Experiment : 1

1] Write a program to declare a class student having data member as roll number, name accept and display data for one object

```
#include <iostream>
using namespace std;
class student
{
int roll_no;
string name;
public:
void accept ()
{
cout <<"Enter roll number: "
cin >>roll no;
cout <<"Enter name:";
cin >>name;
}
void display ()
{
cout <<"\n student Roll No :"<< roll_no.
cout <<"\n Name :" << name << endl;
}
};
int main ()
{
student s;
S.accept ();
S.display ();
return 0;
}.
```

Output :
Student Roll No : 27
Name : Advait

2] Write a program to declare a class group having data members as : book name, price and no of pages. Accept this 'data' for 2 object and display name of book having greater price.

```
#include <iostream>
using namespace std;
class Book Book
{
string book_name;
float price;
int page;
public:
void accept()
{
cout << "Enter book name : ";
cin >> book_name;
cout << "Enter price : ";
cin >> price;
cout << "Enter number of pages : ";
cin >> pages;
}
float get price()
{
return price
}
```

```
string getName()
{
return book_name;
}
};
int main()
{
Book b1, b2;
b1.accept();
b2.accept();
if (b1.getprice() > b2.getprice())
cout << "\n Book with greater price:"
     << b1.getName() << endl;
else
cout << "\n Book with greater price:"
     << b2.getName() << endl;
return 0;
}
```

Output

```
Enter book name: Cpp Basics
Enter price: 300
Enter number of pages: 250

Enter book name: Java
Enter price: 450
Enter number of pages: 300

Book with greater price: Java
```

3] Write a program to declare a class time accept the time in HH:MM:SS and convert it into total second and display

```
#include <iostream>
using namespace std
class Time
{
int hours ; minutes , seconds
public;
void accept ()
{
cout << "Enter time in HH:MM:SS format ;";
cin >> hours >> minutes >> seconds;
}
void convert and display
{
int total_seconds = hours *3600 + minutes*60+sec
cout << "Total time in seconds :" << total_seconds
}
};
int main ()
{
Time t;
t. accept ()
t, convert and display ();
return 0
}
```

30/7/25

Output
Enter time in HH:MM:SS format : 1:30:45.
Total time in second ; 5445

Experiment : 2

Q] W.A.P to declare a '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
{
string name;
int population;
public
void accept()
{
cout<<"Enter city name:";
cin>>name
cout<<"Enter population:";
cin>>population;
}
void display()
{
cout<<"city:"<<name
cout<<"population:"<<population
}
};
int main()
{
city c[5];
for(int i=0;i<5;i++)
```

```
{
cout << "Enter details for city" << i+1;
c[i].accept();
}
int max=0;
for(int i=1; i<5; i++)
{
if(c[i].population > cities c[max].population)
max=i;
}
cout << "city with highest population:"
c[max].display();
return 0;
}
```

Output

Enter details of city : 1
Enter name of city : pune
Enter the population of city : 10000
Enter the details of city : 2
Enter the name of city : Mumbai
Enter the population of city : 175000
Enter the details of city : 3
Enter name of the city : NYC
Enter the population of city : 17772
Enter the details of city : 4
Enter name of the city : los angle
Enter the population of city : 10051
Enter the details of city : 5
Enter name of the city : Tokyo
Enter population of the city : 7000

City with highest population is Mumbai.

2. Write a program to declare a class "account" having data members as acc no and balance. Read the data for 10 accounts and give interest of 10% where balance is equal or greater than 5000 and display them.

```
#include <iostream>
using namespace std;
class account
{
public:
int acc no;
float balance;
void accept()
{
cout << "Enter account number";
cin >> acc no;
cout << "Enter balance:";
cin >> balance;
}
void interest()
{
if (balance >= 5000)
   balance += balance * 0.10;
}
void display()
cout << "Account No:" << acc no
cout << "Balance:" << balance
}
};
int main()
{
```

```
account acc[10];
for(int i=0; i<10; i++)
{
cout << "Account" << i+1 << " " << endl;
acc[i].accept();
}
for(int i=0; i<10; i++)
{
acc[i].intrest();
}
cout << "Account details after applying intrest:";
for(int i=0; i<10; i++)
{
acc[i].display();
}
return 0;
}
```

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

