1. Add two numbers.

→

```
#include <iostream.h>
using namespace std;
int main () {
int a, b, sum=0;
cout<<"Enter 2 numbers";
cin>>a>>b;
sum = a+b;
cout<<"Sum is \d"<<sum;
return 0;
}
```

Output:
Enter 2 numbers
2
3
Sum is 5.

2. Arithmetic operations using switch.

→

```
#include <iostream.h>
~~int main~~ using namespace std;
int main ()
{
float a,b;
char ope;
cout<<"+,-,/,*";
```

```
cin >> ope;
cout << "Enter 2 numbers";
cin >> a >> b;
switch (o)
{
case '+' cout << "sum is" << a+b;
        break;
case '-' cout << "Difference is" << a-b;
        break;
case '/' cout << "Division is" << a/b;
        break;
case '*' cout << "Product is" << a*b;
        break;
case '/' cout << " if (b != 0)
        cout << "Division is" << a/b;
        else
        cout << "can't divide";
        break;
}
return 0;
}
```

Output:

+, -, *, / : *

Enter 2 numbers: 5

2

Product is 10.

PAGE NO.
DATE / /

3. Check if the number is even or odd.

→

```
#include <iostream.h>
using namespace std;
int main ()
{
int num, even, odd;
cout<<" Enter an integer:";
cin>> num;
if (num % 2 == 0)
{
cout<<* num<< "is even";
else
cout<< num<< "is odd";
}
return 0;
}
```

Output:

Enter an integer: 6

6 is even.

4 Print 1-10 no.'s using for loop.

→

```
#include <iostream.h>
using namespace std;
    int main ()
        {
    int i;
for(i=0; i<10; i++){
 cout << "\n" << i ;
    }
 return 0;
    }
```

Output: 1 2 3 4 5 6 7 8 9 10
in vertical

5 Print 1-10 no.'s using while loop.

→

```
#include <iostream.h>
using namespace std;
 int main ()
     {
int i;
  i=0;
while (i<11) {
cout << "\n" << i;
     i++; }
  return 0;
    }
```

output: 1 2 3 4 5 6 7 8 9 10
in vertical

6. Draw below pattern:

a.
```
    *
  * *
* * *
```

→

```
#include<iostream.h>
using namespace std;
{
int i, j, s;
for (i=1; i<=3; i++) {
for (s=1; s<=3; s++) {
cout<<" ";
}
for (j=1; j<=i; j++) {
cout<<"*";
cout<<" ";
}
cout<<"\n";
}
return 0;
}
```

1418

Output:
```
    *
  * *
* * *
```

b. 1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

→

```
#include <iostream.h>
using namespace std;
int main () {
 int i, j;
for (i= 1; i<=5 ; i++)
  {
 for (j=1 ; j<=i ; j++)
  {
 cout<< j;
  }
 cout<<"\n";
  }
 return 0;
  }
```

Output:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

c.
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

→

```
# include <iostream.h>
using namespace std;
    int main () {
      int i, j;
for (i=1; i<=5; i++)
  {
for (j=1; j<=i; j++)
  {
  cout << i;
    }
  cout << "\n";
    }
  return 0;
    }
```

Output:

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

PAGE NO.
DATE / /

Experiment: 1

1. WAP to declare a class student having data members as roll, name, Accept & data for one object.

→

```
#include <iostream.h>
using namespace std;
class student {
private:
  string name;
   int roll;
  public:
 void accept ()
   {
Cout << " Enter student name \n";
  Cin >> name;
Cout<< " Enter roll no. \n";
  cin >> roll;
   }
   void display ()
   {
Cout << " Student name : " << name;
cout << " \n Roll no. : " << roll;
   }
   };
int main ()
   {
student s1;
```

```
    s1.accept();
    s1.display();
  }
```

Output:

```
Enter Student name:
  Arnav
Enter roll no.:
  42
Student name: Arnav
Roll no.: 42
```

2. WAP to declare a class book having data members as b id, name, b price. Accept data for 2 bks & display name of book having greater price.

→

```
#include<iostream.h>
using namespace std;
class book {
private:
 string name;
  float id;
public:
   float price;
 void accept() {
cout<<"Enter book id \n";
   cin >>id;
cout<<"Enter book name \n";
   cin>>name;
cout<<"Enter price \n";
   cin >> price;
     }
 void disp() {
cout<<"Book id:" <<id;
cout <<"Book name:"<<name;
 cout <<"Book price:" << price;
   }
   };
 int main {
book b1, b2;
```

```
b1. accept ();
b2. accept ();
b1. disp ();
b2. disp ();
if (b1.price > b2.price)
cout << "\n Book 1 price is more";
  else
cout << "\n Book 2 price is more";
  }
```

