① Write C++ program to perform arithmetic operations using switch case.

```
#include <iostream>
using namespace std;
int main() {
  char op;
  double num1, num2;
  cout << "Enter operator (+,-,*,/): ";
  cin >> op;
  cout << "Enter two numbers: ";
  cin >> num1 >> num2;
  switch(op) {
    case '+':
      cout << num1 << "+" << num2 << "=" << num1+num2 << endl;
      break;
    case '-':
      cout << num1 << "-" << num2 << "=" << num1-num2 << endl;
      break;
    case '*':
      cout << num1 << "*" << num2 << "=" << num1*num2 << endl;
      break
    case '/':
      if (num2 != 0)
        cout << num1 << "/" << num2 << "=" << num1/num2 << endl;
      else
        cout << "Error: Division by zero!" << endl;
      break;
    default;
      cout << "Error! Operator is not correct." << endl;
  }
  return 0;
}
```

Output:
Enter Operator (+, -, *, /): /
Enter two numbers: 68
25
68/25 = 2.72

---

(2)

```
#include <iostream>
using namespace std;
int main (){
  int num;
  Cout << "Enter a number:";
  cin >> num;
if(num %2==0) {
  C out << num << "is even" << end1;
} else {
  Cout << num << " is odd." << end1;
}
return 0;
}
```

Output:
Enter a number: 96
96 is even

---

③
```
# include <iostream>
using namespace std;
int main () {
  for (int i=1; i<=10; i++) {
  Cout << i << " ";
}
Cout << endl;
return 0;
}
```

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

---

④
```
# include <iostream>
using namespace std;
int main () {
  int i=10;
  while (i>=1) {
  Cout << i << " ";
  i--;
}
Cout << endl;
return 0;
}
```

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

---

(5) i)

```
#include <iostream>
using namespace std;
int main() {
 int rows = 3;
 for (int i=1; i<=rows; i++){
  for (int j=1; j<=i; j++) {
  cout << "*";
 }
 cout << endl;
}
 return 0;
}
```

Output :-

```
*
* *
* * *
```

---

ii)

```
#include <iostream>
using namespace std;
int main() {
 int rows= 5;
 for (int i=1; i<=rows; i++){
  for (int j=1; j<=i; j++) {
  cout << i << " ";
 }
 cout << endl;
}
 return 0;
}
```

A.

Output :-

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

iii)
```
#include <iostream>
using namespace std;
int main() {
int rows = 5;
for (int i=1; i<=rows; i++) {
for (int j=1; j<=i; j++) {
cout << j << " ";
}
cout << endl;
}
return 0;
}
```

Output :-
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

8/7/25

Page No.:
Date:
YOUVA

5/07/25

Experiment 1

1) WAP to declare a class student having data members as name, Roll-no. Accept & display data for one student.

```
#include <iostream>
#include <string>
using namespace std;
class student
{
 int roll_no;
 string name;
 public:
 void accept()
 {
  cout << "Enter Student Name & Roll No \n";
  cin >> name >> roll_no;
 }
 void disp(){
  cout << " Name of Student :- " << name;
   cout << "\n Roll No=" << roll_no;
 }
};
 int main()
{
  student s1;
  s1.accept()
  s1.disp();
   return 0;
}
```

Output :- Enter Student Name & Roll No :-
Atharva
67
Name of Student :- Atharva
Roll No= 67

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

```
#include <iostream>
#include <string>
using namespace std;
class book {
    int b_pages;
    string b_name;
    float b_price;
public
void accept () {
    cout << "Enter book Name:";
    cin >> b_name;
    cout << "Enter book Price:";
    cin >> b_price;
    cout << "Enter book Pages:";
    cin >> b_pages;
}
void disp () {
    cout << "In Book Name:" << b_name << endl;
    cout << "Price:" << b_price << endl;
    cout << "Pages:" << b_pages << endl;
}
float getPrice () {
    return b_price;
}
};
int main () {
    book b1, b2;
    cout << "Enter details of Book1: \n";
    b1.accept();
```

```
cout<<"\n Enter details of Book 2 :\n";
b2.accept();
cout<<"\n Book with higher Price:\n";
if(b1.getPrice() > b2.getPrice())
    b1.disp();
else
    b2.disp();
return 0;
}
```

*) ② Write a C++ program to accept time in hours, mins & sec. for one number & display total time in sec.