```
#include <iostream>
#include <string.h>
using namespace std;
class staff
{
Public
char name[50], post[50];
void accept()
{
cout << "Enter name and post:";
cin >> name >> post;
}
void display()
{
cout << "HOD are:" << name << endl;
}
};
int main()
{
staff s[5];
for (int i=0; i<5; i++)
{
s[i].accept();
}
for (int i=0; i<5; i++)
```

```
{
if (strcmp(S[i].post,"HOD")==0)
{
S[i].display();
}
}
}
```

Output:

Staff 1 name: John_Snow
Staff 1 post: HOD

Staff 2 name: Waiter_White
Staff 2 post: Janitor

Staff 3 name: Bob
Staff 3 post: Assistant

Staff 4 name: David
Staff 4 post: Proffesor

Staff 5 name: Sean
Staff 5 post: Lecturer

Staff members who are HOD:
John_Snow.

30/7/25

2] Output:

```
Enter account number: 201
Enter balance: 4000
Enter account number: 202
Enter balance: 5000
Enter account number: 203
Enter balance: 2500
Enter account number: 204
Enter balance: 6000
Enter account number: 205
Enter balance: 7000
Enter account number: 206
Enter balance: 3000
Enter account number: 207
Enter balance: 8000
Enter account number: 208
Enter balance: 1000
Enter account number: 209
Enter balance: 12000
Enter account number: 210
Enter balance: 14000
```

Account details after applying interest
Account no: 201
Balance: 4400
Account no: 202
Balance: 5500
Account no: 203
Balance: 2750
Account no: 204
Balance: 6600
Account no: 205
Balance: 7700
Account no: 206
Balance: 3300
Account no: 207
Balance: 8800
Account no: 208
Balance: 1100
Account no: 209
Balance: 13200
Account no: 210
Balance: 15400

Qn
30/7/25

Experiment : 3.

1] WAP 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 the object.

```
#include <iostream>
using namespace std;
class Book
{
String book_title;
string author_name;
int price;

public:
void accept()
{
cout << "Enter the book title:";
cin >> book_title;
cout << "Enter the author name:";
cin >> author_name;
cout << "Enter the price of the book:";
cin >> price;
}
void display()
{
cout << "Book Title:" << book_title << endl;
cout << "Author Name:" << author_name << endl;
cout << "Price:" << price << endl;
}
```

```
int main ()
{
Book b;
Book *p = &b;
p -> accept ();
p -> display ();
return 0;
}
```

Output.

Q WAP to declare a class 'student' having data members roll-no and percentage. Using 'this' pointer invoke member function to accept and display this data for one object of a class.

```
#include<iostream>
using namespace std;
class student
{
public;
int roll_no;
float perc;
void accept()
{
cout<<"enter roll no"<<endl;
cin>>this->roll_no;
cout<<"enter percentage;"<<endl;
cin>>this->perc;
}
void display()
{
this->accept();
cout<<"the roll no is -"<<this->roll_no;
cout<<"the percentage is :-"<<this->perc;
}
};
int main()
{
student s;
s.display;
}
```

2] WAP to demonstrate the use of nested class

```
#include <iostream>
using namespace std;
class demo1
{
public:
class demo2
{
public:
int roll no;
char name[50];
void accept ()
{
cout << "enter roll no:-";
cin >> roll no;
cout << "enter the name:-";
cin>>name;
}
void display (){
cout << "the roll no is;" << roll no << endl;
cout << "the name is:-" << name << endl;
}
};
};
int main ()
{
demo1::demo2
student.accept ();
student.display ();
}
```

Q
1/18

Experiment : 4.

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

class result2;
class result1
{
int m1;
public:
void accept1()
{
cout<<"Enter marks of first student:";
cin>>m1;
}
friend void calculate(result1 p, result2 q);
};
class result2
{
int m2;
public:
void accept2()
{
cout<<"Enter marks of second student:";
cin>>m2;
}
friend void calculate(result1 p; result2 q);
};
void calculate(result1 p, result2 q)
{
int total=(p.m1+q.m2)/2;
cout<<"total marks of both students:"<<total;
}
```

```
int main ()
{
result, k;
result2 b;
k.accept1.();
b.accept2 ();
calculate (k, b);
}
```

2] Swapping 2 nos using without friend function object as function argument.

```
#include <iostream>
using namespace std;
class demo
{
public:
int p, q;
void accept ()
{
cout << "enter 2 nos :-" << endl;
cin >> p >> q;
}
void display ()
{
cout << "after swapping :-" << "value of p=" << p;
cout << "after swapping :-" << "value of q=" << q;
}
```

```
void swap (demo & t)
{
int temp = t.p;
t.p = t.q;
t.q = temp;
}
};
int main ()
{
demo K;
K.accept ();
K.swap ();
K.display ();
}
```

