PAGE NO.
DATE: / /

Experiment - 1

Q1) Write a program to declare a class student having data members as roll number. Accept and display data for one object.

```
#include <iostream>
using namespace std;
class student
{
  private:
        string name;
        int roll no;
public :
    void accept()
    {
     cout << " Enter student name" <<endl;
      cin >> name;
      cout << " Enter roll number" <<endl;
      cin >> roll no;
}
  void disp ()
    {
    cout <<" Student name" << name <<endl;
    cout <<" student roll no " << roll no <<endl;
} };
  int main()
  {
   student s1;
   s1.accept ();
   s1. disp ();
    return 0;
}
```

Output

Enter student name
rajas
Enter roll number
37
Student name rajas
Student roll no 37

Q2) Write a program to declare a class book having data members as book id, book name and price. Accept data for 2 books and display name of book having greater price

```
#include <iostream>
using namespace std;
class book
{
  public:
    string name;
    float price;
    int pages;

  void accept()
  {
  cout<<"Enter book name"<<endl;
  cin>>name;
  cout<<"Enter price"<<endl;
  cin>>pages>>price;
  cout<<"Enter no of pages"<<endl;
  cin>>pages;
  }
```

```
    void disp()
{
cout <<"Book name " <<name <<endl;
cout <<"Book price " <<price <<endl;
cout <<" Number of pages" <<pages <<endl;
}
};
int main()
{
  book b1,b2;
b1.accept();
b2.accept();
b1.disp();
b2.disp();
if (b1.price >b2.price) {
cout <<"Book " <<b1.name <<" Is greater in price" <<b1.price;
 }
else {
cout <<"Book " <<b2.name <<" Is greater in price " <<b2.price;
}
return 0;
}
```

Output:

```
Enter book name:
abc
Enter Price
60
Enter no of pages
30
Enter Book name
xyz
Enter price
```

Enter pages
36
Book name abc
Book price 60
Number of pages 36
Book name xyz
Book price 30
Number of pages 36
Book abc is greater in price. 60

eg.) Write a program to declare a class time accept time in HH/MM/SS and display total time in secs.

```
#include <iostream>
using namespace std;
class TimeData
  {
    public:
        float hours;
        float mins;
        float sec;

  void accept()
{
  cout "<<" Enter hours" << endl;
  cin >> hours;
  cout <<" Enter mins "<<endl;
  cin >> mins;
  cout << "Enter sec"  << endl;
  cin >> sec;
}
```

```
void disp()
{
 cout<<"Hours "<<hours<<endl;
 cout<<"Mins "<<mins<<endl;
 cout<<"secs "<<sec<<endl;
}
};
 int main()
{
 int a,b,c,converted;
 TimeData t1;
 t1.accept();
 t1.disp();
 a=t1.mins*60;
 b=t1.hours*3600;
 c=t1.sec;
 converted=a+b+c;
 cout<<" Converted time is "<<converted;
    return 0;
}
```

Output:

```
Enter hours
1
Enter mins
30
Enter sec
30
Hours 1
Mins 30
Sec 30
Converted time is 5430.
```

318

Experiment - 7

1) W.A.P to declare class 'city' having data members as name and population. Accept the data for 5 cities and display name of city having greater pop.

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

class city {
    public:
        string city_name;
        int population;

        void accept() {

    cout<<"Enter name of the city:";
      cin >> city_name;
      cout<<" Enter population:";
       cin>> population;
}
    void display() {
        cout<<"city having highest population:"<<population<<endl;
    }
};
int main() {
  city a[5];
    for(int i=0;i<5;i++) {
        a[i].accept();
};
```

```
int max = 0;
for(int i=1 ; i<5 ; i++) {
    if(a[i].population > a[max].population ) {
    max=i;
}
}
a[max].display();
}
```

Q) W.A.P to declare a class Account having data members as account no, balance. Accept data for 5 members and give interest of 10% to acc having balance greater than 5000.

