Date :

Experiment-1

1) Write a C++ program to declare a class student having data members as roll no and name. Accept and display data for single student.

→
```
#<iosstream>
using namespace std;
class student
{ int roll-no;
  string name;
  public:
  void accept()
  { cout<<"Enter the student name and rollno:";
    cin>>name>>rollno;}
  void disp()
  { count<<"student Name:"<<name;
  count<<"\n Student Roll No:"<<roll no;
      }};
int main ()
{ student s1;
  s1 accept();
  s1 display();
  return 0 ();}
```

Date :

2) Write a C++ code to create a class book having datamembers as book_name, book_price, and book_id. Accept the data for 2 books and display the name of the book having greater price.

→
```
#include <iostream>
using namespace std;
class class_book
  { string book_name;
    int book_id;
    public:
    int book_price;
    void accept()
  {
    cout<<"Enter the book name, price and id :";
  cin>>book_name>>book_price>>book_id;}
    void_disp()
  { count<<"\n Book Name:"<<book_name;
    count<<"\n Book Price:"<<book_price;
    count<<"\n Book ID:"<<book_id;};
    int main()
  {class_book B1, B2,
    B1.accept();
    B2.accept();
```

```
if (B1 book_price > B2 book_price)
 {B1.disp ();
else
 {B2.disp();}
  return 0; }
```

3)
```
#include<iostream>
using namespace std;
class Time
{int H,M,S;
 public:
 void accept()
 {cout<<"Enter the time in hours
  minutes seconds:";
  cin>>H>>M>>S;}
  void calculate()
  {H=H*3600;
  M=M*60;
  S=S+H+M;}
  void disp()
  {Time T1;
  T1.accept();T1.calculate();
  T1.disp()
  return 0;}
```

Date :

Experiment-2

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

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

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

→
```
#include<iostream>
class account
{ public:
int acc_no;
float balance;
void accept()
{ count<<"Enter Account number:";
cin>>acc_no;
count<<"Enter Account Balance:";
cin>>balance;}
void disp()
{ count<<"\n Account Number:"<<acc_no;
count<<"\n Account Balance:"<<balance;}};
int main()
{ account A[10];
int i;
for (i=0; i<10; i++)
{ A[i].accept();}
for (i=0; i<10; i++)
{ if (A[i].balance>=5000)
{A[i].balance=A[i].balance+(0.1*A[i]);
A[i].balance);
A[i].disp();}}
return 0;}
```

3) WAP to declare a class "staff" having data members as name and post. Accept this data for 5 staff and display name of staff who are "HOD".

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

```
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=="HOD")
{S[i].disp();}}
return 0;}
```

Date :

Experiment-3

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

```
#include<iostream>
using namespace std;
class book
{ string booktitle;
  string authorname;
  int price;
public:
  void accept()
{ cout<<"Enter Book Title:";
  getline(cin,booktitle);
  cout<<"Enter Author name";
  getline(cin,authorname);
  cout<<"Enter Book Price:";
  cin>>price;}
  void disp()
{ cout<<"Book Title:"<<booktitle;
  cout<<"\nAuthor Name:"<<authorname;
cout<<"\n Book Price:"<<price;}};
  int main()
{ book B1;
```

Date :

```
book *p;
p=&B2;
p->accept();
p->disp();
return 0;}
```

2) WAP to declare a class 'student' having

→

```
#include<iostream>
using namespace std;
class student
{int roll_no;
float percentage;
public:
void accept()
{count<<"Enter the student Roll No:";
cin>>roll_no;
count<<"Enter student Percentage:";
cin>>percentage;}
void disp()
{this->accept();
cout<<"\n Roll No of the Student:"
<<roll_no;
cout<<"\n Percentage of the student:"
<<percentage;
```

Date :

```
};
int main()
{student S1;
S1.disp();
return 0;}
```

3) Write a program to demonstrate the use of nested class?