3] WAP to find the greatest number among 2 number from 2 different classes using friend function.

```
#include <iostream>
using namespace std;
class B;
class A
{
public:
int num1;
void accept1()
{
cout << "enter first number" << endl;
cin >> num1;
}
friend void greatest num (A p, B q);
};
class B
{
public:
int num2;
void accept2()
{
cout << "enter second number" << endl;
cin >> num2;
}
friend void greatest num (A p, B q);
};
```

```
void greatest num (A p, B q)
{
if (p.num1 > q.num2)
{
cout << "greatest number :-" << p.num1 << endl;
}
else
{
cout << "greatest number :-" << q.num2 << endl;
}
}
int main ()
{
A k;
B f;
k.accept1 ();
f.accept2 ();
greatest num (k, f);
}
```

Q] Swapping 2 number from same class by using concept of friend function.

```
#include <iostream>
using namespace std;
class demo
{
int a, b;
public:
void accept ()
{
cout << "enter 2 numbers:-" << endl;
cin >> a >> b;
}
void display ()
{
cout << "value of a:-" << a;
cout << "value of b:-" << b;
}
friend void swap num S.(demo & t);
};
void swap nums (demo & t)
{
int temp = t.a;
t.a = t.b;
t.b = temp;
}
int main ()
{
```

```
demo K;
K.accept();
swap nums(K);
K.display();
}
```

5] WAP to swap two numbers of different class using concept of friend function

```
#include<iostream>
using namespace std;
class B;
class A
{
int a;
public:
void accept()
{
cout<<"enter a:-"<<endl;
cin>>a;
}
void display()
{
cout<<"value of a:-"<<a;
}
friend void swap numbers(A&P, B&q);
};
class B
{
int b;
public:
```

```
void accept2()
{
cout << "enter b:-" << endl;
cin >> b;
}
void display2()
{
cout << "value of b:-" << b;
}
friend void swapnumbers(A &P, B &a);
};
void swapnumbers(A &P, B &a)
{
int temp = p.a;
p.a = a.b;
a.b = temp;
}
int main()
{
A k;
B b;
k.accept1();
b.accept2();
swapnumbers(k, b);
k.display1();
b.display2();
}
```

Q1

14/8

6] Create two classes, class A and class B, each with private integer write a friend function sum() that can access private data from both classes and return the sum.

```
#include <iostream>
using namespace std;
class B;
class A;
{
int a;
public:
void accept ()
{
cout << "Enter the first number:" << endl;
cin >> a;
}
friend sum (Ap, Bf);
};
class B
{
int b;
public:
void accept ()
{
cout << "Enter the second number:" << endl;
cin >> b;
}
Friend int sum (Ap, Bf);
};
```

```
int sum (A p, B q)
{
int sum;
sum = p.a + q.b;
return sum;
}

int main ()
{
A k;
B f;
k.accept ();
f.accept ();
cout << "The sum of 2 numbers is:" << sum (k,
                                        (k, f);
}
```

7] WAP with a class number that contains a private integer. Use a friend function swap number (number &, number &) to swap the private value of two number objects.

```
#include <iostream>
using namespace std;
class A
{
int a, b;
public:
void accept ()
{
cout << "Enter two number: " << endl;
cin >> a >> b;
}
friend void swapnumber (A & B);
void display ()
{
cout << "Value of a: " << a;
cout << "Value of b; " << b;
}
friend void swapnumber (A & B);
};
void swapnumber (A & B)
{
int temp = t.a;
t.a = t.b;
t.b = temp;
}
```

```
int main ()
{
A k;
k.accept ();
swap numbers (k);
k display ();

}
```

8] Define two classes Box and cube, each having a private volume. Write a friend function find greater (Box, Cube) that determine which object has a larger volume.

```
#include <iostream>
using namespace std;
class cube;
class box;
{
int l, b, h, v;
public:
void accept ()
{
cout << "Enter the dimensions of the box";
cin >> l >> b >> h;
}
friend void greater volume (box k, cube q);
};
```

```
class cube
{
int side, v2;
public:
void accept2()
{
cout << "Enter the dimensions of the cube";
cin >> side;
}
friend void greater volume (box p, cube q);
};

void greater volume (box p, cube q)
{
p.v1 = p.l.p.b;
q.v2 = q.side * q.side * q.side;
if (p.v1 > q.v2)
{
cout << "The box having greater volume is:"
     << p.v1 << endl;
}
else
{
cout << "The box having greater volume is:"
     << q.v2 << endl;
}
}
```

```
int main()
{
box k;
cube b;
k.accept1();
b.accept2();
greater volume(k, b);
}
```