→
```
# include <iostream>
  using namespace std;
class time:
{
  public:
int hr, min, sec;
  void accept()
{ cout << "Enter time (HH:MM:SS)
  cin >> hr >> min >> sec;
}
int convert()
{
int secs = (hr * 3600) + (min * 60) + secs;
  return secs;
}
};
int main()
{
  time t;
t.accept()
Cout << endl << "the time in seconds:" << t.convert() <<
  "seconds";
  return 0;
}
```

Experiment: 2

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

```
#include <iostream>
using namespace std;
class city
{
string name;
int pop;
public:
void accept()
{
cout<<"enter city name & population";
cin>>name>>pop;
}
void display()
{
cout<<" Name of city having highest population:"<<name;
}
};
int main()
{
city c[5];
int i, max;
for(i=0; i<5; i++)
{
c[i].accept();
}
max=c[0].pop;
```

```
for (i=0 ; i<5; i++)
{
if (c[i].pop > max)
  {
    max=i ;
  }
}
c[max] display ()
return 0;
}
```

Output :- Enter City Name : A
" " Population : 01
" " Name : B
" " Population. : 02
" " Name : C
" " Population : 03
" " Name. D
" " Population ; 04
" " Name : E
" " Population : 05

City Having Highest Population : D.

(2)

```
#include <iostream>
using namespace std;
Class account
{
int acc_no;
float bal;
public :
void accept ()
{
cout<<" Enter account number :" ;
cin >> acc_no;
Cout <<"Enter balance";
cin >> bal;
}
void display ()
{
cout <<" account number:" << acc
cout <<"Balance :" <<bal ;
}
};
Int main ()
{
account A[10] ;
int i;
for (i=0; i<1<, i++)
{
A[i].accept ();
}
for (j=0; i<10; i++)
{
if (A[i] balance >=5000)
{
```

```
A[i].bal = A[i].bal + (0.j * A[i].bal)
A[i].display();
}
}
return 0;
}
```

Output

```
Enter Acc. Number : 2564
Balance : 2094
: 4879
: 7890
: 6789
: 4567
: 3578
: 2478
: 3678
: 2634
: 9705
: 5738
: 5723
: 5835
: 3789
: 4627
: 8906
: 7568
: 4689
```

```
Acc. No. :- 7890
Acc. Balance :- 7467.9
Acc. No :- 2634
Acc. Balance :- 106755
Acc. No :- 5738
Acc Bal :- 62950.8
```

(3)

```
#include<iostream>
using namespace std;
class staff;
{
String name;
string post;
Public:
void accept()
{
Cout<<"Enter staff name";
cin>>name;
cout<<"Enter staff post:";
cin>>post;
}
void display()
{
cout<<"staff name:"<<name;
cout<<"staff post :"<<post;
}
}
int main()
{
staff s[5];
int i;
for(i=0;i<5;i++)
{
s[i].accept();
}
for(i=0;i<5;i++)
{
s[i] accept()
}
}{ s1. display()
```

Output

```
Enter staff name : A
   "    "   post : lecturer
   "    "   name : B
   "    "   post : HOD
   "    "   name : C
   "    "   post : manager
   "    "   name : D
   "    "   post : CEO
   "    "   name : E
   "    "   post : HOD
```

```
staff name : B
   "  post : HOD
   "  name : E
   "  post : HOD
```

2818

Experiment : 3

a)

```
#include <iostream>
using namespace std;
class book
{
string title;
string author_name;
int price;
public;
void accept ()
{
Cout << "Enter book title";
Cin >> title;
Cout << "Enter author name",
Cin >> author name;
Cout << "Enter book price";
cin >> price;
}
void display ()
{
Cout << "Book title" << title;
Cout << "Author name:" << author name;
Cout << "Book price :" << price;
}
}
int main ()
{
book bs;
book *p
p = & b1
p->accept ();
p->display ();
}
```

Output

Enter book title : Defender
" Author Name : Atharva
" Price of book : 2500
Book title : Defender
Author Name : Atharva
Book Price : 2500

b)
```
# include <iostream>
using namespace std;
class student;
{
int roll no ;
float p ;
public :
void accept()
{
cout <<"Enter Roll-no";
Cin >> this -> roll no ;
Cout <<"Enter percentage :" ;
cin >> this -> p ;
}
void display ()
{
this -> accept ();
cout <<" Roll No " << roll no ;
cout <<" percentage "<< p ;
}
};
```

```
int main ()
{
 Student S1;
 S1. display ();
  return 0;
}
```