Output:

```
Enter book id:

Enter book name:

Enter book price:

Enter book id:

Enter book name:

Enter book price:

Book name: abc
Book id: 123
Book price: 986
Book name: xyz
Book id: 456
Book price: 1065
 Book 2 price is more.
```

3. WAP to declare class time. Accept time in HH:MM:SS & display total time in secs.

→

```
#include <iostream.h>
using namespon std;
class Time
  {
public :
 float hr, min, sec;
 float sum;

 void accept () {
cout<< "Enter hours:";
cin >> hr;
cout << "Enter minutes:";
 cin >> min;
cout<< "Enter seconds:";
 cin >> sec;
  }

void disp()
    {
cout <<" Hours :"<< hr<< \n;
cout << " Minutes:"<< min<< \n;
cout << " Seconds:"<< sec<< \n;
  }
 void to second ()
   {
```

```
sum = ((hr * 3600) + (min * 60) + sec);
cout << " Total seconds: " << sum << "\n";
    }
};

int main () {
    Time. t1;
    t1. accept();
    t1. disp ();
    t1. to second;
    return 0;
    }
```

Output:

```
Enter hour : 1
Enter minutes: 1
Enter seconds: 1
  Hour: 1
  Minute: 1
  Second: 1
Total seconds: 3661.
```

14/8

Experiment : 2

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

→

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

class City {
    int population;
    string name;

public:
    void acceptdata () {
    cout<< "City Name:";
    cin >> name;
    cout<< "City Population:";
    cin >> population;
    }

    void displaydata {
    cout<<"\n City Name: "<< name;
    cout<< "\n City Population: "<< population;
    }

    int getPopulation () {
```

```
    return population;
    }

string getName(){
    return name;
    }

int main(){
City c1,c2,c3,c4,c5;

c1.acceptdata();
c2.acceptdata();
c3.acceptdata();
c4.acceptdata();
c5.acceptdata();

c1.displaydata();
c2.displaydata();
c3.displaydata();
c4.displaydata();
c5.displaydata();

City cities[5]={c1,c2,c3,c4,c5};
City maxCity = cities[0];

for(int i=1; i<5; i++){
  if(cities[i].getpopulation()>maxCity.getpopulation())
    {
      maxCity = cities[i];
    }
```

```
}

cout<<"\n City with highest population:"<<
      maxCity.getName()<<" with "<<
      maxCity.getPopulation()<<" people."<<
      endl;
return 0;
}
```

Output:

```
City Name: Pune
City Population: 1234
City Name: Mumbai
City Population: 2345
City Name: Sangli
City Population: 456
City Name: Goa
City Population: 998
City Name: Dubai
City Population: 9999

City Name: Pune
City Population: 1234

City Name: Mumbai
City Population: 2345

City Name: Sangli
City Population: 456
```

City Name: Goa
City Population: 998

City Name: Dubai
City Population: 99999

City with highest population: Dubai with 9999 people.

2. WAP to declare a class 'Account' having data members as Account no. & balance. Accept this data for 10 accounts & give interest of 10% where balance is equal to or greater than 5000 & display them

→

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

class Acc {
    int acc;
float bal, bon=0;

public:
 void acceptdata () {
Cout<< "Enter Account Number:";
   cin>> acc;
Cout<< "Enter Balance ";
   cin>> bal;
   }

void CalculateBonus (){
if (bal>=5000){
 bon = bal*10/100;
   }
else
   {
    bon = 0;
```

```
    }
}

void displaydata() {
cout<<"\n Account No.: "<<acc;
cout<<"\n Balance: "<<bal;
cout<<"\n Bonus: "<<bon;
cout<<"\n Total (Balance + Bonus): "<<bal+bon
        << endl;
    }
};

int main() {
acc a[10];
 int i;
for (i=0; i<10; i++) {
 a[i].acceptdata();
      }
for (i=0; i<10; i++) {
  a[i].calculateBonus();
    }
for (i=0; i<10; i++) {
  a[i].displaydata();
    }
  return 0;
    }
```

- Output:

Enter ~~No.~~ Acc No.: 1
Enter Balance: 1000
Enter Acc No.: 2
Enter Balance: 2000
Enter Acc No.: 3
Enter Balance: 3000
Enter Acc No.: 4
Enter Balance: 4000
Enter Acc No.: 5
Enter Balance: 5000
Enter Acc No.: 6
Enter Balalance: 1500
Enter Acc No.: 7
Enter Balance: 2500
Enter Acc No.: 8
Enter Balance: 3500
Enter Acc No.: 9
Enter Balance: 4500
Enter Acc No.: 10
Enter Balance: 6500

Account No.: 5
Balance: 5000
Bonus: 500
Total (Balance + Bonus) = 5500.

Account No.: 10
Balance: 6500
Bonus: 650
Total (Balance + Bonus) = ~~6500~~. 7150.