```
#include <iostream>
using namespace std;
class Account {
        public:
         int account_no;
         float balance;
       void accept() {
        cout << "Enter Account No:";
        cin >> account_no;
        cout << "Enter balance!";
         cin >> balance;
    }
    void display() {
            cout << "Account no: " << account_no << endl;
            cout << "Balance:" << balance << endl;
    }
};
```

```
int main() {
    Account a[5];
    for(int i=0 ; i<5 ; i++) {
      a[i].accept();

    if(a[i].balance >= 5000) {
      a[i].balance += a[i].balance * 0.10;
    }
}
  cout << "Accounts with updated :\n";
    for(int i=0 ; i<5 ; i++) {
      a[i].display();
    }
    return 0;
}
```

Q. WAP to declare a class staff having data members name and post. Accept the data for 5 employees and display the name of the person having post HOD.

```
#include <string.h>
#include <iostream>
using namespace std;
class staff {
    public :
        char name[50], post[50];
    void accept() {
      cout<<" Enter name of employee";
       cin >> name;
       cout << "Enter the post: ";
         cin >>post;
    }
    void display() {
         cout <<"Member having post HOD: "<<name<<endl;
    }
};
int main() {
      staff a[5];
      int i;
    for(i=0 ; i<5;i++) {
      a[i].accept();
    }
    for(i=0 ; i<5 ; i++) {
      if (strcmp (a[i].post ," HOD" )==0) {
          a[i] display;
    }
}
}
```

1318

Experiment-3

#include <iostream>
using

Q1) W.A.P to declare a class 'book' containing data members book title, author and price. Accept data and display this data for one obj using pointer

```
#include <iostream>
using namespace std;
class book {
    string book_title;
    string author_name;
    int price;
    void accept() {
        cout << "Enter title :" << endl;
        cin >> this -> book_title;
        cout << "Enter author name:" << endl;
        cin >> this -> author_name;
        cout << "Enter price :" << endl;
        cin >> this -> price;
    }
    void display() {
        this -> accept()
        cout << "The book title :" << this -> book_title << endl;
        cout << "The author :" << this -> author_name << endl;
        cout << "The price :" << this -> price << endl;
    }
};
```

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

b) W.A.P to declare a class 'student' having data members roll_no and percentage. Using 'this' pointer invoke member functions to accept and display this data for one object of the class.

```
#include <iostream>
#include <string.h>
using namespace std;
class student {
public:
    int roll_no;
    float per;
    void accept() {
        cout << "Enter roll no: " << endl;
        cin >> this -> roll_no;
        cout << "Enter percentage: " << endl;
        cin >> this -> per;
    }
    void display() {
        this -> accept();
        cout << "roll no of student: " << this -> roll_no << endl;
        cout << "percentage of student: " << this -> per << endl;
    }
};
```

```
int main(){
  student s;
  s.display();
}
```

Q) W.A.P to demonstrate nested class

```
#include <iostream>
using namespace std;
class student {
public:
    class student {
    public: int roll_no;
    char name[50];
      int m1;
      int m2;
    void accept() {
    cout<<"Enter roll no:";
     cin>>roll_no;
     cout<<"Enter name:";
     cin>>name;
     cout<<"Enter marks for m1:";
     cin>>m1;
     cout<<"Enter marks for m2:";
      cin>>m2;
    }
    void display() {
     cout<<"Roll no:"<<roll_no<<endl;
     cout<<"Name:"<<name<<endl;
     cout<<"Marks for m1 and m2:"<<m1<<m2<<endl;
    }
};
};
```

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

Q1

13/8

Experiment-4

aim: W.A.P to swap two numbers from same class using object as function argument. Write swap function as member function

Swapping using friend function (2 classes)

```
#include <iostream>
using namespace std;
class n2;
class n1 {
        int a;
        public:
          void accept() {
          cout << "Enter your first number: ";
          cin >> a;
        }
      void display() {
       cout << "First Number:" << a << endl
        }
       friend void swap (n1 &s, n2 &t);
};
class n2 {
        int b;
        public:
         void accept() {
             cout << "Enter 2nd number:"
              cin >> b;
        }
```