→

```
#include<iostream>
using namespace std;
class student
{
private:
    string name;
    int roll_no;
public:
void input student Details()
{
    cout<<"Enter Student's Name:";
    getline(cin,name);
    cout<<"Enter roll number:";
    cin>>roll no;
}
void display student Details()
```

Date :

```
{
    cout << "\n Student Name:" << name <<
endl
cout << "Roll No:" << roll_no << endl;
}
class Marks
{
private:
int science;
int english;
public:
void input Marks()
{
    cout << "Enter marks in science:";
    cin >> science;
    cout << "Enter marks in English:";
    cin >> english;
}
void display Marks()
{
    cout << "Marks in Science:" << science
    << endl;
cout << "Marks is English:" << English <<
    endl;
```

Date :

```
}
};
};
int main()
{
    Student S1;
    Student::Marks M1;
    S1.input Student Details();
    M1.input Marks();
    S1.display Student Details();
    M1.display Marks();
    return 0;
}
```

Q4

21

Date :

Experiment - 4

1) 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 numbers{
private:
int a,b;
public:
int temp;
void accept(){
cout<<"Enter the first number:";
cin>>a;
cout<<"Enter the second number:";
cin>>b;
}
void disp(){
cout<<"\nAfter swapping:"<<endl;
cout<<"First number="<<a<<endl;
cout<<"Second number="<<b;
}
void swap(numbers &n){
n.temp=n.a;
n.a=n.b;
n.b=n.temp;
}
};
int main(){
numbers n;
n.accept();
n.swap(n);
n.disp();
return 0;
}
```

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

→
```
#include <iostream>
using namespace std;
class number{
private:
int a,b,temp;
public:
void accept(){
cout<<"Enter the first number:";
cin>>a;
cout<<"Enter the second number:";
```

```
    cin>>b;
}
void disp(){
    cout<<"\nAfter swapping."<<endl;
    cout<<"First number="<<a<<endl;
    cout<<"Second number="<<b;
}
    friend void swap (number &);
};
void swap (number &n){
    n.temp=n.a;
    n.a=n.b;
    n.b=n.temp;
}
int main(){
    number n;
    n.accept();
    swap(n);
    n.disp();
    return 0;
}
```

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

→

```
#include <iostream>
    using namespace std;
    class num2;
    class num1 {
        private:
        int a;
        public:
        void accep(){
        cout<<"Enter the first number:";
        cin>>a;
    }
    friend void swap(num1 &,num2 &);
};
class num2{
    private:
    int b;
    public:
    void accept(){
cout<<"Enter the second number:";
    cin>>b;
}
friend void swap(num1 &,num2 &);
```

```
};
void swap(num1 &x, num2 &y){
    int temp;
    temp = x.a;

    x.a = y.b;
    y.b = temp;
    cout << "\nAfter swapping:" << endl;
    cout << "First number = " << x.a << endl;
    cout << "Second number = " << y.b << endl;
}
int main(){
    num1 n1;
    num2 n2;
    n1.accept();
    n2.accept();
    swap(n1, n2);
    return 0;
}
```

4) WAP to create two classes Result and Result2 which stores the marks of the students. Read the value of marks for both the class objects and compute the average of two results.

```
#include <iostream>
using namespace std;
class result2;
class result1{
    private:
    string name;
    float marks;
    public:
    void accept(){
    cout << "Enter student name:";
    getline(cin, name);
    cout << "Enter the total marks obtained in first semester out of 1000:";
    cin >> marks;
    }
    friend void average(result1, result2);
};
class result2{
    private:
    float marks;
    public:
    void accept(){
    cout << "Enter the total marks obtained in second semester out of 100:";
```

```
        cin>>marks;
    }
    friend void average(result1,result2);
};

void average(result1 x,result2 y){
    float avg;
    avg=(x.marks+y.marks)/2;
    cout<<"\n Average of both the
    results="<<average
}
int main(){
    result1 r1;
    result2 r2;
    r1.accept();
    r2.accept();
    average(r1,r2);
    return 0;
}
```

5) WAP to find the greatest number among two numbers from two different classes using friend function.