Output :-

```
Enter Roll-NO : 72
Enter Percentage : 100
Roll_No : 72
Percentage : 100
```

Q.)

```
#include <iostream>
using namespace std;
Class student
{
public :
String name;
int roll;
void accept ()
{
cout << "The name of student << name;
cin >> name
cout << " Enter the roll no. \n";
 cin >> roll no :
}
void display ()
{
cout << "The name of student << name;
 cout << "The roll no \n << roll no;
```

```
};
  class marks
{
public :
  int m1, m2 ;
void accept 1 ()
{
cout << "Enter marks of two students " ;
  cin >> m1, >> m2 ;
};
void display1 ()
{
float per = ((m1+m2) / 100.0) * 100
cout << " The percentage of 2 subject << per
};
};
int main ()
{
student s1;
 student :: marks m;
s1. accept ()
m. accept 1 ()
s1. display 1 ();
m. display 1 ();
return 0;
};
```

26/8

Output

Enter the name : Atharva
Enter roll no : 66
Enter Name : Atharva
Reel No : 66

Enter two marks :
32
16
Percentage : 48

Experiment: 4

D

```
#include <iostream>
using namespace std;
class numbers {
private:
int a,b;
public:
int temp;
void accept() {
cout << "Enter the first number";
cin >> a;
cout << "Enter the second number :";
cin >> b;
}
void display() {
cout << "\n After swapping :" <<endl;
cout << "First number =" << a <<endl;
cout << "Second number =" << b <<endl;
}
void swap (number & n)
{
n.temp = n.a
n.a = n.b;
n.b = n.temp;
}
int main() {
numbers n;
n.accept();
n.swap (n);
n.disp();
return 0;
}
```

Output :-

Enter the first no. = 9
" " second " = 2

After Swapping :
First no : 2
Second i' : 9

---

(2) WAP to swap two numbers from same class using friend function

```
#include <iostream>
using namespace std;
class number {
    int value;
public;
    void value (int v) {
        value = v;
    }
void dispy () {
cout << "value :" << value << endl;
}
friend void swap value (Number of N1, Number of N2) {
};
void swap values (Number of N1, Number of N2) {
    int temp = N1.value;
    N1.value = N2.value
    N2.value = temp;
}
int main.() {
    Number A,B;
A.value (10);
B.value (20);
```

```
A.display ();
B.display ();
swap values (A,B);
Cout << "New values:" << endl;
   A.display ();
   B.display ();
return 0;
}
```

Output :-

A = 10
B = 20

new values :-

A = 20
B = 10

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

```
#include <iostream>
using namespace std;
class ClassB;
class ClassA; {
int numA;
public:
void value (int a){
   numA = a;
}
void display (){
cout << "class A value = " << numA << endl;
}
friend void swap values (ClassA & ClassB);
};
```

```
class ClassB {
  int numB;
Public:
  void values (int b) {
    numB = b;
  }
  void display () {
Cout << " class B value = " << numB << endl;
}
friend void swap values (class A & class B);
};
  void swap values (class A & a, class B & b) {
    int temp = a.numA;
    a.numA = b.numB;
    b.numB = temp;
  }
  int main () {
    class A  A;
    class B  B;
    A.value (20);
    B.value (30);
Cout << " values : " << endl;
A.display ();
B.display ();
Swap values (A,B);
Cout << "swapped values :" << endl;
    A.display ();
    B.display ();
  Swap values A(B);
Cout << "Swapped values :" << endl;
    A.display ();
    B.display ();
  return 0;
}
```

⑨

```
#include <iostream>
using namespace std;
class Result 2;
  class Result 1 {
  float marks1;
  Public ;
void readmarks () {
  cout << " Enter marks for Result 1;
  cin >> marks 1;
}
friend float compute average (Result x1, Result 2 x2);
};
  class Result 2{
    float marks 2;
public;
void readmarks () {
  cout << "Enter marks for Result 2:";
  cin >> marks 1;
}
friend float computing average (result R1, Result R2);
};
float compute average (Result R1, Result R2) {
  return (v, mark 1 + v1. marks 1 ) / 2;
}
int main () {
Result 1 A1;
Result 2 A2;
A1 . readmarks ();
A2 . readmarks ();
float avg = compute average (A1, A2);
Cout << "\n average of two results :" << avg << endl;
return 0;
}
```