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

```
#include <iostream>
using namespace std;
class cmplx
{
int re, im;
public:
void accept()
{
cout << "Enter the real and imaginary parts";
cin >> re >> im;
}
friend cmplx *add cmplx(cmplx *C1, cmplx *C2
                        cmplx result);
void display()
{
cout << re << "+" << im << "i";
}
```

```
friend cmplx *add cmplx *(cmplx *C1, cmplx *C2,
                          cmplx *result);
};
cmplx *add cmplx::(cmplx C1, cmplx C2,
                   cmplx *result);
result re = C1.re + C2.re;
result.im = C1.im + C2.im;
return result;
};
int main()
{
cmplx k, b, r;
k.accept();
b.accept();
r = add cmplx(k, b, r);
r.display();
}
```

18. Create a class student with private data members: name and three subject marks. write a friend function calculate average (student) that claculate and display the average marks.

```
#include <iostream>
using namespace std;
class student
{
char name [50];
int m1, m2, m3;
public:
void accept ()
{
cout << "Enter the name of the students";
cin >> name;
cout << "Enter m1:";
cin >> m1;
cout << "Enter m2:";
cin >> m2;
cout << "Enter m3:";
cin >> m3;
}
friend void calculating (student p);
};
void calculate avg (student p)
{
int avg;
avg = (p.m1 + p.m2 + p.m3)/3;
cout << "The average of the student marks is:"
<< avg;
```

```
int main ()
{
student k;
k.accept ();
calculate.avg (k);
}
```

11] Create three classes, Alpha, Beta, 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
{
int a;
public;
void accept ()
{
cout << "Enter a:";
cin >> a;
}
friend void sum (alpha p, beta q, gamma x);
};
class beta
{
int b;
public;
```

```
void accept_2()
{
cout << "Enter b:";
cin >> b;
}
friend void sum(alpha p, beta q, gamma x);
};
class gamma
{
int c;
public:
void accept_3()
{
cout << "enter c:";
cin >> c;
}
friend void sum(alpha p, beta q, gamma x)
};
void sum(alpha p, beta q, gamma x)
{
int sum;
sum = p.a + q.b + x.c;
cout << "The sum is:" << sum << endl;
}
int main()
{
alpha k;
gamma z;
beta q;
k.accept_1();
q.accept_2();
z.accept_3();
sum(k, q, z); }
```

12] Create a class point with private member. x and y. Write a friend function that calculates and returns the distance between two point objects.

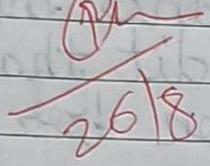
```
#include <iostream>
using namespace std;
class point
{
private:
int x, y;
public:
void accept ()
{
cout << "Enter x and y :";
cin >> x >> y;
}
friend void find difference (Point p1, point p2);
};
void find difference (point p1, point p2)
{
int diff x = p2.x - p1.x;
int diff y = p2.y - p1.y;

cout << "Difference in x :" << diff x << endl;
cout << "Difference in y :" << diff y << endl;
}
int main ()
{
point a, b;
cout << "Enter point :" << endl;
a.accept ();
```

```
cout << "Enter point 2:" << endl;
b.accept();
Find difference(a,b);
return 0;
}
```

8] Create two classes: Bank Account and Audit. Bank Account hold private balance information. Write a friend function in audit that accesses and prints balance information for auditing.