```
#include <iostream>
using namespace std;
class num2;
class num1{
    private:
    int a;
    public:
    void accept(){
    cout<<"Enter the first number:";
    cin>>a;
    }
    friend void grt(num1,num2);
};
class num2{
    private:
    int b;
    public:
    void accept(){
    cout<<"Enter the second number:";
    cin>>b;
    }
    friend void grt(num1,num2);
};
void grt(num1 x, num2 y){
```

```
if (x.a > y.b){
  cout << x.a << "is greater than" << y.b;
}
else{
  cout << y.b << "is greater than" << x.a;
}
}
int main (){
  num1 n1;
  num2 n2;
  n1.accept();
  n2.accept();
  gt(n1,n2);
  return 0;
}
```

Experiment-5

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

```
#include <iostream.h>
  Using namespace std;
   class classB;
   class classA;
   int A;
   Public;
   Void accept(){
   cout <<"Enter value for A:";
    cin>>A;
   }
   friend int sum (class A oA, class B oB);
  } nab;
   Class B {
    int B;
  Public:
   void accept(){
   cout<<"Enter value for A";
     cin>>A;
   }
  friend int sum (class A, class B oB);
```

Date :

```
} na,nb;
  int sum ( class OA, class B OB );
  return OA. + OB. B;
}
int main (){
    na.accept ();
    nb. accept ();
cout <<"Sum:" <<Sum (na, nb) <<endl;
    return 0;
  }
```

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

→
```
#include <iostream>
using namespace std;
    class cube;
    class Box
  {
    double vi
    public:
    void accept () {
```

Date :

```
double w, l, h;
(out <<"Enter length, width, & height of
the box:";
(in>>) l >>w>> h;
v = l*w*h;
}
friend void findgreater (Box b, cube C);
} b;
  class cube {
  double v;
  Public:
      void accept () {
        double s;

(out <<"Enter side length for the cub:";
    (in>) s;
    v = s*s*s;
  }
  friend void find Greater (box B, cube ();
  } C;

width find Greater (Dox, Cube) {
  String res = (b.v > c.v) } "Box" : ((c.v>
b.v)?()
    "cube": "Both equal");
```

Date :

```
cout << " Greater volume " << res << endl;
}
int main(){
  b.accept();
  c.accept();
  find greater or (b,c);
  return 0;
}
```

Date :

Experiment-6

1) Single Inheritance
Create a base class called Person with attributes name and age. Derive a class Student from person that adds an attribute roll Number. Write function to display all details of the student.

→
```
#include <iostream>
using namespace std;
class person
{
  String n;
  int a;
  Protected:
  void accept ()
  {
  cout << "Enter name:";
   (in>>) n;
   cout << "Enter age:";
   (in>>) a;
  }
   void display ()
   {
    cout << "Name:" << n;
    cout << "Age:" << a;
```

```
};
class student : public person
{
    int roll;
    public:
void accept() { person::accept();
    cout << "Enter roll no:";
    cin >> roll;
}
void display()
{ person::display();
    cout << "\n Rollno:" << roll;
}
};
int main()
{
    student s;
    s.accept();
    s.display();
    return 0;
}
```

2) Multiple Inheritance

Create two base classes Academic and sports.

- Academic class contains marks of a student
- Sports class contains sports score.

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

→

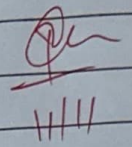
```
#include <iostream>
using namespace std;
class academic
{
  protected
  int marks;
  public:
  void acceptm()
  {
    cout << "Enter marks:";
    cin >> marks;
  }
};
class sports
{
  protected;
  int score;
```

Date :

```
public:
void accepts()
{
cout <<"Enter score:";
cin >> score;
}
};
class result :public academic, public sports
{
public:
void display()
{
int total;
total = marks + score;
cout << "\n Marks:" << marks;
cout << "\n Score:" << score;
cout << "\n total:" << total;
}
};
int main()
{
result r;
r.acceptm();
r.accepts();
```

Date :

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



Date :

Experiment-7

Q- WAP using function overloading in calculating area of laboratory (rectangle) & class room (square).

→
```
#include <iostream>
using namespace std;
class area {
public:
void calc (float len, float breath){
cout << "Area of laboratory:" << len*
breadth << endl;
}
void calc (float side){
cout << "Area of classroom:" << side* side << endl;
}
};
int main(){
area a;
cout << "Laboratory:" << endl;
a.calc (12.5, 10.0);
cout << "Classroom:" << endl;
a.calc (9.0);
return 0;
}
```

Date :

2) WAP in implement the unary ++ generator

→
```
#include <iostream>
using namespace std;
class counter {
int count;
public:
void setdata (int c) {count= c;}
void disp () {cout << "count =" << count
<< endl;}
void operator ++ (int) {count++;}
};
int main(){
counter c;
c.setdata (5);
cout << "After pre-increment"; c.disp();
c++;
cout << "Original"; c.disp();
c++;
cout << "After post increment:";
return 0;
}
```

Date :

Experiment-8

1) Operator overloading

