "\n student Details:\n";
  s.displaystudentdetails();

return 0;
}
```

O/P
```
Enter name : anushka
Enter age: 17
Enter Roll Number : 7

Student Details:
Name: anushka
Age: 17
Roll No:7
```

Q2.) Multiple Inheritance:

Create 2 base classes Academic and sports. Academic class contains marks of a student sports class contains ~~marks of a student~~ sport score. Create derived class Result that inherits from both academic & sports. write a function to calculate total score & display details.

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

class Academic {
public:
    int marks;
    void getMarks(){
      cout << "Enter academic marks:";
      cin >> marks;
    }
};

class Sports {
public:
   int score;
   void getscore(){
    cout << "Enter sports score :";
    cin >> score;
   }
};

class Result : public Academic, public Sports {
public:
   void display () {
    int total = marks + score;
    cout << "In Academic marks : " << marks
    cout << "In sports score : " << score;
    cout << " In Total Score : " << total << en
   }
};

int main() {
   Result r;
   r. getMarks();
    r. getScore();
   r. display();

return 0;
}
```

```
Enter Academic Marks: 500
Enter sports score: 85

Academic Marks : 500
Sports score : 85
Total score : 585
```

Q03) Multilevel Inheritance:
Create class Vehicle with attributes like brand and model. Derive a class Car from Vehicle which adds an attribute type. Further derive a class ElectricCar from car which adds battery capasity. Write function to display details.

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

class Vehicle
{
  public:
     string brand;
     string model;
};

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

class EV_car : public Car
{
   Public:
     int battery_capacity;

 void input ()
{
   cout <<" Enter Brand :";
```

base clm. protected
derive clm. private
except/display - public

```
cin >> brand;
cout << " Enter model :";
cin >> model;
cout << " Enter type :";
cin >> type;
cout << " Enter battery capasity :";
 cin >> battery_capasity;
}

void display()
{
    cout << "Brand :" << brand << endl;
    cout << "Model :" << model << endl;
    cout << "Type :" << type << endl;
    cout <<" Battery capacity :" << battery_
     capacity << endl;
  }
};

int main()
{
    EV_car e;
    e.input();
    e display();

  return 0;
}
```

Enter Vehicle Brand : Tata
Enter Vehicle Model : Curvv
Enter type : SUV
Enter battery capacity : 45 kwh

Brand : Tata
Model : Curvv
Type : SUV
Battery Capacity : 45 KWH

Q4) Hierarchial Inheritance :
Create a base class Employee with attributes empID & name. Derive 2 classed manager and developer from Employee. Manager has an attribute department & Developer has an attribute progLang. Write functions to display details.

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

class Employee {
protected:
    int empID;
    string name;

public:
    void setEmployeeDetails (int id, string n)
    {
        empID = id;
        name = n;
    }

    void displayEmployeeDetails () {
        cout << "Employee ID :" << empID << endl;
        cout << " Name: " << name << endl;
    }
};

class Manager : public Employee {
private:
    string department;

public:
    void setManagerDetails (int id, string n,
        string d) {
        setemployeeDetails (id, n);
        department = d;
    }

    void displayManagerDetails () {
        displayEmployeeDetails ();
        cout << "Department : " << department
        << endl;
    }
};

class Developer : public Employee {
private:
    string programmingLanguage;

public:
    void setDeveloperDetails (int id, string n,
    {
        setEmployeeDetails (id, n);
        programmingLanguage = pl;
    }

    void displayDeveloperDetails () {
        displayEmployeeDetails ();
        cout << "Programming Lang : " << prog
        language << endl; } };
```

```
int main() {

    Manager m;
    Developer d;

    m. set  int id,
            String name, dept, lang;


    cout << "Enter Manager ID : ";
    cin >> id;
    cin.ignore();
    cout << "Enter Manger name:";
    getline ( cin, name);
    cout << "Enter Department:";
    getline (cin, dept);

    m. setManagerDetails (id, name, dept);

    cout << " In Enter devloper ID : ";
    cin >> id;
    cin. ignore();
    cout << "Enter Developer name : ";
    getline ( cin, name);
    cout << " Enter pl : " ;
    getlinr (cin, lang);

    d. setDeveloperDetails ( id, name, lang);
    m. displayManagerDetails ();
    d. display Developer Details();
return 0;
}
```

Q5) Hybrid Inheritance:

Combine multilevel & multiple inheritance creat base class person with name & age Derive student from person. Create sports and academic. Derive class result from Student & Sports.