```
#include <iostream>
using namespace std;
class audit;
class bank account
{
int balance;
public:
void accept()
{
cout << "Enter the balance" << endl;
cin >> balance;
}
friend void access balance(bank account.k);
};
void access balance(bank account k)
{
int put balance;
put balance = k.balance;
cout << "the balance for auditing is"
     << put balance;
}
```

```
int main ()
{
bank account x;
x.accept ();
access balance (x);
}
```



Experiment : 5

a] WAP to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

```
#include <iostream>
using namespace std;
class sum
{
private:
int num, summy=0;
Public:
sum ()
{
num = 5
for (int i=0; i < num; i++)
{
summy = summy + num;
}
cout << "Sum of n number is" << summy;
}
sum (sum :and t)
{
num=t, num;
for (int i=0; i< num; i++)
{
summy = summy + num;
}
cout << "Sum of n number is" << summy;
}
};
```

```
sum (int n)
{
num=n
summy=0;
for (int i=1; i<=num; i++)
{
summ += i;
}
cout << "Sum of numbers from 1 to" << num;
<< summy;
```

```
int main ()
{
sum k;
sum z(34);
sum x(3);
}
```

```
int main()
{
sum s(5);
sum s(10);
return 0;
}
```

b] WAP 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

```
#include <iostream>
#include <string>
using namespace std;
class student
{
private:
string name;
float perc;
public:
student ()
{
cout << "name of student:";
cin >> name;
cout << "percentage of student"
cin >> perc;
}
void display ()
{
cout << "name of student is" << name;
cout << "Percentage of student" << perc;
}
```

student (student &s)
{
name = s.name;
perc = s.perc;

cout << "name of student is "; << name;
cout << "percentage of student is "; << perc;
}
};
int main()
{
student k;
k.display();
student s("Advait", 87);
student z(s);
}

student (string n, float p) = an-Uar
{
name = n;
per = p;

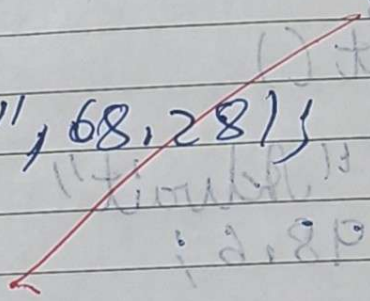
c] Define a class "college" members variable as roll-no, name, course. WAP using constructor with default value as "Computer Engineering" for course. Accept this data for two object of class and display the data.

```
#include<iostream>
#include<string>
using namespace std;
class college
{
int roll-no;
string name;
string course;
public:
college(int r, string s, string c4="CSE")
{
roll-no=r;
name=s;
course=c4;
}
cout<<"name of student "<<name;
cout<<"roll no of student "<<roll_no;
cout<<"course of student:"<<course;
}
};
int main()
{
college k(24,"Advait");
college l(25,"Devang","Civil");
}.
```

d] WAP to demonstrate constructor overloading.

```
#include <iostream>
#include <string>
using namespace std;
class student
{
private:
string name;
float perc;
public:
student()
{
name = "Advait";
perc = 98.6;
}
void display()
{
cout << name;
cout << perc;

student(string n, float p)
{
name = n;
perc = p;
cout << "name of student is " << name;
cout << "percentage of student." << perc;
}
student(student &s)
```

```
student (student &S)
{
name = S.name;
pere = S.pere;
cout << "name is " << name;
cout << "percentage is " << pere;
}
};
int main ()
{
student K;
K.display();
student r("Adrait", 68.28);
student S(r);
}.
```



Experiment : 6

1] Single Inheritance

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

```
int main ()
{
student S1;
S1.accept ();
S1. display ();
}
```

2] Multiple Inheritance

```
#include <iostream>
using namespace std;
class Academic
{
Protected:
int marks;
};
class sports
{
protected :
int S_score;
};
class result : Protected Academic, Sports
{
private :
int total ;
public :
void accept ()
{
cout << Enter academic marks and sports score;
cin >> marks >> S.score;
}
```

```
void calculate ()
{
total = ((marks + S. score) / 200.) * 100.
}
void display ()
{
cout << "Marks" << Marks;
cout << "Sports score" << S_score
cout << "Total score" << total;
}
};
int main ()
{
Result R1;
R1. accept ();
R1. calculate ();
R1. display ();
}
```

3] Multi - level Inheritance

```
#include <iostream>
using namespace std;
class Vechicle
{
protected:
string brand;
string model;
};
class car : Protected vechicle
{
protected:
string type;
};
class EV : protected car
{
private:
int battery;
public:
void accept ()
{
cout << "Enter brand model type and battery
cin >> brand >> model >> type >> battery;
}
void display ()
{
cout << "Brand name" << brand;
cout << "Model name" << model;
cout << "Type :" << type;
cout << "Battery capacity << " battery << "KWH
}
};
```

```
int main()
{
Emp e1;
e1.accept();
e1.display();
}
```

4] Hierarchial Inheritance