Output

Enter marks for Result 1: 10

" " " 2: 20

Average of two results : 15

---

(5) WAP to find the greatest no. among two numbers from diff. classes using friend function.

```
#include <iostream>
using namespace std;
class classB;
class classA;
  int numA;
public:
  void accept(int a){
  numA = a;
}
  void display(){
  cout << "Class A. Number: " << numA << endl;
}
friend void findgreatest(classA, classB);
};
class classB {
  int numB;
Public:
  void accept(int b){
    numB = b;
}
  void display(){
    Cout << "Class B Number : " << numB << endl;
}
  friend void findgreatest(classA, classB);
};
```

```
void find greatest (class A, class B ){
  if (a.numA > b.numB) {
  cout << " Greatest number is from class A." << a.numA << endl;
} else if (b.numB > a.numA) {
  Cout << "Greatest number is: from class B" << b.numB <<
} else {
  Cout << "Both are equal." << endl;
}
}
int main () {
  class A A;
  class B B;
  A. accept (35);
  B. accept (40);
  A. display ();
  B. display ();
find greatest (A,B);
  return 0;
}
```

Output :-

Class A Number : 35
Class B Number : 40

Greatest number is from class B : 42

⑥

```
# include <iostream.>
using namespace std;
class class B;
class class A {
int num A;
public;
void accept (int a) {
num A = a;
}
void display () {
Cout << " Enter Number :" << endl;
Cin >> num A;
}
friend int sum (class A & class B, b);
};
Class class B {
int num B;
Public;
void accept (int b) {
num B = b;
}
void display () {
Cout << "Enter number " << endl;
Cin >> num B;
}
friend int sum (class A &, class B, b);
};
void int sum (class A & Class B) {
return a num A + b num B ;
}
```

```
int main () {
 class A A (28);
  class B B (67);
  int Sum (A &B)
Cout << "Sum = " << sum (A,B) << endl;
 return 0;
}
```

Output:- Sum = 145

---

(7) WAP to swap numbers of same class using friend function

```
#include <iostream>
 using namespace std;
 class Number {
 Private ;
   int value ;
 Public ;
Number ~~int~~ (int val ): value (val) {}
 void show () {
  cout << " Value " << value << endl;
}
friend void swap values ( Number & , Number & );
};
void swap values ( Number 1 & n1 , Number 2 & n2 ) {
int temp = n1.value;
 n1.value = n2.value ;
  n2.value = temp ;
}
```

~~Input~~

```
int main () {
  Number num1 (56);
  Number num2 (44);

cout << " Before Swap :" << endl;
  num1.show ();
  num2.show ();
Swap values (num1, num2);
cout << " Swap :" << endl;
  num1.show ();
  num2.show ();
return 0;
}
```

---

(8)

```
# include <iostream>
using namespace std;
class cube;
class box {
Private ;
  int volume;
Public :
Box (int v); volume(v) {}
friend void find greater (Box & b1, cube & c1);
};
class cube {
Private :
  int volume;
  Public;
   Cube (int v): volume(v) {}
  friend void find greater (Box & b1, cube & c1);
};
```

```
void find greater (Box b1, Cube c1){
 if (b.volume > c.volume) {
cout << "volume of Box is greater" << endl;
} else if (c.volume > b.volume){
  cout << " Volume of cube is greater." << endl;
} else {
 cout << Equal volumes. " << endl;
 }
int main () {
  Box box 1 (50);
   Cube cube 2(50);
find greater (box 1, cube 1);
 return 0;
}
```

---

(9)

```
# include <iostream>
 using namespace std
class complex {
private:
 int real;
 int imag;
public;
complex (int r=0, int i=0 ) { real=r; imag = i; }
 void show () {
 cout << real << "+" << imag << ", << endl; }
 friend complex add ( complex , complex );
};
 Complex add (Complex c1, complex c2) {
 Complex result;
 result.real = c1.real + c2.real;
 result.imag = c1.imag + c2.imag;
 return result;
}
```

```
int main() {
    complex a(2,3);
    Complex b(4,5);
    Complex c = add(a,b);
    Cout << "Sum of complex numbers:";
    c.show();
return 0;
}
```