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

class Person {
    protected:
    string name;
    int age;
};

class Stud : Public Person {
    protected:
    int roll;
};

class Academics {
    protected:
    int marks;
};

class Sports {
    protected:
    int Sport_score;
};
```

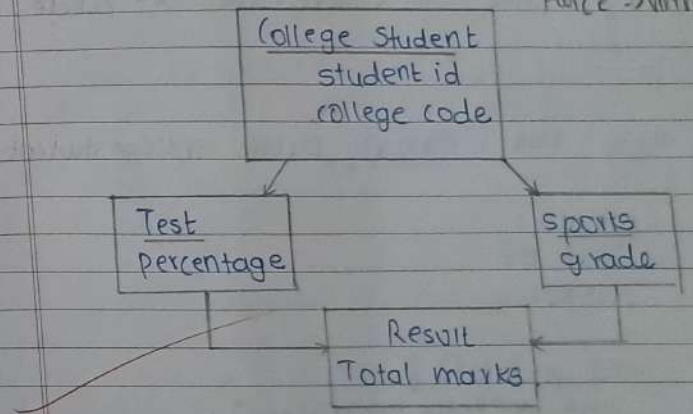
```
class score : public stud, public academics,
public sports {
 public:
  score () {

   cout << " Enter name, roll, sports score,
           marks : ";
   cin >> name >> age >> roll >> sports_score >>
    marks;
}

 void disp() {

   int score = marks + sports_score;
   cout << endl << " Name : " << name << endl;
   cout << " Age : " << age << endl;
   cout << " Roll no : " << roll << endl;
   cout << " Sports score : " << sports_score << endl;
   cout << " Result : " << score;

  }
};

int main() {
    score s1;
    s1. disp ();

return 0;

}
```

O/P- Enter Name, roll, sports score, marks:
Anushka
17
7
85
500

Name : Anushka
Age : 17
Roll no : 7
Sports score : 85
Result : 585

Q6) Virtual base class:

class being inherit
twice -> virtual




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

class college_student
```

```
{
  private :
  int sid ;
  string ccode;

  public:
  void accept () {
   cout << " enatter student id : ";
   cin >> sid;
   cout << " Enter scollege code :";
   cin > ccode;
  }
   void display () {
      cout << " Student id :" << sid;
      cout << " College code :" << ccode;
   }
};

class test : virtual public college student
{
  private :
  float p;
  public:

  void getdata() {
    cout << " Enter percentage: " ;
    cin >> p;
  }

  void showdata() {
    coutt" Percentage : "<<p;
  }
};
```

```
class sports : public virtual college student
{
  Private :
   int grade ,
   public:      grade
   void accept ()
   {
      cout << " Enter grade : " << endl;
      cin >> grade;
   }              grade
   void display () {

      cout Grade : " << grade;
   }
};

class Result : public test , public spor
{
   private :
   int total_marks ;
   public :

   void accept() getResultdata() {
   cout << " Enter total Marks : ";
   cin >> total_marks;
   }

   void displayresultdata() {
    display();
    showdata();
    displaygrade();
    cout << " Total marks : " << total_m
};
```

```
int main() {

    Result r;

    cout << "Enter student details:";
    r.accept();

    cout << "Enter test details:";
    r.getdata();

    cout << "Enter sports details:";
    r.acceptgrade();

    cout << "Enter result Details:";
    r.getResultdata();

    cout << "STUDENT REPORT";
    r.displayResultdata();

return 0;
}
```

17/10

30-9-24 Experiment 7

write a program to demonstrate compile time polymorphism (Function overloading & Operator overloading - unary)

reusing same function name.

define purpose of operator according to obj

Q1) WAP using function overloading to calculate area of laboratory (rectangle) and area of classroom square.