```
void display () {
cout << "Second number = " << b << endl;
}
friend void swap (n1 &s, n2 &t)
};
void swap (n1 &s, n2 &t) {
int t = s.a;
s.a = t.b;
t.b = t;
}
int main () {
n1 s;
n2 t;
s.accept ();
t.accept ();
cout << "Before swapping : \n";
s.display ();
t.display ();
swap (s, t);
cout << "After swapping : \n";
s.display ();
t.display ();
}
```

2) Swapping using friend function one class]

```
#include <iostream>
using namespace std;
class M {
    int a,b;
    public:
      void accept () {
      cout << "Enter your numbers:";
      cin >> a >> b;
    }
    friend void swap (M &s);
};
void swap (M &s) {
    int t=s.a;
    s.a=s.b
    s.b=t;
    cout << "First No:" << s.a << endl << "second number " << s.b;
}
  int main () {
      M m;
      s.accept();
      swap (s);
}
```

Q3) W.A.P to create 2 classes Result 1 and Result 2 which stores the marks of the students. Read the value of marks for both the class objects and compute average.

```
#include <iostream>
using namespace std;
class Result 2;
class Result 1 {
        int m1;
        public:
        void accept() {
        cout << "Enter your marks for first subject: ";
        cin >> m1;
        }
        friend void avg(Result 1&s, Result 2&r);
};
class Result 2 {
        int m2;
        public:
        void accept() {
        cout << "Enter your marks for second subject: ";
        cin >> m2;
    }
    friend void avg(Result 1&s, Result 2&r);
};
Void avg (Result 1&s, Result 2&r);
float a(s.m1 + r.m2)/2;
cout << "Average = " << a;
}
int main() {
```

m1.s
m2.t
s.accept();
t.accept();
avg();
}

Q47 WAP to find greatest number among 2 numbers from 2 diff classes using friend function.

```
#include <iostream>
using namespace std;
class n2;
class n1 {
    int a;
    public:
    void accept() {
    cout << "Enter first number:";
     cin >> a;
  }
 friend void greatest (n1 &s, n2 &t);
};
class n2 {
      int b;
      public: void accept() {
      cout << " Enter second number:";
       cin >> b;
     }
   friend void greatest (n1 &s, n2 &t);
};
void greatest (n1 &s, n2 &t) {
      float g = s.a;
       if (s.a > t.b) {
       cout << " Greater number is:" << s.a;
       else {
       cout << "Greater number is:" << t.b;
}
```

```
int main() {
    n1.s;
    n2.t;
    s.accept();
    t.accept();
    greatest(s,t);

}
```

Experiment-5

Write a C++ program to implement types of constructors

a) 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 add {
    int total=0, n, i;
  public:
add() {
    n=5;
}
void calculate() {
    total=0;
    for(i=1;i<=n;i++) {
      total=total+i;
    }
    cout << "Sum of first 5 numbers is:" << total << endl;
}
};
int main() {
  add a1;
  a1.calculate();
  return 0;
}
```

Output

Sum of first 5 numbers is 15:

Copy constructor.

```
#include <iostream>
using namespace std;
class add {
    int total=0, n, i;
public:
    add(int num) {
        n=num;
    }
    add(add &obj) {
        n=obj.n;
    }
    void calculate() {
        total=0;
        for(i=1;i<=n;i++) {
        total=total+i;
    }
        cout<<"Sum of 1 to n numbers is:" <<total<<endl;
    }};
int main() {
    int n;
    cout<<"Enter value of n: ";
    cin>>n;
    add s1(n);
    add s2(6);
    s2.calculate();
    return 0;
}
```

Output: Enter the value of n: 5
Sum of 1 to n numbers is :15

Parameterized constructor

```
#include <iostream>
using namespace std;
class add {
    int total=0,n,i;
  public:
add(int num) {
   n=num;
}
void calculate() {
  total=0;
  for(i=1;i<=n;i++) {
       total=total+i;
  }
  cout<<"sum of 1 to n numbers is:"<<total<<endl;
}};
int main() {
  int n;
  cout<<"Enter the value of n: ";
    cin>>n;
    add s1(n);
     s1.calculate();
     return 0;
  }
```

output:

Enter the value of n: 5

Sum of 1 to n numbers is. 15

b) W.A.P to declare a class student having data members as name and percentage. Write a constructor to initialize these data members. Accept and display data for one student.