3 WAP to declare a ~~ea~~ class 'Staff' having data members as name & post. Accept this data for 5 staff & display them.

→

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

class Staff{
string name, post;

public:
void accept(){
{
cout<<"Enter Staff Name:";
cin>> name;
cout<<"Enter Staff Post:";
cin >> post;
}

void display()
{
cout<<"In Staff Name:"<<name<<"In Staff
        Post:"<<post<<endl;
if (post=="HOD"||post=="hod")
{
cout<<"name"<< is Head of Department\n"
}
else {
```

```
cout<<"name"<< is a staff.\n";
        }
    };

int main () {
  staff s[2];
    int i;
for (i=0; i<2; i++)
  {
  s[i]. accept ();
      }
 for (i=0; i<2; i++)
     {
   s[i]. display ();
      }
   return 0;
  }.
```

- Output:

```
Enter staff name: a
Enter post: sales
Enter staff name: b
Enter post: manager
Enter staff name: c
Enter post: hod
Enter staff name: d
Enter post: tally
```

Enter staff name: e
Enter post: random

Staff name: c
Staff post : hod.
c is the hod of department

1/1/8

PAGE No.
DATE / /

Practical 3

① Pointer to that object:

```
#include <iostream>
using namespace std;
class book {
string title;
string a_name;
int price;
public:
void info () {
cout<<"Enter the book title, author name
        and price of that book :";
cin>> title>> a_name>> price;
  }
void display () {
cout<<"Book Name:"<< title<< endl;
cout<<"Author Name:"<< a_name<< endl;
cout<<"Price:"<<price;
  }
};
int main () {
book * p;
book b1;
p=&b1;
p= info ();
p= display ();
}
```

PAGE No.
DATE / /

Output:
Enter the book title, author name & price of that book: test
ABC
100.

Book Name: test
Author name: ABC
Price: 100.

② This pointer.

```
#include <iostream.h>
using namespace std;
class student {
 int roll;
 float perc;
 public:
 void info (int rolln, float percen) {
  this -> roll = rolln ;
  this -> perc = percen ;
   }
  void display () {
 cout<<" Roll no. :"<<roll<<" \t Marks : "<<perc
  }
  };
int main () {
 student s1;
 s1. info (3, 89.7);
  s1. display;
   }
```

Output:
Roll no. : 3
Marks : 89.7

PAGE No.
DATE / /

③ Nested class

```
#include <iostream>
using namespace std;
class marks {
public:
class percentage {
 int m,n;
 float p,i;
 public:
 void info (){
cout<<"Enter the marks you got & total marks:";
 cin>> m>>n;
    }
void disp (){;
i= (float) m/n;
 p= i*100;
cout << "Percentage" <<p;
  }; };
 int main (){
 marks m1;
marks: percentage p1;
 p1. info ();
 p1. disp ();
  }
```

14/8.

Output: Enter the marks you got & total marks:
56
60
Percentage : 93.33

PAGE NO.
DATE

Experiment - 4.

① ② ③ Passing of object as a function (swap)

```
#include <iostream>
using namespace std;
class number {
int value;
public:
number (int v=0) {
  value = v;
    }
void swap (number & other) {
int temp = value;
value = other.value;
other.value = temp;
    }
Void disp () {
cout << " Value: " << value << endl;
  } };
int main (){
number n1(10), n2 (20);
Cout << " Before swap:" << endl;
n1.disp ();
n2: disp ();
n1. swap ();
Cout << "\n After swap:" << endl;
  n1.disp();
  n2.disp();
   return 0;
    }
```

Output:

~~Enter 2 numbers: 5~~
4

~~Values after swapping: 4~~
5.

Before swap
Value: 10
Value: 20
After swap
Value: 20
Value: 10

② Friend function (swap some class)

```
#include <iostream.h>
using namespace std;
class AB {
int a, b;
public:
void info () {
cout<<" Enter 2 numbers :");
cin >> a >> b;
}
friend void swap (AB a1);
};
void swap (AB a1) {
int temp;
temp = a1.a;
a1.a = a1.b;
a1.b = temp;
cout<<" In values after swapping :"<< a1.a<<
a1.b ; }
int main () {
AB .a1;
a1.info;
swap (a1) ;
}
```

o/p: Enter 2 numbers : 5 4
Values after swapping: 4 5

PAGE No.
DATE / /

③ (B) (E) Friend Function

```
#include <iostream>
Using namespace Std;
class CB;
class CA {
int numA;
 public:
 CA(int val) : numA (val) {}
 void disp (){
 Cout << "Value in Class A:" <<numA <<endl
 }
 friend voidswap (CA &, CB&);
};
class CB {
 private:
 int numB;
 CB (intval) : numB (val) {}
 Voiddisp (){
 Cout<<" Value in classB:" <<numB<<endl;
}};
void friend voidswap (CA&, CB&);
};
void swap (CA&a, CB&b)
{ int temp = a.numA;
  a.numA = b.numB;
   b.numB = temp;
}
int main(){
CA objA (10);
CB objB (20);
```

```
cout<<" Before swapping:"<<endl;
objA.disp();
ObjB.disp();
swap(objA, objB);
cout<<" After swapping:"<<endl;
objA.disp();
objB.disp();
return 0;
}
```

Output : Before swapping :
Value in class A : 10
Value in class B : 20
After swapping :
Value in class A : 20
Value in class B : 10

(4)

Avg of 2 results.