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

class Area
{
  public:
  float calculate (float length, float breadt
    return length * breadth;
  }

    float calculate (float side) {
      return side * side;
    }
};

int main() {
Area a;
float length, breadth, side;

cout "Enter length, breadth of laboratory :"
cin >> length >> breadth;
cout << "Area of laboratory (rectangle):"
```

```
a.calculate (length, breadth) << endl;
        Enter side
cout << " Area of classrooom:" ;
cin >>side;
cout << " Area of classroom (sauare):" ;
  a.calculate(side);

return 0;
}
```

O/P - Enter length, breadth for laboratory:
5 4
Area of laboratory (rectange): 20
Enter side of classroom:
2
Area of classroom (sauare): 4

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

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

class sumcalculator {
public:

  float sum ( float a, float b, flaat c, float d,
     float e) {
     return  a+b+c+d+e;
  }

int sum( int a, int b, int c, int d, int e,
        int f, int g, int h, int i, int j){

   return  a+b+c+d+e+f+g+h+i+j;
}
};

int main() {

  sumcalculator s;

float fsum = s.sum ( 1.1f ,2.2f ,3.3f ,5.5f ,4.4
 int isum = s.sum ( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

 cout << " Sum of 5 foat numbers:" <<fsum
 cout << " Sum of 10 integer nos:" << isum

return 0;

}
```

O/P Sum of 5 float numbers: 16.5
Sum of 10 integer nos: 55

Q3) WAP to implement unary (-) operator when used with the objects so that numeric data members of class is neglected.

```
# include <iostream>
using namespace std;
Class number
{
```

```
int a;
public:
void accept(){
 cout << "a :" << endl;
  cin >> a;

 void display() {

  cout << "a =" << a << endl;
}

  void operator -()
{
    a = -a;
}
};

int main() {

 number n1;
 n1.accept();
 -n1;
 n1. display();

return 0;

}
```

OP
```
a : 5
a = -5
```

Q4) Write a program to implement Unary (++) Operator (for pre & post increment) when used with object so that numeric data member of class is incremented.

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

class Number {
private:
  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 () {
   cout << value << endl;
}
};

int main() {
```

```
Number n(5);

cout << Original : ";
n. display();

cout << " After pre-increment (++n): ";
++n;
n.display();


cout << "After post-increment (n++):";
n++;
n. display ();

return 0;
}
```

OP- Original : 5
After pre-increment (++n): 6
After post-increment (n++): 7

```
int value
class (int v=0)                    // Parameterized
  v = value; value = v;              Const.

void operator ++()
  value ++;

Void operator ++ (int)
  value ++

Class n (5)
n.display →  n++
             ++n
```

12.10.24 Experiment 8

Q1:) WAP to overload the '+' operator so that 2 strings can be ~~concat~~ concatenated eg 'xyz' + 'pqr' then string will be xyzpqr

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

class string1 {
 private:

   string str [80];

 public:
   void accept() {
       cout "Enter a string :";
       cin >> name, str;
   }
     void disp() {

   cout <<"concatinated string :"
   }

   void operator + ( string1 &s1
     {
        strcat (str, s1.str);
     }
   };
   int main ()
```

```
}
    string1  s1, s2;
     s1.accept();
     s2. accept();
      s1+s2  ;
      s1. display();

return 0;
}
```

```
Enter a string : xyz
Enter a string : pqr
Concatinated string : xyzpqr
```

02) Write a program to create base class Ilogin having dm name & password. Delare accept() function virtual. Derive emailLogin & Membershiplogin classes from Ilogin. Display details of employee.