```
#include<iostream>
using namespace std;
class Emp
{
protected:
string name;
int emp-id;
};
class Manager:Protected Emp.
{
Private
string dept;
public:
void accept();
{
cout<<"Enter name, emp-id and dept";
cin>>name>>emp-id>>dept;
}
void display()
{
cout<<"Name "<<name;
cout<<"Emp-id"<<emp-id;
cout<<"Dep"<<dept;
}
```

```
class Developer : Protected Emp
{
Private:
string language;
public:
void input ()
{
cout << "Enter Name, emp-id & programming
        language";
cin >> name >> emp-id >> language;
}
void output ()
{
cout << "Name" << name;
cout << "Emp-id" << emp-id;
cout << "Dept" << dept;
}
};
int main ()
{
Manager m;
m.accept ();
m.display ();
Developer d;
d.input;
d.output;
return 0;
}
```

5] Hybrid :

```
#include <iostream>
using namespace std;
class Person
{
Protected:
string name;
int age;
};
class student : protected person.
{
protected:
int roll;
Academics aca;
};
class Academies
{
public:
int marks;
};
class sports.
{
public:
int s, score;
};
class Result : public sports, public student
{
private
int total;
public:
```

```
void accept()
{
cout << "Enter name and age";
cin >> name >> age;
cout << "Enter Roll";
cin >> roll;
cout << "Enter academics and sports marks";
cin >> aca.marks >> S.score;
}
void calculate()
{
total = (aca.marks + S.score)/2;
}
void display()
{
cout << "Name" << name;
cout << "age" << age;
cout << "Roll" << roll
cout << "Aca.marks" << Aca.marks;
cout << "Sports score" << S.score;
cout << "Total" << total;
}
};
int main()
{
Result r;
r.accept();
r.calculate();
r.display();
}
```

6] Virtual base class

```
#include <iostream>
using namespace std;
class clg_student
{
protected;
string clg_code;
}
class Test : virtual public clg_student
{
protected
int perc;
}
class sports : virtual public clg_student
{
protected :
int s_score;
};
class Result : Protected test, sports:
{
public:
void accept()
{
cout << "Enter clg code, per, S_score;
cin >> clg code >> perc >> S_score;
}
```

```
void display ()
{
cout << " cly code =" << cly code;
cout << " percentage " << per;
cout << " sports score " << S_score;
}
};
int main ()
{
Result r
r. accept ();
r. display ();
}
```

Qu

26/9/25

Experiment : 7

a] Write a program using function overloading to calculate the area of a laboratory and area of a classroom

```
#include <iostream>
using namespace std;
class area1
{
public
int l, b;
void area (int a, int b)
{
int c = a * b;
cout << "laboratory Area :" << c << "sq";
}
void area (int s)
{
int b = s * s;
cout << "class Area" << b << "sq";
}
};
int main ()
{
area1 m;
m.area (20, 30);
m.area (20);
}
```

12 WAP using function overloading to calculate the sum of 5 float value and sum of 10 int value

```
#include <iostream>
using namespace std;
class sum.
{
public
int i;
void sum(float a[5])
{
float s=0;
for(i=0;i<5;i++)
{
s+=a[i];
}
cout<<"sum of 10 integer numbers"<<s;
}
};
int main()
{
sum s1;
float c[5];
int d[10];
cout<<"Enter 5 float Nos:";
for(int i=0;i<5;i++)
{
cin>>c[i];
}
cout<<"Enter 10 integer nos:";
{
```

```
s1.sum(c);
s1.sum(d);
}
```

c] WAP to implement Unary operator; when used with the object so that the numeric data members of the class is negated.

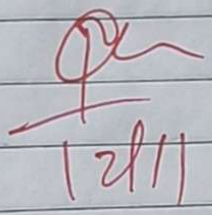
```
#include <iostream>
using namespace std;
class num
{
int a;
public:
void accept()
{
cout << "Enter value of a";
cin >> a;
}
void disp()
{
cout << "Value of a:" << a;
}
void operator-().
{
a = -a;
}
};
```

```
int main()
{
num n1;
n1.accept();
--n1;
n1.disp();
}
```

// WAP to implement the unary ++ operator using with the object so that the numeric data member of the class is incremented.