```
#include <iostream>
using namespace std;
class result2;
class result {
int a;
public:
void accept() {
cout<<"Enter marks out of 50";
cin>>a;
}
friend void (a) result r1, result2 r2);
};
class result2; {
int b;
public:
void accept () {
{
cout<<"Enter marks out of 50";
cin>>b;
}
friend void cal (result r1, result r2);
};
void cal (result r1, result r2)
{
float avg = (float) (r1.a+r2.b)/2;
```

```
cout<< " Average" <<avg;
 }
int main ()
  {
result x;
result2 y;
x.accept ();
y.accept ();
cal (x,y);
}
```

Output: Enter marks got out of 50: 45
Enter marks got out of 50: 46
Average: 45.6

5 Greatest among 2 no.'s (diff class) (friend function)

```
#include<iostream>
using namespace std;
class B;
class A
 {
 int a;
 public:
 void acc() {
cout<<" Enter a value";
   cin >> b;
   }
friend void gr (A a1, B a1);
   };
 void gr (A a1, B a1)
   {
   if (a1.a > b1.b)
   {
cout<<" first value is greater";
   }
  else
 {
cout<<" second value is greater;
   }
 int main() {
   A x;
   B y;
```

```
x. acc();
y. acc();
gr(x,y);
}
```

Output: Enter value : 10
Enter value : 100
Second value is greater.

6. Create three classes: Alpha, Beta, Gamma each with a privata data member. Write a single friend function that can access all three ~~end points~~ & print their sum.

→
```
#include <iostream>
using namespace std;
class Beta; class Gamma;

class Alpha {
  int a;
  public:
Alpha(int val) : a(va) { }
friend void printsum (Alpha, Beta, Gamma);
  };

class Beta {
  int b;
  public:
Beta (int val) : b(val) { }
friend void printsum (Alpha, Beta, Gamma);
  };

class Gamma {
  int c;
  public:
Gamma (int val) : c(val) { }
friend void printsum (Alpha, Beta, Gamma)
  };
```

```
void printsum (Alpha x, Beta y, Gamma z)
  {
cout<<"Sum:" << x.a + y.b + z.c <<endl;
    }

int main () {
Alpha a(10); Beta b(20); Gamma c(30);
 printsum (a, b, c) ;
    return 0;
     }
```

Output:

Sum = 60

Practice questions on friend function.

1. Create 2 classes, class A & B, each with a private integer. Write a friend func^c sum() that can access pvt data from both classes & return the same.

→

```
#include<iostream>
using namespace std;
  class Class B;

class Class A {
   int a;
public:
Class A(int val) : a(val) {}
 friend int sum (Class A, Class B);
   };

class Class B {
   int b;
public:
class B (int val): b(val) {}
friend int sum (Class A, Class B);
   };

int sum (Class A obj A, Class B obj B) {
 return obj A.a + obj B.b;
   }
```

```
int main () {
Class A a(10);
Class B b(20);
cout<<"sum: "<< sum(a, b)<<endl;
return 0;
```

o/p:

sum: 30

2 WAP to with a cls no. that contains a pvt. integer. Use a friend func swap no.'s (Number &, Number &) to swap the pvt values of 2 no. projects.

→

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

class Number {
 int value;
public:
Number (int val): value (val) { }
void display () {
cout<< value<< endl;
friend void swapNumbers (Number &, Number &);
    };

void SwapNumbers (Number &n1, Number &n2) {
 int temp = n1.value;
  n1. value = n2.value;
   n 2.value = int temp;
   }

int main () {
Number n1 (5), n2 (15);
cout<<"Before Swap:";
  n1. display ();
  n 2. display ();
```

```
swap Numbers(n1, n2);
Cout<<" After swap:";
 n1. display ();
 n2. display ();
 return 0;
   }
```

o/p : Before swap: 5
15

After swap: 15
5.

3. Define 2 cls's Box & Cube, each having a pvt. volume. Write a friend func. Find Greater (box, cube) that determines which obj has a larger volume.