Default constructor:

```
#include <iostream>
using namespace std;
class student {
    int per;
    string name;
public:
    student() {
        per=90;
        name="Arnav";
    }
void display() {
    cout<<"Name : "<<name<<endl;
    cout<<"Percentage : "<<per<<endl;
};
int main() {
    int per;
    string name;
    student s1;
    s1.display();
    return 0;
}
```

output:

Name = Arnav
Percentage = 90

Copy constructor

```
#include (iostream)
using namespace std;
class student {
        int per;
        string name;
public:
  student (int p, string n) {
        per = p;
        name = n;
  }
  student (student &s) {
        per = s.per;
        name = s.name;
  }
  void display() {
     cout << "Name : " << name << endl;
     cout << "Percentage: " << per << endl;
  }
};
int main() {
  int per;
  string name;
  cout << "Enter name: ";
    cin >> name;
  cout << "Enter percentage : ";
    cin >> per;
  student s1(per, name);
     student s2(s1);
       s2.display();
       return 0;
}
```

Output:
Enter Name: Aman
Enter Percentage: 56
Name: Aman
Percentage: 56

Parameterized constructor

```
#include <iostream>
using namespace std;
class student {
    int per;
    string name;
    public:
    student (int p, string n) {
        per=p;
        name=n;
    }
    void display() {
        cout << "Name: " << name << endl;
        cout << " percentage: " << per << endl;
    }
};
int main() {
    int per;
    string name;
    cout<<"Enter Name: ";
    cin>> name;
    cout<<"Enter percentage: ";
    cin>>per;
    student s1 (per, name);
    s1.display();
    return 0; }
```

Output: Enter Name: Aman
Enter Percentage: 55
Name: Aman
Percentage: 55

c) W.A.P to demonstrate constructor overloading.

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

Output:

```
Area = 2
Area = 25
Area = 20
```

10/11

Experiment-6

Write C++ programs to implement inheritence

. Single Inheritence

```
#include (iostream)
using namespace std;
class person {
    protected:
        string name;
        int age;
};
class student : protected person {
    int roll_no;
    public:
    void accept() {
    cout<<"Enter name : ";
        cin>>name;
    cout<<" Enter age : ";
    cin>>age;
    cout<<"Enter roll_no :";
    cin >>roll_no;
}
    void display() {
    cout<<"Name : "<<name<<endl;
    cout<<" Age: "<<age<<endl;
    cout<<"Roll no: "<<roll_no<<endl;
};
    int main() {
        student s;
        s.accept();
        s.display();
        return 0; }
```

Output : Enter Name: Awati
Enter age: 17
Enter roll no: 36
Name : Awati
Age : 17
Roll_no : 36

- Multiple inheritence.

```
#include <iostream>
using namespace std;
class academic {
    protected:
    string name;
    int marks;
};
class sports {
    protected:
    int score;
};
class student : protected academic , protected sports {
    int total;
    public:
    void accept() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter marks: ";
        cin >> marks;
        cout << "Enter score: ";
        cin >> score;
        total = marks + score;
}
```

```
void display() {
cout << "Name : " << name << endl;
cout << "Marks : " << marks << endl;
cout << "Score : " << score << endl;
cout << "Total : " << total << endl;
}};
int main() {
    student s;
    s.accept();
    s.display();
    return 0;
}
```

Output

```
Enter name: Rajas
Enter marks: 99
Enter score: 1
Name: Rajas
Marks: 99
Score: 1
Total 100
```