```
#include <iostream>
using namespace std;
class Mystring{
string str;
public:
void setstring(){
cout<<"Enter string:";
cin>>str;
}
void disp(){
cout<<str<<endl;
}
mystring operator+(Mystring s){
mystring temp;
temp.str=str+sttr;
return temp;
}
};
int main(){
Mystring s1,s2,s3;
cout<<"First String";
s1.setstring();
cout<<"second string";
```

Date :

```
s2.setstring();
s3=s1+s2;
cout<<"concatenated string:";
s3.disp()
return 0;
}
```

b) WAP to create a base class. I login having data members names password - Declare accept() function virtual. Desire email login membership login in classes from I login. Display log in details and employee

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

```
}
};
class Gmaillogin: public Ilogin Details";
I Login::accept()
{
void display() override{
  cout << "Email login →";
  Ilogin:: display();
  }
 };

class Membership log in:: public I login{
public:
 void accept() override {
 cout << "Membership login Details";
  I Login :: accept ();
  }
void display() override {
cout << "Membership login →";
 I log in:: display();
 }
};
 int main (){
 Email logina e;
```

```
membership Login m;
  e.accept();
cout << "Displaying Details:";
  e.display();
  m.display();
  return 0;
}
```

Date :

Experiment - 9

1) WAP to copy contents of one file into another. Open "first.txt" in read (ios::in) mode & "second.txt" file in write (ios::app) mode. copy elements of first into second. Assume "first" is already awarded.

→
```
#include <iostream>
#include <fstream>
using namespace std;
int main (){
 if stream fin ("first.txt");
 of stream fout ("second.txt");
if (!fin) {cout<<"first.txt not found";
 return 0; }
string line;
while (getline (fin, line)) {
  fout <<line<<endl;
}
fin.close ();
 fout.close ();
cout <<" Contexts copied successfully";
 return 0;
}
```

Date :

2) WAP to count digits & space using file handling

→
```
#include <iostream>
#include <fstream>
using namespace std;
int main (){
 ifstream fin ("First.txt");
if (!Find ) {cout <<"File not found!\n";
return 0; }
 char ch;
int digits =0, spaces =0;
while (fin.get (ch)) {
 if (is digit (ch)) digits++;
 if (ch == ' ') spaces++;
}
  fin.close ();
cout << "digits": "<<digits <<"spaces" <<
spaces:
  return 0;
      }
```

Date :

c) WAP h count words using file handling

→
```
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
int main(){
ifstream fin("First-txt");
if (!fin){
 cout<<"File not found!");};
 return 0;

 string word;
 int count=0;
while (fin>>word){
 count++;
 }
 fin.close();
cast<<"Total words:" "<<count<<endl;
 return 0;
 }
```

Date :

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

→
```
#include <iostream)
 #include <fstream)
 #include <sstream>
  using namespace std;
  int main() {
  ifstream fin ("First.txt");
   if (!fin) {cout <<"File not found!";
  return 0;}
string word,target;
   int count = 0;
   caust <<"Enter word to search";
   cin>> target;
  while fin word {
  if (sword==target) count++;
  }
   fin.close();
cout <<"The word" <<target <<"occured"
 << count << " times" return 0;
                    }
```

Date :

Experiment-10

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

→
```
#include<iostream>
using namespace std;
 template std;
 T arrsum (T arr[], int n){
  T sum=0
   for (int i=0; i<n:i++)
   sum+= arr[i];
    return sum;
  }
  int main (){
  int ArrI[5]={1,2,3,4,5};
cout<<" Sum of Array: "(<arr sum
(ArrI, 5); }
```

b) WAP of square function using template specification can square of integer no I string. Write a specialized function string.

→
```
#include <iostream>
 #include <string>
  using namespace std;
   template <class T>
T sq (In ){
```

Date :