→

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

class Cube;

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

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

int void findGreater(Box b, Cube c) {
cout<<"Greater volume:"<<(b.volume > c.
  volume ? b.volume : c.volume)endl;
    }
```

```
int main () {
 Box box (118) ;
Cube  cube (90) ;
find Greater (box, cube);
     return 0;
    }
```

o/p: Greater volume: 118

4. Create a class Complex with real & imaginary parts as pvt members. Use a friend func to add 2 no.'s & return the result as a new complex obj.

→

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

class Complex {
  int real, img;
  public:
complex(int r=0, int i=0): real(r), img(i){}

void display() {
cout<< "real" <<"+"<< img <<"i"<<endl;
 friend complex add(complex, complex);
    };

Complex add(Complex c1, Complex c2) {
return Complex(c1.real+c2.real, c.img+
  c2.img);
    }

 int main() {
complex c1(4,5), c2(2,3);
complex result = add(c1,c2);
cout<<"Sum of complex numbers:";
  result.display();
```

```
    return 0;
}
```

Output:

```
Sum of complex numbers : 6+8i
```

5 Create a cls Student with pvt data members of 3 sub. marks. Write a friend func. Calculate Avg (Student) that calculate & display the average marks.

⇒

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

class Student {
 string name;
 int m1, m2, m3;
  public:
Student(string n, int a, int b, int c), name(n),
    m1(a), m2(b), m3(c) {}
friend void calculate Avg (Student);
   };

void calculate Avg (Student s) {
float avg = (s.m1+s.m2+s.m3)/3;
cout<<"Average marks of "<<s.name<<":"
  << avg<<endl;
    }

int main () {
student s ("Arnav", 90, 93, 91);
 calculate Average (s);
  return 0;
    }
```

o/p: Average marks of Arnav : 91

7. Create a clg Point with pvt mmbrs x & y. Write a friend func that calculates & returns the distance betn 2 pt. objects.

→

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

class Point {
int x, y;
public:
point(int a, int b): x(a), y(b) {}
friend double distance (Point, Point);
};

double distance (Point p1, Point p2) {
return sqrt (pow (p2.x - p1.x, 2) + pow (p2.y -
p1.y, 2))
};

int main () {
Point p1(0,0), p2(3,4);
cout<< "Distance:" << distance (p1, p2) << endl;
return 0;
};
```

o/p: Distance: 5

8. Create 2 cls's. Bank Acc & Audit Bank Acc holds pvt. balance info^n. write a friend func in Audit that access & prints balance info^n for auditing.

→

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

class BankAccount {
  double balance;
  public:
BankAccount(double b): balance(b) {}
  friend class Audit;
    };

class Audit {
  public:
void printBalance (BankAccount acc) {
cout<<"Audited Balance:" << acc.balance<<
    endl;
  }
};

int main () {
BankAccount acc (5000.75);
  Audit audit;
audit.printbalance (acc);
  return 0;
    }
```

12/11

o/p: Audited Balance: 5000.75.

Practical-5

★ Write a CPP program to implement type of constructor.

① WAP to find the sum of no.'s betn 1 to 'n' using a constructor where the val. of 'n' will be passed to the constructor.

→ Default constructor:

```
#include <iostream>
using namespace std;
class Sum {
  int n;
  public:
  Sum () {
   n = 10;
    }
  void calc () {
  int s=0, i;
  for(i=0; i<=n; i++) {
  s=s+i;
    }
  cout<< "Sum is :" <<s;
    } };
  int main() {
  Sum s1;
  s1. calc();
    }
```

→ o/p: sum is : 55

→ Parameterized constructor.

```
#include<iostream>
using namespace std;
class sum {
  int no;
 public:
 sum(int n){
 no=n;
   }
 void calc(){
 int s=0; i;
 for(i=0; i<=no; i++){
 s=s+i;
   }
 cout<<"sum is:"<<s;
   }};
 int main(){
 sum s1(10);
 s1.calc();
   }
```

→ o/p: sum is 55.

→ Copy Constructor

```
#include<iostream>
using namespace std;
class sum{
 int no;
public:
 Sum(int n){
  no=n;
    }
 Sum(Sum &su){
  no= su.no;
    }
 void calc(){
 int s=0; i;
for(i=0;i<=no;i++){
s=s+i;
 }
cout<<"sum is:"<<s;
  }};
int main(){
Sums s1(10);
Sums s2(s1);
s1.calc();
s2.calc();
   }
```

→ o/p: sum is 50
sum is 50.

PAGE NO.
DATE / /

② WAP to declare a cls 'student' having data members as name & %. WAC to initialize these data members. Accept & display data for one student.

→ Default constructor

```
#include <iostream>
using namespace std;
class Student{
 string name;
int m1, m2, total;
public:
student(){
name = "ABC";
m1 = 72;
m2 = 90;
total = 200;
 }
void calc(){
float perc = (float) (m1+m2) / total * 100;
cout << "Student name :" << name << endl;
cout << " Percentage :" << perc << endl;
 };
int main(){
student s1;
 s1. calc();
  }
```

→ o/p: Student name: ABC
Percentage: 81

→ Parameterized constructor.