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

class Ilogin {
protected:
    string name, password;
public:
  virtual void accept ()
{
    cout << " Enter name:";
    cin >> name;
```

(creates new version of function (overrides))

```
cout << "Enter pass word:";
 cin >> password;
}

virtual void display ()
 {
    cout << " Name :" << name <<
     " password:" << password << endl;
  }
};

class Emailogin : public Ilogin {
   string email;
public:
     void accep() {
     cout << " Enter email Login name :"
      cin >> name;
      cout << "Enter password :";
      cin >> password;
      cout << "Enter emailID :";
      cin >> email;
}

void display() {
 cout << "Name:" << name << "Passwor
 password << " \n emailID :" <<
  email id << endl;
 }
};

class Membershiplogin : public Ilo
   int memberID;
```

```
public:
  void accept() {
   cout << "Enter membershiplogin name:";
   cin >> name;
   cout << "Enter password:";
   cin >> password;
   cout << "Enter membership ID :";
    cin >> member ID;
}

void display() {
  cout << "Name:" << name << "In Password:"
  << password << " In MembrshipID:" <<
  member ID : " endl;
 }
};

int main ()
{
    Login *ptr;
    Email login e;
    membrshiplogin m;

    ptr = &e;
    ptr -> accept();
    ptr -> display();

    ptr = &m;
    ptr -> accept();
    ptr -> display();

return 0;
}
```

OR

```
Enter email login name: ABC
Enter password: 7989
Enter email ID : abcd@gmail.com.

Name: ABC
Password : 7989
EmailD : abcd@gmail.com.

Enter membership login name : DEF
Enter password : 7264
Enter membership id : 345

Name: DEF
Password: 7264
Membership ID: 345
```

7/10

15.10.25 Experiment 9

* Creating / opening file using open() function:

```
#include <iostream>
#include <fstream>          for file handeling
Using namespace std;
int main() {
                    input & operations
fstream new_file;
new_file.open("new_file", ios::out);
                          open
                    create if absent
if (!new_file)      overrite
{
  cout << "File creation failed";
}
else
{
  cout <<"New file created";
  new_file.close();
}

return 0;
}
```

O/P - New file created.

* Copy content from one file to another:

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

int main() {
```

create & write text
read content & copy

write to file

add @ end

```
ofstream file1 ("file1.txt", ios::app);
if (!file1) {
  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" <<endl;
return 1;
}

if (!file2) {
  cout << "Error opening file2.txt
for writing" <<endl;
return 1;

char ch;          stores one char at a time
while (file1_read.get(ch)) {
  file2.put(ch);
}                 loop till end   read   writes
cout << "Content copied successfully
from file1.txt to file2.txt" <<endl;

file1_read.close();
file2.close();
return 0;
}
```

* Count number of digits & spaces -

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

int main() {
 ifstream inputFile ("input.txt");
 if ( !inputFile){
   cout << " Cannot open input.txt " << endl;
 return 1;
}

char ch ;
int spacecount =0;
int digitcount =0;

while ( inputFile.get(ch)) {
  if (ch = = ' ')    or   isspace (ch)
      Spacecount ++ ;
 else if ( isdigit(ch))
      digitcount ++ ;
}
 inputfile.close();

cout << " Total spaces :" << Spacecount << endl;
cout << " Total digits :" << digitcount ;

return 0;
}
```

17/10.

* Count number of words.

```
# include < iostream>
 # include < fstream>
# include < string>

using namespace std;

int main() {
  string filename;
  cout << " Enter file name :";
  getline ( cin. filename);

  ifstream inFile ( filename);
   if ( ! infile) {
      cerr << "Error opening file:" << filename <<
      return 1;
  }

  string word;
int / size_t wordcount = 0;
  while ( infile >> word ) {
       ++ wordcount ;  extraction
    }

  infile. close();

  cout << total number of words: " <<
  wordcount << endl;

  return 0;
  }
```

31.10.25

Experiment 120

Q1) Sum of array elements using function templ

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

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

return sum → T (datatype of sum)

cout sum → void

```
int main(){
   int intArr [5] = { 2, 4, 6, 8, 10};
   cout << " Sum of int array = " <<
      sumArray (intArr, 5) << endl;

   return 0;
}
```

O/P Sum of int array = 30

Q2) Square function using template Specialization

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

template < class T > void func^ (T) { }

T → return.