---

(10)

```
#include <iostream>
using namespace std;
class student {
private:
    string name;
    int marks1, marks2;
    int marks3;
public:
student(string n, int m1, int m2, int m3) {
    name = n;
    marks1 = m1;
    marks2 = m2;
    marks3 = m3;
}
    friend void Calculate average (Student);
};
void calculate average (student s) {
    float avg = (s.marks1 + s.marks2 + smarks3);
cout << "Student name :" << s.sname << endl;
cout << "average marks :" << avg << endl;
}
```

```
int main () {
    Student s1 ("Ani", 85, 90, 95);
    Calculated average (s1);
    return 0;
}
```

---

(11)

```
#include <iostream>
using namespace std;
class Beta;
class Gamma;
class Alpha {
private:
    int a;
public:
Alpha(int val) {
    a = val;
}
friend void total sum (Alpha, Beta, Gamma);
};
class Beta {
private:
    int b;
public:
    Beta (int val) {
        b = val;
    }
friend void total sum (Alpha, Beta, Gamma);
};
class Gamma {
private:
    int c;
```

```
public:
    Gamma(int val) {
        c = val;
    }
    friend void total sum(Alpha, Beta, Gamma);
};
void total sum(Alpha x, Beta y, Gama z) {
    int sum = x.a + y.b + z.c;
    cout << "Sum of all values: " << sum << endl;
}
int main() {
    Alpha A1(10);
    Beta B1(20);
    Gamma C1(30);
    total sum(A1, B1, C1);
    return 0;
}
```

---

(12)

```
#include <iostream>
#include <cmath>
using namespace std;
class point {
private:
    int x, y;
public:
    Point(int x val, int y val) {
        x = x.val;
        y = y.val;
    }
    friend double calculate Distance(Point, Point);
};
```

```
double CalculateDistance(Point p1, Point p2){
    int dx = p2.x - p1.x;
    int dy = p2.y - p1.y;
    return sqrt(dx*dx + dy*dy);
}
int main(){
    Point p1(3,4);
    Point p2(7,1);
    double distance = CalculateDistance(p1,p2);
    Cout << "Distance between Points:" << distance << endl;
    return 0;
}
```

---

(13)

```
#include <iostream>
using namespace std;
Class audit;
class BankAccount {
  private:
int balance;
  public:
BankAccount(int b){
  balance = b;
}
friend void auditBalance(BankAccount, audit);
};
class audit {
Public:
void startAudit(BankAccount ac){
  audit.balance(ac.this);
}
```

```
friend void auditbalance(BankAccount, Audit);
};
void auditBalance(BankAccount, Audit) {
cout << "Auditing Account = " << endl;
cout << "Balance: " << acc.balance << endl;
}
int main() {
BankAccount myACC(150000);
Audit auditor;
auditor.startAudit(myACC);
return 0;
}
```

26/8

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.

(i)

```
# Include <iostream>
using namespace std
class sum {
int n, i, sum = 0;
public:
 Sum ()
{
  n = 20;
}
 void calculate ()
{
 for (i = 1 ; i <= n ; i++)
   {
      Sum = Sum + i ;
   }
}
  void display ()
{
 Cout << " Sum = " << Sum ;
}
};
 int main ()
{
 sum n;
 n. Calculate ();
 n. display ()
 return 0;
}
```

output :- Sum = 210

(ii) Using parameterized constructor :

```
#include <iostream>
using namespace std;
class Sum
{
 int n, i, sum=0;
public :
 Sum (int num)
{
  n = num;
}
 void calculate ()
{
 for (i=1; i<=n; i++)
{
  sum = sum + i;
}
}
 void display ()
{
 cout <<"Sum = " << Sum;
}
};
 int main ()
{
 Sum n (20);
  n.calculate;
  n.display;
 return 0;
}
```

Output : Sum = 210.

b) WAP to declare a class "student" having data members
(i) as name & percentage. Write a constructor to initialize these data members, insert & display data for one student.

```
# include <iostream>
using namespace std;
class student
{
int per;
string per;
public:
student ()
{
name = "dharwa";
per = "91";
}
void display ()
{
cout << "Name :" << name;
cout << "Perantage :" << per;
}
};
int main () {
student s;
s.display ();
return 0;
}
```