3) Multilevel inheritence.

```
#include <iostream>
using namespace std;
class vehicle {
    public:
     string model;
     string brand;
};
class car : public vehicle {
    protected:
     int type;
};
class electric car : public car {
   public:
   int battery_capacity;
    void accept() {
        cout << "Enter Model : " ;
        cin >> model;
        cout << "Enter brand : ";
        cin >> brand;
        cout "Enter type(1 for sedan , 2 for SUV ) : ";
        cin >> type;
        cout << Enter battery capacity (in KWh ) : ";
        cin >> battery_capacity;
    }
  void display() {
     cout << "Model : " << model << endl ;
     cout << "Brand : " << brand << endl;
     cout << " Type : " << (type == 1 ? "Sedan " : "SUV ") << endl;
     cout << "Battery Capacity : " << battery_capacity << "Kwh " << endl;
}};
```

```
int main() {
    ElectricCar c;
    c.accept();
    c.display();
    return 0;
}
```

Output:

```
Enter Model : M++
Enter brand: XYZ
Enter type (1 for Sedan, 2 for SUV): 2
Enter Battery Capacity (in Kwh): 44
Model : bmw M++
Brand : XYZ
Type : SUV
Battery Capacity : 44 Kwh
```

4) Hierarchical inheritance

```
#include <iostream>
using namespace std;
class employee {
    protected:
        string name;
        int id;
};
class manager : public employee {
    private:
        string dep;
};
class Developer : public employee {
  public:
    string lang;
    string dep;
  void accept() {
    cout<<"Enter emp name:";
      cin>>name;
      cout<<"Enter emp ID:";
        cin>>id;
      cout<<"Enter programming language:";
        cin>>lang;
        cout<<"Enter department";
          cin>>dep;
}
void display() {
  cout<<"Name:"<<name<<endl;
  cout<<"ID:"<<id<<endl;
  cout<<"Programming Language:"<<lang<<endl;
  cout<<"Department:"<<dep<<endl;
}};
```

```
int main() {
  Developer d;
  d.accept();
  d.display();
  return 0;
}
```

Output:

Enter emp name: arnav
Enter emp ID : 001
Enter Programming language : C++
Enter department: Technical
Name: Arnav
ID: 1
Programming Language: C++
Department: Technical

5.) Hybrid Inheritence

```
#include <iostream>
using namespace std;
class teacher {
    private:
    string name;
    string age;
    public:
     void accept() {
      cout<<" Enter name : ";
       cin>>name;
      cout<<" Enter age : ";
       cin>>age;
    }
     void display() {
      cout<<"Name : " << name <<endl;
      cout<<" Age : " << age <<endl;
};
class student : protected teacher {
      private:
        int roll;
        string name;
        public:
         void accept() {
          cout<<"Enter student name : ";
           cin >> name;
          cout<<"Enter roll number : ";
           cin>> roll;
    }
      void display() {
        Cout<<" Student Name : " <<name<<endl;
        cout<<" Roll number : " << roll <<endl; };
```

```
class academics {
    private:
    int total;
    public:
    void total marks() {
        total = m1+m2+m3
};
class marks : protected academics {
    private:
    int m1, m2, m3;
    public:
    void maccept() {
        cout<<"Enter marks of 3 subjects : ";
        cin>>m1>>m2>>m3;
    }
    void mdisplay() {
        cout<<"Marks in subject 1 :"<<m1<<endl;
        cout<<" Marks in subject 2 :"<<m2<<endl;
        cout<<"Marks in subject 3 :"<<m3<<endl;
};
class academics : protected marks {
        private:
        int total;
        public:
        void total marks() {
            total = m1+m2+m3;
            cout<<"Total Marks :"<<total<<endl;
};
```

```
int main() {
    teacher t;
    t.accept();
    t.display();
    student s;
    s.saccept();
    s.sdisplay();
    marks m;
    m.maccept();
    m.mdisplay();
    academics ac;
    ac.totalmarks();
        return 0;
}
```

Output

```
Enter name    : abc
Enter age     : 12
  Name: abc
  age: 12
Enter student name : abc
"    roll number : 12
Student name:
  roll number : 12
Enter marks of 3 subjects : 11 12 13
Marks in subject 1 : 11
"    "    2 : 12
"    "    3 : 13
```

Q. W.A.P to implement virtual base class Assume suitable data.

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

class Person {
  public:
    string name;

  void getName () {
  cout << "Enter name: ";
    cin>> name;
  }
};
class student : virtual public person {
    public:
      int roll no;
      void get roll () {
      cout << " Enter roll no: ";
          cin>> roll no;
    }
    };
class Exam : virtual public person {
  public :
      float marks;
      void get marks () {
      cout << " Enter marks";
        cin>> marks;
      }
    };
```