```
#include <iostream>
using namespace std;
class num
{
int a,b,c;
public:
void accept()
{
cout << "Enter value of a";
cin >> a >> b;
}
void disp()
{
cout << "Value of a:" << a;
}
void operator ++()
{
a = ++a;
}
};
```

```
int main ()
{
num n1;
n1.accept ();
++n1
n1.disp ()
}
```



Experiment: 8

a] WAP to overload the '+' operator so that two strings can be concatenated. Eg "x y z" + "p q r" then output will "x y z p q r"

```
#include <iostream>
using namespace std;
class mstring
{
string str;
public:
mstring (string s)
{
str = s;
}
mstring()
{
str = "";
}
void operator +(mstring obj)
{
str = str + obj.str;
}
void disp ()
{
cout << str;
}
int main ()
{
mstring s1("x y z"), s2("p q r"), s3;
s1 + s2;
cout << "concatenated string :";
```

```
S2=S1;
S2.display();
}
```

[b] WAP to create a base class ILogin having data members name and password. Declare accept() function virtual. Derive Email Login and Membership Login classes from ILogin, display Email Login details and membership login details of the employee

```
#include <iostream>
using namespace std;
class ilogin
{
protected
string name;
string pass;
public:
virtual void accept()
{
cout << "Enter name and password";
cin >> name >> pass;
}
virtual void disp()
{
cout << "Name" << name;
cout << "Password << pass;
}
}
```

```
class elogin : public ilogin
{
string email;
string pass;
public;
void accept ()
{
cout << "Enter Email and Password"
cin >> email >> pass;
}
void disp ()
{
cout << "Email" << email;
cout << "Password" << pass;
}
~~void disp~~
};
class mlogin: public ilogin
{
string mid;
string pass;
public:
void accept()
{
cout << "Enter M-id and Password";
cin >> mid >> pass;
}
void disp ()
{
cout << "M-id" << mid << "Password" << pass;
}
```

```
int main ()
{
ilogin *iptr
ilogin i;
elogin e;
mlogin m;
iptr = &i;
iptr => accept ();
iptr -> display ();
iptr = &m;
iptr -> accept ();
iptr -> disp ();
return 0;
}
```

Q1

12/11

Experiment: 9

1]
```
#include <iostream>
#include <fstream>
#include <string>
int main ()
{
ifstream fin;
ofstream fout;
fout.open ("destination.txt");
fin.open ("source.txt");
if (!fin)
{
cout << "Error opening source file";
return 1;
}
char ch;
while (fin.get (ch))
{
fout.put (ch);
}
fin.close ();
fout.close ();
cout << "File opened successfully";
fin.open ("source.txt");
int wordcount = 0;
string word;
while (fin >> word)
{
wordcount++;
}
cout << ".Wordcount :" << wordcount;
```

```
fin.close();
fin.open("source.txt");
string target;
int count=0;
cout<<"Enter target";
cin>>target;
while(fin>>word)
{
if(word==target)
count++;
}
cout<<"Word occurrence:"<<count;
fin.close();
fin.open("source.txt");
int digitcount=0;
int spacecount=0;
while(fin.get(ch))
{
if(isdigit(ch))
digitcount++;
if(isspace(ch))
spacecount++;
}
cout<<"digit"<<digitcount;
cout<<"space"<<spacecount;
fin.close();
return 0;
}
```

12/11

Experiment : 10

1]
```
#include <iostream>
#include <math>
using namespace std;
template <class T>
class A
{
T m1, m2;
void accept()
{
cout << "Enter the first number";
cin >> m1;
cout << "Enter the second number:";
cin >> m2;
}
void calc()
{
int choice;
cout << "-- Simple Calculator Menu --\n";
cout << "1. addition \n";
cout << "2. Multiplication \n";
cout << "3. division \n";
cout << "4. Substraction \n";
cout << "5. Square root \n";
cout << "6. Percentage (m1 is what % m2)";
cout << "7. Power (square of Both numbers)\n";
cout << "8. Trigonometric (Sin Values) \n";
cout << "Enter your choice";
cin >> choice;
```

```
switch (choice)
{
case 1:
cout << "sum" << m1+m2 << endl;
break;
case 2:
cout << "Multiplication" << m1 * m2 << endl;
Break
case 3
cout << "Division" << (m1/m2) << endl;
Break
case 4
cout << "Subtraction =" << m1 - m2 << endl;
Break
case 5
cout << "Square root of" << m1 << "="
<< Sqrt (m1) << endl;
cout << "Square root of" << m2 << "="
<< Sqrt (m2) << endl;
Break
case 6:
cout << m1 << "is" << (m1 * 100.00)/m2).
<< "% of" << m2 << endl;
Break
case 7
cout << "Square of" << m1 << "=" << pow (m1,2
cout << "Square of" << m2 << "=" << pow (m2,2
Break
case 8
cout << "sin (" << m1 << ") << sin (m1);
cout << "sin (" << m2 << ") << sin (m2);
```