Output:-
Name : dharwa
Perantage : 91

(ii) Parametrized Constructor

```
#include <iostream>
using namespace std;
class student
{
 int per;
 string name;
public:
 Student (string n, int percent)
{
 name = n;
 per = percent;
}
 void display () {
 cout << "Name = " << name;
 cout << "Percentage = " << per;
}
};
int main ()
{
Student s ("Dharma", 93)
s.display ();
return 0;
}
```

Output: Name = Dharma
Percentage = 93

c:) Define a class 'College' members variables as roll_no, name, course. WAP using constructor with default value as "Computer Engineering" for course, accept & display data for two objects of a class & display the data.

```
#include<iostream>
#include<string>
using namespace std;
class College {
 private:
int roll_no;
 string name;
 string course;
public:
 College(int r, string n, string c="Computer Engineering"){
 roll_no = r;
 name = n;
 course = c;
}
 void display(){
 cout << " Roll No:" << roll_no << endl;
 cout << " Name:" << name << endl;
 cout << " Course:" << course << endl;
 }
};
int main(){
int roll_no1, roll_no2;
string name1, name2, course1, course2;
cout <<"Enter Roll No, Name & Course for first student:" <<endl;
 cout << " Roll No:";
```

```
cin >> roll no1;
cin.ignore();
cout << "Name:";
getline (cin, name1);
cout << "Course (Press Enter for default):";
get line (cin, course1);

cout << "In Enter Roll No, Name & Course for second student:" << endl;
cout << Roll No:";
cin >> roll_no2;
cin.ignore();
cout << "Name:";
getline (cin, name2);
cout << "Course (Press Enter for default):";
getline (cin, course2);

College student1 (roll_no1, name1, course1.empty() ? "Computer Engineering" : course1;
College student2 (roll_no2, name2, course2.empty() ? "Computer Engineering" : course2;

cout << "In Student 1 Data:" << endl;
Student1.display();

cout << "In Student 2 Data:" << endl;
student2.display()

return 0;
}
```

d) Write a program to demonstrate constructor overload

```
#include <iostream>
using namespace std;
class rectangle
{
int l,b;
public:
rectangle()
{
 l=2;
 b=5;
}
 rectangle (int x)
{
   l=x;
   b=x;
}
rectangle (int x, int y)
{
   l=x;
   b=y;
}
rectangle (rectangle &r3)
{
  l=r3.l;
  b=r3.b;
}
void calculate ()
{
 cout <<" The area is :" << (l*b) << endl;
}
};
```

```
int main()
{
  rectangle r1;
  r1.calculate();
  rectangle r2(3);
  r2.calculate();
  rectangle r3(6,6);
  r3.calculate();
  rectangle r4(r3);
  r4.calculate();
  return 0;
}
```

Output :-

The area is : 10
The area is : 9
The area is : 36
The area is : 36

4/11

Experiment: 6

a) WAP to implement multilevel inheritance. Assume suitable data

Q.1] Create a base class called person with attributes name & age. Derive a class student from person that adds an attribute roll no. write functions to display all details of the student.

→
```
#include <iostream>
using namespace std;
{
  protected :
  string name ;
int 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 : Public Person {
private :
  int roll number ;
public :
void accept student details () {
  accept () ;
cout << " Enter roll number :" ;
  cin >> roll number ;
}
```

```
void display student details () {
  display ();
cout << " Roll Number : " << roll number << endl;
  }
};
int main () {
 student s;
 s.accept student details ();
 s. display student Details ();
 return 0;
}
```

—

Q.2] Create 2 base class academic & sports for multilevel inheritance.

- Academic Class - marks of student
- Sports class - score
- Derived Class - Result inherited from both classes & total score.

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

```
class results : public academic, public sports {
public:
  void display () {
int total = marks + score;
cout << "\n Academic marks!" << marks;
cout << "\n Sports score :" << score;
cout << "\n Total score :" << total << endl;
  }
};
int main () {
Result R;
R.get marks ();
R.get score ();
R.display ();
  return 0;
}
```

—

Q.3) Create a class vehicle with attributes brand & model.
Derive class 1 - attribute size
Derive class 2 - car battery capacity
Display all functions.