```
class result : public student , public exam {
  public :
    void display () {
    cout <<"Name :" << name <<" Roll no :" << rollno <<"marks:" << marks <<
        endl;
    }
};
int main() {
    Result v;
   v.getname()
   v.getroll()
    v.getmarks();
    v.display();
    return 0;
}
```

Output

Enter name : Sneha
Enter roll no : 32
Enter marks : 92.5
Name : Sneha
Roll no : 32
Marks : 92.5

10.11

Experiment-7

a): W.A.P Using function overloading to calculate the area of a laboratory (which is rectangular in shape) and area of classroom (which is square in shape)

```
#include (iostream)
using namespace std;
class xyz {
public:
    int length, breadth, side, area;
    int calculate area(int length, int breadth) {
        return length * breadth;
    }
    int calculate area (int side) {
        return side * side;
    }
};
int main() {
    xyz a1;
    int length, breadth, side;
    cout << "Enter length and breadth of rectangle: ";
    cin >> length >> breadth;
    cout << "Enter side of square: ";
    cin >> side;
    cout << "Area of rectangle : " << a1.calculate area (length, breadth)
    <<endl;
    cout << "Area of square : " << a1.calculate area(side) << endl;
```

O/P.

b)

O/P:1
Enter length and breadth of rectangle: 11 12
Enter side of square: 11
Area of rectangle: 132
Area of square: 121

b) W.A.P to calculate using function overloading to calculate the sum of 5 float values and sum of 10 integer values

```
#include <iostream>
Using namespace std;
  class xyz {
  public:
  float Calculate( float n1, float n2, float n3, float n4, float n5) {
        return n1+n2+n3+n4+n5;
    }
    int calculate(int num[10]) {
        int sum=0;
        for(int i=0; i<10; i++) {
            sum = sum+ num[i];
        }
        return sum;
  }
};
int main() {
xyz a1;
cout<<"Enter 5 float numbers:";
float n1, n2, n3, n4, n5;
cin>> n1 >>n2 >> n3 >>n4 >>n5;
cout<<"sum of float numbers:"<< a1.calculate(n1,n2,n3,n4,n5)<<endl;
 cout<<"Enter 10 integer numbers:";
```

int num[10];
for(int i=0; i<10; i++) { }
cin>>num[i];
}
cout<<"Sum of integer numbers:"<<a1.calculate(num)<<endl;
return 0;
}

O/P.

Enter 5 float numbers: 1.1, 2.2, 3.3, 4.4, 5.5
Sum of float numbers: 16.5
Enter 10 integer numbers: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
sum of integer numbers = 55

c) W.A.P to implement Unary - operator when used with the object so that the numeric data member of the class is negated.

→
```
#include <iostream>
using namespace std;
class Number {
    int a;
    public:
    void accept() {
        cout << "Enter an integer : ";
        cin >> a;
    }
    void display() {
        cout << "The integer is : " << a << endl;
    }
    void operator- () {
        a = -a;
    }
};
int main() {
    Number n;
    n.accept();
    -n;
    n.display();
    return 0;
}
```

OIP:-

Enter an integer : 5
The integer is : -5

23 ~~Write~~ W.A.P to implement the Unary ++ operator (for the increment and post Increment) when used with the object so that the numeric data member of the class is Incremented

```
#include <iostream>
using namespace std;
class Number {
    int n = 10;
public:
    void display() {
        cout << "The number is : " << n << endl;
    }
    void operator ++() {
        ++n;
    }
    void operator ++ (int) {
        n++;
    }};
int main() {
    number n1;
    ++n1;
    n1.display();
    n1++;
    n1.display();
    return 0;
    }
```

The number is: 11
The number is :12

Experiment-8

a) W.A.P to overload the '+' operator so that two strings can be concatenated. Eg "xyz"+"pqr" then output will be "xyzpqr"

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

output:

abcxyz

b) W.A.P to create a base class Ilogin having data members name and password. Declare accept() function virtual. Derive Emaillogin and Membership login classes from Ilogin. Display Email login details and membership login details of the employees.