```
return n*n;
}
 template()
 string sq <string> astrings) {
  return 5+5;
 }
 int main (){
 int i=5;
 string str="ABC";
 cout<<" Square of Integer :"<<sq
 (i)<<endl;
 cout<<" Square of string: "<<sq (str)<<endl;
 }
```

c) Calculator (10, operations)

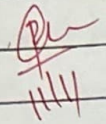
→
```
#include<iostream>
 using namespace std;
  template <class T>
class calc { T num1, num2;
   public:
    calc (Ta,Tb){
    public:
     calc (Ta, tb){
      num1=a;
      num2=b;
```

```
};
void disp(){
cout<<"Addition:"<<num1+num2;
cout<<"Subtraction:"<<num1-num2;
cout<<"Multiplication:"<<num1*num2;
cout<<"Division:"<<num1/num2
cout<<"Modulus:"<<num1%num2
cout<<" Square of Num1:"<<num1*num1;
cout<<"Square of Num2:"<<num2*num2;
cout<<"Average:"<<(num1+num2)/2;
cout<<"Maximum:"<<(num1>num2)?num1:num2);
cout<<"Min:mum"<<(num1<num2)?num1:num2);
}
};
int main(){
int a,b;
cout<<"Enter two integer:";
cin>>a>>b;
calc<int>calc(a,b);
cout<<"Calculable:"<<endl;
calc.disp();
}
```

d) WAP to implement push & pop methods from stack using class template.

```
#include<iostream>
using namespace std;
template<class T>
class stack{T arr[5];int top;
public:
stack(){top=-1;}
void push(T val){if(top==4)cout<<"
stackoverflow";
else
arr[++top]=val;
}
void pop(){if(top==-1)cout<<"Stack underflow";
else
cout<<"Popped"<<arr[top--]<<endl;}
void disp(){cout<<"stack elements:";
for(int i=0;i<=top;i++)
cout<<arr[i]<<" ";
cout<<endl;
}
};
int main(){
stack<int>s;
```

```
s.push(10);
s.push(20);
s.push(30);
s.display();
s.pop();
s.display();
}
```



Experiment - 11

a. WAP hi create a vector modify the value multiply by a scalar value, display the vector.

⇒

```
#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]<<" "<<endl;
    }
    return 0;
}
```

Date :

* Experimen-11 (with iterators)

```
#include <iostream>
#include <vector>
Using namespace std;
int main(){
  vector <int> v = {1,2,3,4,5,6,7,8,9,10};

cout << "Intial vector :" <<endl;
  for (vector <int>::iterator it=v.begin()
; it!=v.end() &++it ){
cout << *it << " ";
 }
 cout<<endl;
cout << "Multiply by 10" <<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 <<" ";
}
  cout <<endl;
      return 0; }
```

Date :

Experiment-12

Q- WAP to implement stack

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

stack <int> stack1;

void disp(){
  stack <int> temp = stack1;
  while (!temp.empty()){
  cout << temp.top() <<" ";
    temp.pop();
  }
   cout <<endl;
}
int main(){
  int num;
  cout<<"Enter a no.:" <<endl;
  cin >> num;

 for (int i=0; i<num; i++){
   int temp;
(cout <<"Enter element at portion" <<(i+1)
<<endl;
```

```
    cin >> (temp);
}
cout << endl;
    cout << "Top most element :" <<
stack.top() << endl;
    cout << endl;
cout << "Stack elements (top to bottom) --->"
<< endl;
 disp()
 cout << endl;
 cout << "Pop Function :" << endl;
 stack1.pop();
  disp();
 stack1.pop();
 disp()
  stack.pop();
  disp();
 }
```

Q - WAP to implement STL Queue

```
#include <iostream>
#include <queue>
using namespace std;
int main(){
queue <int> q;
for (int i=0; i <=10; i++){
q.push(i*10);
}
cout << endl;
cout << "front element " << q.front() << endl;
cout << "Back element" << q.back() << endl;
cout << endl;
q.pop();
cout << "After one pop front= " << q.front() << endl;
cout << endl;
   cout << "Queue elements (front to back:)";
while (!q.empty()){
cout << q.front() << " ";
}
}
```

////