```
#include <iostream>
using namespace std;
class student{
string name;
int m1, m2, total;
public:
student(string n, int mv1, int mv2, int t){
name=n;
m1=mv1;
m2=mv2;
total=t;
  }
void calc(){
float perc=(float)(m1+m2)/total*100;
cout<<"Student name:"<<name<<endl;
cout<<"Percentage:"<<perc<<endl;
  }};
int main(){
student s1("ABC",90,90,200);
s1.calc();
  }
```

→ o/p: Student name: ABC
Percentage: 90

→ Copy Constructor

```
#include <iostream>
using namespace std;
class student {
string name;
int m1, m2, total;
public:
student(string n, int mv1, int mv2, int t) {
 name = n;
 m1 = mv1;
 m2 = mv2;
 total = t;
student (student &st) {
 name = st.name;
 m1 = st.m1;
 m2 = st.m2;
 total = st.total;
  }
void calc() {
float perc = (float)(m1+m2)/total * 100;
cout<<"Student name:" <<name<<endl;
cout<<" Percentage:" <<perc<<endl;
  }};
int main() {
student s1("ABC", 90, 90, 200);
student s2(s1);
s1.calc;
s2.calc;
  }
```

2/11

o/p: Student name: ABC
Percentage: 90
Student name: ABC
Percentage: 90.

Practical - 6

① Single Inheritance

Problem: Create a base cls called person with attributes name & age. Derive a cls student from person that adds an attribute roll no. Write funct. to disp. all details of the student.

```
#include <iostream>
using namespace std;
class person {
protected:
string name;
 int age;
 };
class student: protected person {
 int roll_no;
public:
void setdata(string n, int a, int r){
name = n;
 age = a;
roll_no = r;
 }
 void display(){
cout<<"Name" <<name<<endl;
cout<<" Age" << age << endl;
cout<<"Roll No." <<roll_no<< endl;
 }};
int main() {
```

```
student S;
S. setdata ("A", 20, 101);
S. display ();
}
```

o/p: Name: A
Age: 20
Roll No.: 101

② Multiple Inheritance

Problem: Create two base cls's Academic & Sports.
- Academic cls contains marks of a student.
- Sports cls contains sports score.

```
#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 acc() {
```

```
cout<< "Enter name:";
cin>> name;
cout<< "Enter marks & score:";
cin>> marks>> score;
total = marks + score;
  }
void disp(){
cout<< "Name:" << name << endl;
cout<< "Marks:" << marks << endl;
cout<< "Score:" << score << endl;
cout<< "Total:" << total;
};
int main(){
student s1;
s.acc();
s.disp();
  }
```

⇒ o/p: Enter name: ABC
Enter marks & score: 95 5
Name: ABC
Marks: 95
Score: 5
Total: 100.

③ Multilevel

```
#include<iostream>
using namespace std;
class vehicle{
public:
string model, string brand;
  };
class car: public vehicle{
protected:
int type;
};
class ev: public error{
```

```
public:
int bc;
void acc() {
cout<<" Model:"<< model;
cout<<"Brand:"<< brand;
cout<< Enter type(1.Sedan 2.Suv):"<<type;
cout<<" Battery Capacity:(KWH) ";<< bc;
 }
void disp() {
Cout<< "Model:"<< model<<endl;
Cout<< "Brand:"<< brand<<endl;
Cout<< "Type:"<< (type ==1 ? "Sedan":"Suv") <<endl;
Cout<< "Battery Capacity:" << bc << "KWH";
 };
int main () {
ev c;
c. acc();
c. disp();
 }
```

⇒ o/p: Model : Nexon
Brand : Tata
Enter type (1.Sedan 2.Suv) : 2
Battery Capacity: 20000

Model: Nexon
Brand: Tata
Type: Suv
Battery Capacity: 20000 Kwh

PAGE No.
DATE / /

④ Hierarchical

```
#include <iostream>
class emp{
protected:
string name;
int id;
};
class man: public emp{ private:
string dep;
};
class der: public emp{ public:
string lang;
string dep;
void acc(){
cout<<"Emp name:";
cin>>name;
cout<<"Emp ID:";
cin>>id;
cout<<"Enter Programming Lang.:";
cin>>Lang;
cout<<"Enter dept:";
cin>>dep;
}
void disp(){
cout<<"Name:"<<name<<endl<<"ID"<<id<<endl;
cout<<"Programming Lang:<<lang<<endl<<"Dept:"
<<dept;
}};
int main(){
der d;
```