```
#include <iostream>
using namespace std;
class Ilogin {
  public:
    string name, pass;
    virtual void accept() {
      cout<<"Enter name:";
      cin>>name;
      cout<<"Enter password:";
      cin>>pass;
    }
    virtual void display() {
      cout<<"Name:"<<name<<" Password:"<<pass<<endl;
    };
class Emaillogin : public Ilogin {
    public:
    void display() override {
    cout<<"Email login →" <<name<<"password "<<
      password}};
class Membership login : public Ilogin {
  public:
    void display() override {
    cout<<"Membership login →"<<"name"<<"pass"<<endl;
    };
```

```
int main ()
{
Login *p;
Email login e;
Membership login m;
p -> &e;
p -> accept();
p -> display()
p = &m;
p -> accept();
p -> display();
return 0;
}
```

Experiment-9

a) W.A.P to copy the contents of one file into another. Open "First.txt" in read (ios::in) mode and "Second.txt" file in write (ios::out) mode. Copy the contents of "First.txt" into "Second.txt". Assume "First.txt" is already created.

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

int main() {
    ifstream fin("first.txt");
    ofstream fout("second.txt");
    char ch;
    while(fin.get(ch)) {
        fout.put(ch);
    }
    cout<<"File copied successfully!";
    fin.close();
    fout.close();
    return 0;
}
```

b) W.A.P to Count Digits and Spaces using File Handling

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

int main() {
ifstream fin("Data.txt");
  char ch;
int digits = 0, spaces = 0;
while (fin.get(ch)) {
      if (isdigit(ch)) digits++;
      if (ch == ' ') spaces++;
  cout << "Digits " << digits << endl;
  cout << "Spaces" << spaces << endl;
      fin.close();
    return 0;
}
```

(7) W.A.P to Count Words using File Handling

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

int main() {
ifstream fin("Data.txt");
string word;
  int count=0;

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

d) W.A.P to count occurence of a given word using file handling

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

int main() {
ifstream fin("Data.txt");
string word, target;
int count = 0;
cout << "Enter word to search: ";
   cin >> target;
while (fin >> word) {
  if (word == target) {
     cout count++; }}
cout << " Occurence of " << target << ": " << count. << endl;
   fin.close();
      return 0;
  }
```

10/11

Experiment - 10

a:) W.A.P to find Sum of ~~Array~~ Array elements using function template (eg. Pass Integer, Float and Double array of 10 elements)

```
#include <iostream>
using namespace std;
template <typename T>
T sum (T a[], int n){
    T s = 0;
    for (int i=0; i<n; i++)
        s = s+a[i];
        return s;
}
int main(){
    int i[10] = {1,2,3,4,5,6,7,8,9,10};
    float f[10] = {1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.1};
    cout << sum(i,10) << endl;
    cout << sum(f,10) << endl;
    return 0;
}
```

O/P.

55

59.6

b) W.A.P of square functions using template specialization
- calculate the square of int no. and a string. (Square of string is nothing but concatenation of string with itself).
- Write a specialized function for the square of a string

```
#include <iostream>
#include <string>
using namespace std;
template <typename T>
T square (T x) {
    return x*x;
}
template <>
string square <string> (string s) {
    return s+s;
}
int main() {
cout << square (5) << endl;
cout << square <string ("Aaa")) << endl;

    return 0;
}
```

O/P
25
AaaAaa

6) W.A.P to build Simple Calculator using a class template

```
#include <iostream>
using namespace std;
template <typename T>
class Calculator {
  public:
    T add (T a, T b)
    {
     return a+b;
    }
    T sub (T a, T b)
    {
      return a-b;
    }
    T mul (T a, T b)
    {
     return a*b;
    }
    T div (T a, T b)
    {
      if (b==0)
        return 0;
        else
        a/b;
    }
};
```

```
int main(){
    Calculator<int> c;
    int a,b, ch;
    while(1){
      cout<<"Enter choice:";
       cin>>ch;
      if(ch==0)
       break;
      cin>>a>>b;
      switch(ch)  {
        case 1:
        cout<< c.add(a,b) <<endl;
        break;
        case 2;
        cout<< c.sub(a,b) <<endl;
        break;
        case 3:
         cout<< c.mul(a,b) <<endl;
        break;
        case 4:
          cout<< c.div(a,b) <<endl;
          break;
```

```
    }
  }
}
```