```
Break;
default:
cout << "Invalid choice" << endl;
}
}
int main()
{
A<double> obj;
obj.accept();
obj.calc();
return 0;
}
```

Experiment 11

1]
```
#include <iostream>
#include <queue>
using namespace std;
void prints(queue<string> q)
{
if (q.empty())
{
cout << "Queue is Empty!!!";
}
else
{
cout << "Queue Elements are:";
while (!q.empty())
{
cout << q.front()
 q.pop();
}
cout<<cout << endl;
}
}
int main()
{
int ch;
string name;
queue<string> games;
while (1)
{
cout << "1. Push \n. 2. Pop All Elements \n. 3 Pop
One Elements \n 4. Size \n. 5. Display Queue \n
6. Exit \n";
```

```
cout << "Enter choice : \n";
cin >> ch;
switch (ch)
{
case 1:
cout << "Enter Game Name : " << endl;
cin >> name;
games.push (name);
break;
case 2:
cout << "Popped All Elements : \n \n";
while (!games.empty ())
{
cout << games.front () << endl;
games.pop ();
}
cout << endl;
break;
case 3:
cout << "Popped One Elements :" << games.front ();
games.pop ();
break;
case 4:
cout << "Queue size : " << games.size ();
break;
case 6
exit (0)
  break;
default
cout << "Invaild choice ! \n")
break;
}
}
```

2]
```
#include <iostream>
#include <vector>
#include <cstype>
using namespace std;
int main ()
{
vector <int> vec (5);
int i;
cout << "Enter Vector 5 Elements: ";
for (i=0; i<5; i++)
{
cin >> vec [i];
}
cout << "Vector Elements are: " << endl;
for (i=0; i<5; i++)
{
cout << vec [i] << endl;
}
cout << "Modified Elements are ";
for (i=0; i<5; i++)
{
vec [i] = vec [i] + i * 2;
}
for (i=0; i<5; i++)
{
cout << vec [i] << " ";
}
int scalar;
cout << "Enter a Scalar Value to Multiply: ";
cin >> scalar;
cout << "After Multiplying by scalar";
```

```
for(i=0; i<5; i++)
{
vec[i] = vec[i] * scalar;
}
for(i=0; i<5; i++)
{
cout << vec[i] << " ";
}
return 0;
}
```

Experiment : 12

1] Implement stack

```
#include <iostream>
#include <cstring>
#include <stack>
using namespace std;
template <class T>
class stack
{
stack <T> my_stack;
Public:
void accept ()
{
int size;
cout <<"Enter the size of the stack";
cin >> size;
T data;
for(int i=0; i < size; i++)
{
cout <<"Enter the data for the stack";
cin >> data;
my_stack.push (data);
}
}
void Pop()
{
T.a;
cout <<"Enter the value which you want to
delete \n" << endl;
cin >> x;
```

```
If (my_stack.empty ())
{
cout << "The stack is empty ";
}
If (My Stack empty ())
{
cout << "the stack is empty";
}
If (!My_ Stack empty ())
{
my-stack.pop (x);
}
}
void display ()
{
If (My_ stack . empty ())
{
cout << "The stack is empty";
}
while (! My_ stack . empty ())
{
cout << "My_stack.top ();
my-stack . pop ();
}
}
int main ()
{
stack < int > K;
K.accept ();
K.display ();
}
```

2] Implement queue

```
#include <iostream>
#include <queue>
using namespace std;
template <class T>
class queue
{
Private
Queue <T> q;
Public:
void Enqueue (T. Element)
{
q. Push (Element);
cout << "Element" << "Enqueued Successfully";
}
void dequeue ()
{
if (q. Empty ())
{
cout << "Queue is Empty cannot dequeue";
}
else
{
cout << q. front () << "dequeued Succefully";
q. pop ()
}
}
void display ()
{
cout << "Queue . Elements :";
Queue <T> temp = q;
```

```
while (!temp.Empty ())
{
cout << temp.front () << " ";
temp.pop ();
}
}
int main ()
{
Queue <int > mq;
mq.Enque (30);
mq.Enque (20);
mq.Enque (10);
mq.Display ();
mq.dequeue ();
return 0;
}
```

12/11