```
d. acc();
d. disp();
}
```

→ o/p: Emp Name: ABC
Emp ID: 101
Enter Programming Lang: CPP
Enter dept: HOD
Name: ABC
ID: 101
Programming Lang: CPP
Dept: HOD

⑤ Hybrid

```
#include <iostream>
using namespace std;
class Teacher {
protected:
string name;
int age;
public:
void acc() {
cout<<"Enter teacher name:";
cin>> name;
cout<<"Teacher age:";
cin>> age;
}
void disp() {
cout<<"Teacher name:" <<name<<endl;
cout<<"Age:" <<age<<endl;
```

```
};
class student: public teacher {
protected:
int roll;
string name;
void accs() {
cout<<"Enter student name:";
cin>>sname;
cout<<"Student roll no.:";
cin>>roll;
}
void disps() {
cout<<"Student name:"<<sname<<endl;
cout<<"Roll no.:"<<roll<<endl;
};
class marks: public student {
protected:
int m1, m2, m3;
public:
void accm() {
cout<<"Enter marks of 3 subj:";
cin>>m1>>m2>>m3;
}
void dispm() {
cout<<"Marks in subj 1,2,3:"<<m1<<endl<<
m2<<endl<<m3<<end;
};
class acad: public marks {
public:
int total() {
```

```
return m1+m2+m3;
}};
int main () {
acad a;
a.acc();
a.accs();
a.accm();
a.disp();
a.dispS();
a.dispm();
int total = a.total();
cout<<" Total Marks :"<< total<<endl;
  }
```

⇒ o/p: Enter teacher name: ABC
Teacher age: 40
Enter Student name: XYZ
Student roll no.: 36
Enter marks of 3 subj: 90
90
90

Teacher name: ABC
Age: 40
Student name: XYZ
Roll no.: 36
Marks in Subj 1, 2, 3: 90
90
90

Total: 270

12/11

Practical -7

① Area of laboratory (rectangle) & area of classroom (square). (Constructor overloading)

```
#include <iostream>
using namespace std;
class xyz {
 public:
int l, b, s, a;
void calc (int len, int br) {
 l = len;
 b = br;
 a = l * b;
 }
void calc (int si) {
 s = si;
 a = s * s;
 }};
int main () {
xyz a;
cout<< "Enter length & breadth of rectangle:";
cin>> a.l >> a.b;
a.calc (a.l, a.b);
cout<< "Area of rectangle:" << a.a << endl;
xyz b;
cout<< "Enter side of square:";
 cin>> b.s;
b.calc (b.s);
```

```
cout<<" Area of square:" << b.a<<endl;
return 0;
 }
```

→ o/p: Enter length & breadth of rectangle: 10
5

Area of rectangle: 50
Enter side of square: 5
Area of square: 25

② Constructor overloading (Sum of 10 integer & 5 float)

```
#include<iostream>
using namespace std;
class Sum {
public:
float fsum=0;
int isum= 0;
void calc(float a) {
fsum = fsum+a;
  }
void calc (int a) {
 isum = isum + a;
  } };
int main() {
Sum s;
float f;
 int i;
cout<<" Enter 5 float values:";
for (int j=0; j<5; j++);
```

```
    cin >> f;
    S.calc();
    }
cout<<"Sum of float values:"<<S.fsum<<endl;
cout<<"Enter 10 integer values:");
for(int j=0; j<10; j++){
    cin>> i;
S.calc(i); }
cout<<"sum of integer values:"<<S.isum<<endl;
    return 0;
    }
```

③ Unary(-) operator

```
#include<iostream>
using namespace std;
class numb{
public:
int val;
void acc(){
cout<<"Enter a numb:";
cin>> val;
    }
void disp(){
cout<<"value:" val;
    }
void operator-(){{
val= -val;
    }};
int main(){
numb obj; obj.acc();
-obj; obj.disp;
    }
```

→ o/p: Enter a numb: 5
Value = -5

PAGE No.
DATE / /

⑤ Unary ++ operator (Pre & post increment)

```
#include <iostream>
using namespace std;
class ab {
public:
int num;
cout<<"Enter number:";
cin>>num;
 }
void disp() {
cout<<"Number:" << endl;
   }
Void operator ++ () {
++num;
}
void operator ++ (int) {
num ++;
}};
int main() {
ab obj;
obj. acc();
++ obj;
obj. disp();
return 0;
  }
```

⇒ o/p: Enter number: 5
Number: 7

12/11

Practical- 8

① Overload '+' operator so that two string can be concatenated.

Code:

```
#include <iostream>
#include <string>
using namespace std;
class abc {
public:
string str;
void acc() {
cout<<"Enter string:";
cin>> str;
 }
abc operator + (abc s) {
abc temp;
temp.str = str + s.str;
return temp;
 }
void disp() {
cout<<"Concatenated string:"<<str<<endl;
};
int main() {
abc s1, s2, r;
s1. acc();
s2. acc();
r = s1 + s2;
r. disp();
return 0;
 }
```

→ o/p: Enter string: Hello
Enter string: World
Concatenated string:
Hello World.

⑤ WAP to create a base cls ILogin having data members name & password. Declare accept funct. virtual. Derive email login & membership - logic classes from Ilogin. Display email login & membership login details of the empl.

Code:

```
#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 disp() {
cout<<"Name"<<name<<" Password"<<pass<<endl;
  }};
class Emaillogin : public Ilogin {
public:
void disp() override {
cout<<"Email login→"<<name<<name<<"password"
 <<password }};

Class Membership login: public Ilogin {
 public:
void disp() override {
```

```
Cout << " Membership login => " << " name " << "
  endl; }};
int main () {
  login * p;
Email login e;
Membership login m;
  p -> e;
  p -> accept ();
  p -> disp ();
  p -> m;
  p -> accept ();
  p -> disp ();
  return 0;
    }
```

Qu
12/11

PAGE NO.
DATE / /

Experiment 9

a. WAP to copy content of one file another open "First.txt" in read (ios.) mode & "second.txt" file in write (ios) mode, Copy the contents.

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

int main() {
ifstream fin ("First.tx");
 ostream fout("second.txt");
  char ch;

 while (fin.get(ch)) {
    Fout.put(ch);
      }
cout<<"File copied successfully!";
 fin.cloo();
 fout.clou();
 return 0;
  }
```

b Count digit & space 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;
   }
```

c. 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. 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)
    {count++;}}

cout<<"Occurence of"<<target<<":"<<count
cout<<endl;
fin.close();
return 0;

}
```

12/11

Experiment -10.

a Sum of array elements

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

template <typename>
T SumArray (Tarr[], int size) {
        T sum = 0;
for (int i= 0; i<size ; i++) {
    sum+ arr [i] ;
      }
   return sum;
      }

int main () {
const int SIZE=10;
int Array [SIZE] = {1,2,3,4,5,6,7,8,9,10};
int intsum = sumArray (int Array, SIZE);

Cout<<"Sum of integer array :"<< int Sum<<
  endl;

return 0;
  }
```

b. Square Function (Template specialization).

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

template <typename T>
T square (T value) {
  return value * value;
    }
template <>
string square<string>(string str) {
  return str+str;
    }

int main() {
int intNum = 5;
cout<< "Square of integer"<<intNum<< ":" <<
  square(um)
 String mgstr = "Hello";
cout<< "Square of string 1"<<" mystr"<<":"<<
 Square (Byshn)
  return 0;
    }
```

C. Simple Calculator (uses template)

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

template <typename T>
class calculator {
  private:
    T num1;
    T num2;
public:

Calculator (T n1, T n2): num1(n1) num2 (n2);
  T add () const {
    return num1 - 1num2;
        }
T subtract () const {
return num1 - num2;
    }
T multiply () const {
return num1 * num2;
    }
T divide () const {
return num1 / num2; }
  return 0;
    }
```

d Stack implementation (class template)

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

template <typename T>
class stack {
 private:
vector <T> elements ;

 public:
 void push (T val) {
element. push-back (val) ;
cout<< " Pushed : " <<val <<endl;
   }
T pop () {
if (elements empty ()) {
cout<<"Error : stack is empty "<<endl;
  return T();
T topelement = element . bach ();
   elements . pop-back<() ;
     return top Element;
        }.
  bout is Empty () const {
     return element. empty ()
        } };
     return 0;
        }
```

12/11

Q2. WAP to count digits & spaces using file handling.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
fstream file("file.txt");
    if(!file) {
cout<<"unable to open file";
    return;
    }
char ch;
int digit_count=0, space_count=0;
  while(file.get(ch)) {
   if (ch>='0' && ch<='9') {
        digit_count++;
        }
   else if (ch==' ')
      space_count++;
      } };
cout<<"digits:" << digit_count<<endl;
cout<<" spaces:" << space_count <<endl;
   file.close()
   return 0;
   }
```

Experiment - 11

Q1. Write a C++ program to implement generic vector.

A. To modify the value of a given element.

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

Exp - 11
continued...

```
int main () {
  vector <int> v(5);
  for (int = 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

b. To multiply by a scalar value

```
# include < iostream>
# include < vector>
  using namespace std;
    int main () {
  vector<int> ~~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). (2/11

Experiment -12

① Write 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;
        }
```

⇒ o/p: Top element: 30
Top element after top: 20
Stack size: 20
Stack is not empty.

b. Implement Queue

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
queue <int> q ;
  q. push (10) ;
   q. pop (20);
   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, count_
    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 = 0; i< (int) people.size(), i++) {
if (people [i].age = search.age ?!=0) {
        found index = i;
        break;
      }
    }
if (found .!= index)
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