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

```
class electric car : public car {
Public :
  int battery capacity ;
  void input () {
  cout << " Enter brand :" ;
  cin >> brand ;
  cout << " Enter model :" ;
  cin >> model ;
  cout << " Enter type :" ;
  cin >> type ;
cout << " Enter battery capacity (Kwh):" ;
  cin >> battery capacity ;
}
void display () {
cout << " \n -- car details --" ;
cout << " Brand :" << brand << endl ;
cout << " Model :" << model << endl ;
cout << " type:" << type << endl ;
cout << " Battery capacity :" << battery capacity << end l ;
}
};
int main () {
Electrical e ;
e.input () ;
e.display () ;
return 0 ;
}
```

Q.4) Hierarchial inheritance

```
#include <iostream>
using namespace std;
class employee {
public:
int empID;
string name;
void setdata (int id, string n) {
emp ID = id;
  name = n;
}

void display () {
cout << "Employee ID:" << empID << endl;
cout << " Name :" << Name << endl;
}
};
class manager : public employee {
public:
string department;
void set department (string dept) {
department = dept;
}
void display () {
Employee :: display ()
cout << " Department : " << department << endl;
}
};
  class developer : public employee
{
Public:
string programming lang;
```

```
void set programing lang (string lang) {
Programming lang = lang;
}
void display () {
employee :: display ()
cout << " Programing language : " programming lang << endl;
}
};
int main () {
manager m;
m.set Data (101, " Atharwa ");
m.set Department (" CEO ");
Developer d;
d.setData (102, " Pathi ");
d.set Programing lang (" Python ");
Cout << " Manager Details : " << endL;
m.display ();
cout << " \n Developer Details : " << endl;
d.display ();
return 0;
}
```

8.5) Hybrid Inheritance

```
#include <iostream>
using namespace std;
class person {
public :
string name;
int age;
void set Person (string n, int a) {
name = n;
age = a;
}
```

```
void displayPerson () {
cout << "Name :" << name << endl;
cout << " Age " << age << endl;
}
};
class Student : public Person
{
public :
int students ID;
void setStudent (int ID) {
  student ID = 10;
}
void display student () {
cout << " Student ID :" << student ID << endl;
}
};
class sports
{
public :
int sports score;
void setscore (int score) {
  sports score = score;
}
void display score ()
{
cout << " Sports score :" << sports score << endl;
}
};
class Result : public student, public sports
{
public :
  int marks;
```

```
void setMarks (int m) {
marks = m;
}
void
int calculateTotal () {
return marks + sportsScore;
}
void displayResult () {
    displayPerson ();
    displayStudent ();
    displaySports();
Cout<<"marks :"<<marks<<endl;
cout <<"Total (marks + sports score) :" << calculateTotal<<endl;
}
};
int main () {
Result R;
R.setPerson ("BOB", 35);
R.setStudent (102)
R.setScore (45);
R.setMarks (90);
R.displayResult ();
return 0;
}
```

4/11

Experiment: 7

a) WAP using function overloading to calculate the area of a laboratory (which is rectangular in shape) & area of classroom (which is square in shape)

```
#include <iostream>
using namespace std;
class area
{
  private:
  float l,b;
  public:
  void area (float l)
  {
    float area;
    area = l*l;
    cout << "area of square: " << area << endl;
  }
  void area (float l, float b)
  {
    float area;
    area = l*b;
    cout << "area of rectangles: " << area;
  }
};
int main ()
{
  area a1;
  a1.area (8);
  a1.area (4,12);
  return 0;
}
```

Output:- area of square: 64
area of rectangle: 48

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

```
#include <iostream>
using namespace std;
class sum
{
  private:
int a,b,c,d,e,f,g,h,i,j;
float k,l,m,n,o;
public:
void sum (float k, float l, float m, float n, float o)
{
  float sum;
sum = k+l+m+n+o;
cout << "Sum of 5 floating numbers:" << sum << endl;
}
void sum (int a, int b, int c, int d, int e, int f, int g, int h, int i
  , int j)
{
  int sum;
  sum = a+b+c+d+e+f+g+h+i+j;
  cout << "Sum of 10 integers:" << sum;
  }
};
int main ()
{
  sum s1;
s1.sum (1.2, 3.5, 5.6, 8.4, 1.5);
s1. sum (1, 2, 3, 4, 5, 6, 7, 8, 9, 10); 97
  return 0;
}
```