O/p.

```
Enter choice : 3
3
3
9
```

Q) W.A.P to implement push and pop methods from stack using class template

```
#include <iostream>
using namespace std;
template <typename T>
class Stack {
    T arr[10];
    int top;
public:
    Stack()
{ top=-1; }
void push(T val) {
    if(top<9)
        arr[++top]=val;
}
T pop {
  if (top==-1)
    return 0;
    else
      return arr[top--];
}};
int main() {
  Stack <int> s;
    s.push(10);
    s.push(20);
    cout << s.pop() <<endl;
    cout << s.pop() <<endl;
      return 0;
  }
```

O/P: 20

Experiment-11

To create a vector.

a) To modify the value of a given element

```
#include <iostream>
template <typename T>
    class vector {
      T a[100];
        int size;
        public:
      vector(int s): size(s) { }
       void set(int i, T val) {
        if (i>=0 && i<size)
            a[i]=val;
            else
            cout<<"invalid";
        }
        T get(int i) {
          if (i>=0 && i<size)
              return a[i];
              cout<<"invalid";
               return T;
          }
    void display() {
        for(int i=0;i<si;i++)
        cout<<a[i]<<" ";
          cout<<endl;
      }
    };
```

```
int main() {
    vector<int> v(5);
    for(int i=0; i<5; i++) {
        v.set(i, i*10);
    }
    v.display();
    v.set(2, 99);
    cout << "After modification";
    v.display();
    return 0;
}
```

O/P:

0 10 20 30 40

After modification: 0 10 99 30 40

b) To multiply by a scalar value

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

O/P

3 6 9 12 15

c.) To display the vector in the form (10,20,30,....)

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
vector<int> vec = {10,20,30,40,50};
    cout<<"(";
    for(int i=0;i<vec.size();i++) {
        cout<<vec[i];
        if(i!=vec.size()-1)
        cout<<",";
    }
    cout<<")"<<endl;
    return 0;
    }
```

O/P

(10,20,30,40,50)

10/11

Experiment-12.

Write a C++ program using STL

a) Implement stack

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

Output:

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

b) Implement queue

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
    queue<int> q;
    q.push(10);
    q.push(20);
    q.pop();
    q.push(30);

cout << "front elements : " << q.front() << endl;
cout << " Back elements = " << q.back() << endl;
    q.pop();
cout << " After pop operations " << endl;
cout << " front element " << q.front() << endl;
cout << "queue size" << q.size() << endl;

    if (q.empty) {
    cout << "Queue is empty :" << endl;
    } else {
    cout << " Queue is not empty : " << endl;
    }
    return 0;
}
```

O/P

front element : 10
back " : 30
after pop :
front element : 20
queue : 2
Queue is not empty

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

struct person {
    string name;
    int age;
    Person(string n, int a): name(n), age(a) { }
    };
    int compare_age (const person &p1, const person &p2) {
        return (p1.age < p2.age) ? 1:0;
    }

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

    sort (people.begin(), people.end()) {
        [] (const person & a, const Person &b) {
                return compare_age (a, b); }

    cout << "Sorted records by age : \n ";
            for(auto &p: people) {
        cout << p.name " - " << p.age << "\n";
        }
    int sample_age = 28;
        int found_index = -1;
for(int i=0; i<(int) people.size(); i++) {
```

```
if (people[i].age = search_age?1:0) {
        found_index = i;
    break;
  }
}
if (found_index != -1)
cout << "person with age " << search_age << endl;
else
cout << " person with age " << search_age << "not found";
        return 0;
    }
```

O/P

Sorted records by Age:

Bob = 25

David = 28

Alice = 30

Charlie = 35.

10/11