```
template <typename T>
T square (const T& value)        // generic
   return value * value;
```

template <> — special

return type — string square < string > (const string & str) {

```
   return str + str;
}
```

concat (str, str)

```
int main() {
   int num = 5;
   string s = "Hello";

   cout << " Square of Integer : " << square(num) <<
   cout << " Square of string : " square(s)
      << s << "\n" = \" <<
      << "\" << endl;
```

O/P Square of integer : 25

Savar of string : " HelloHello

Q3) Write a C++ Program to boild simple calculator using a class template.

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

template <class T>
class Calculator {
   T num1, num2;
```

```
public:
  void acceptTwo(){
   cout << "Enter first number:";
   cin >>num1;
   cout << "Enter second number:";
   cin >> num2;
  }

void acceptOne(){
  cout << "Enter number:";
  cin >> num1;
}

void addition()
{ cout << "Sum =" << num1 + num2 <<endl;}
void subtraction()
{ cout << "Difference =" << num1-num2 << endl;}
void multiplication()
{ cout << " Product=" << num1 * num2 << endl;}
void division()
{ if (num2 != 0) cout << " Quotient ="
 << num1 /num2<<endl;
   else cout << " Error: divided by 0" <<endl;}
void modulus()
{ cout << "Modulus =" << (int)num1 % (int)num2
  << endl; }

void sine_op()
{ cout <<"sin = " << sin (num1) <<endl;}
void cosine_op()
{ cout << "Cos =" << cos (num1) <<endl;}
void tangent_op()
```

```
{ cout << "Tan =" << tan (num1) <<endl;}

void square_root()
{ cout << "Square root =" <<sqrt (num1)
 << endl; }
 void power_op()
{ cout << "Power =" << pow (num1, num2)<<
 endl; }
};

int main(){
 Calculator <double> obj;
 int choice;

 cout <<" Calculator menu:\n";
 cout <<" 1. Addition \n";
 cout <<" 2. Subtraction \n";
 cout <<" 3. Multiplication \n";
 cout <<" 4. Division \n";
 cout <<" 5. Modulus \n",
 cout <<" 6. Sin \n" ;
 cout <<" 7. Cos \n";
 cout <<" 8. Tan \n";
 cout <<" 9. Square Root \n";
 cout <<" 10. Power \n";

 cout << Enter choice :";
 cin >> choice;

 if (choice >=1 && choice<= 5 || choice ==10)
 { obj.acceptTwo();
}
```

```
else
if (choice >= 6 && choice <= 9){
    obj.acceptOne();
}

switch (choice){

    case 1: obj.addition(); break;
    case 2: obj.subtraction(); break;
    case 3: obj.multiplication(); break;
    case 4: obj.division(); break;
    case 5: Obj.modulus(); break;
    case6: obj.sine_op(); break;
    case7: obj.cosin_op(); break;
    case8: Obj.tangent_op(); break;
    case 9: Obj.square_root(); break;
    case 10: obj.power_op(); break;
    default: cout << "Invalid choice!" << endl;
}

return 0;
}
```

O/P - Enter choice: 1
Enter first number: 2
Enter second number: 4
Sum = 6

Q4)

31.10.25

Experiment 11

Q- Write a C++ program to implement generic vectors: modify value of an element, multiply scalar value, to display vector in form(10,20,30...)

```
# 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++){
        ~~cout~~ v[i] = v[i] *10;
    }

    cout << "New Vector:" << endl;
    for (int i=0; i<10; i++){
        cout << v[i] << " " << endl;
    }

return 0;
}
```

O/P Initial vector:

1
2
3
4
5
6
7
8
9
10

Multiply by 10

New vector:

10
20
30
40
50
60
70
80
90
100

—x—

* without using iterator -

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

template <class T > class Vector {
 vector <T > v;
public:
   void createvector (int n){
   cout << "enter :" << n << " elements: \n"
    v.resize(n);
    for ( int i=0 ; i<n; i++)
        cin >>v[i];
   } ;
   void modify (int index, T value){
*   if (index >=0 && index < v.size()
    v [index ] = value ;
    else
      cout << " Invalid ! \n";
   }

   void muliply byscalar (T scalar) {
    For (int i=0 ; i< v.size() ; i++)
       v[i] * = scalar;
   }

void display (){
   cout << " (";
   for ( int i=0 ; i<v.size(); i++){
   cout << v[i];
   if ( i != v.size() - 1)
```

```
    cout << ", ";
}
    cout << " ) \n";
  }
};

int main() {
    vector <int> vec;
    int n, index, newvalue, Scalar;

cout << "Enter size of vector:";
cin >> n;
vec.create vector (n);

cout << "Original vector :";
vec.display();

cout << "\n Enter index & new value to
modify :";
cin >> index >> newvalue;
vec.modify (index, newvalue);

cout << "After modification:";
vec.display();

cout << "Enter scalar to multiply:";
cin >> scalar;
vec.multiply Byscalar (Scalar);

cout << "After multiplication:";
vec.display();

return 0;

}
```

op

```
Enter size of vector: 4
Enter 4 elements:
1
2
3
4

Original vector : Vector elements : 1 2 3 4
Enter index & new value to modify: 1
0
After modification : vector elements : 1 0 3 4
Enter scalar to multiply : 7
After multiplication : Vector elements : 7 0 21 28
```

* with iterator :

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

int main() {
    vector <int> v = { 8, 7, 6, 4, 3, 2, 9, 11, 9};

    cout << "Initial vector :" << endl;
    for (vector <int> :: iterator it = v.begin()
    it != v.end(); ++it) {
    cout << *it << " " << endl;
}
```

```
for (vector<int>::iterator it = v.begin();
  it != v.end(); ++it){
  *it = (*it) * 10;
}

cout << "New Vector:" << endl;
for (vector<int>::iterator it = v.begin();
  it != v.end(); ++it) {
  cout << *it << " " << endl;
}

return 0;
}
```

O/P:

```
Initial vector:
8
7
6
4
3
2
9
11
New vector:
80
70
60
40
30
20
90
110
```

4.11.25

Experiment 12

Q1) Implement Stack using STL

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

int main(){
stack <int> s;
int ch, x;

do {
  cout << "\n1.Push \n2.pop \n3.Top \n4.
       Size \n5.Exit \n";
  cout << "Enter choice:";
  cin >> ch;

  switch (ch) {
    case 1:
    cout << "Enter value:";
    cin >> x;
    s.push(x);
    break;

    case 2:
    if (!s.empty()) {
    s.pop();
    cout << "popped \n";
    } else {
      cout << "stack empty.\n";
    }
    break;
```

s.push(x) s.size()
s.pop()
!s.empty
s.top()

```
case 3:
if (!s.empty()){
 cout << "Top =" << s.top() << endl;
}else{
  cout << "Stack is empty.\n";
}
break;

case 4:
 cout << "size =" << s.size() << endl;
  break;

case 5:
   cout << "Exit.\n";
    break;

default:
  cout << "Invalid.\n";
}
}
while (ch!=5);

return 0;

}
```

027 Implement queue using stl.

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

int main(){
 queue <int>q;
 int ch, x;

 do {
 cout << "\n1. Enqueue 2. dequeue \n3. Front-
 \n4. Size \n5. Exit \n";
 cout << "Enter choice;
 cin >> ch;

 switch (ch) {

 Case 1:
  cout << "Enter value :";
  cin >> x;
  q.push(x);
  break;

  case 2:
  if (!q.empty()) {
  q.pop();
  cout << "Dequeued.\n";
 } else {
    cout << "Queue empty.\n";
 }
 break;
```

```
case 3:
if (!q.empty()) {
  cout << "Front =" << q.front() << endl;
} else {
  cout << "Queue empty.\n";
}
break;

case 4:
cout << size =" q.size() << endl;
break;

case 5:
cout << "Exit.\n";
break;

default:
  cout << "Invalid \n";
}

} while (ch!=5);

return 0;

}
```