Output :- Sum of 5 floating numbers : 20.2
sum of 10 integers : 55

Exp 8

*) WAP to overload '+' operator so that two strings can be concatenated. eg "xyz" + "pqr" then output will "xyzpqr".

```
#include <iostream>
#include <string>
using namespace std;
class mystring {
    public: string str;
    mystring operator + (mystring obj) {
        mystring temp; temp.str = str + obj.str;
        return temp;
    }
    void display () { cout << str << endl; }
};
int main () { mystring s1, s2, s3;
    s1.str = "xyz"; s2.str = "pqr";
    s3 = s1 + s2
    cout << "concatenated string : ";
    s3.display ();
    return 0;
}
```

output :

concatenated string : xyzpqr

---

(2)

```
#include <string>
#include <iostream>
using namespace std;
class login {
    public: virtual void accept () {
        cout << "Enter name : "; cin >> name;
        cout << "password "; cin >> password;
    }
};
```

```
class emaillogin : public login {
    string email;
public : void accept () override {
    login :: accept ()
    cout << " enter email :"; cin >> email;
}
void display () {
    cout << " Name :" << ename << " password" << password << "
    "email" << email << endl;
}
};
class membership login : public login {
    string membership;
public : void accept () override {
    login : accept ();
    cout << " inter membership ID : ";
        cin >> membership ID;
}
void display () {
    cout << "\n --- Membership details ---\n";
    cin >> membership ID; cout << "Name :" name << "\n password :"
    << password << " membership :" << membership ID << endl;
}
};
int main () { emaillogin membership login m;
    l.accept (); m.accept (); l.display (); m.display ();
    return 0;
}
```

output :-

enter name :- Atharva
enter password :- 123
enter email :- abc@gmail.com
enter name :- Atharva
password :- 745
enter membership ID :- 5745

--- email login details ---
Name :- Atharva
password : 123
email : abc@gmail.com

--- membership login details :- ---
Name : Atharva
password : 745
membership ID :- 5745

Exp: 9

a) WAP to copy the contents of one file into another

```
#include <iostream>
#include <fstream>
using namespace std;
int main () { fstream first-file;
        fstream second-file;
        firstfile.open ("first text", ios::in);
    if (! first-file) {
    cout << "error " << endl;
     return 1;
    }
  second-file open ("second text", ios::out);
   if (! second-file) {
     cout << "error" << endl;
    return 1;
   }
  while (first.file get (ch) {
        second_file.put(ch); }
  cout << " file copied sucessfully !" << endl;
   return 0;
  }
```

b) WAP to count digits & spaces using file handeling ?

```
#include <iostream>
#include <fstream
#include <cctype>
using namespace std;
int main () { fstream new_file;
     new-file.open ("first.text", ios::in);
   if (! new.file) {
    cout << " error" << endl;
     return 1;
  }
```

```
int digits_count = 0;
int spaces_count = 0;
char ch; while (new_file.get(ch)){
  if (isdigit(ch)){
    digits_count++;
  }
  if (isspace(ch)){
    spaces_count++;
  }
new_file.close();
cout << "No. of digits: " << digits_count << endl;
cout << "No. of spaces: " << spaces_count << endl;
return 0;
}
```

c) WAP to count words using file handling?

```
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;
fstream new_file;
new_file.open("file.text", ios::in);
if (!new_file){
cout << "Enter correct file" << endl;
return 1;
}
char ch;
int words_count = 0;
bool inword = false;
while (new_file.get(ch)){
if (space(ch)){
  inword = false;
```

```
        else if (inWord)
        {
            words_count++;
            inWord = false;
        }
    }
};
new_file.close();
cout << "No. of words: " << words_count << endl;
return 0;
}
```

d) WAP to count occurances of a given word.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
    fstream new_file;
    new_file.open("first.text", ios::in);
if (!new_file
{
    cout << "error" << endl;
    return 1;
}
string target = "[redacted]" "Atharwa"
string word;
    int count = 0;
while (new_file >> word)
{
    if (word == target)
    {
```

```
        count++;
    }
}
new_file.close();
cout << "the word " << target << " occurs " <<
    count << " times" << endl;
return 0;
}
```

exp. 10

1) WAP to find sum of array using function template.

```
#include <iostream>
using namespace std;
template <class t>
t sum (t arr[10])
{ int i; t add=0;
  for (i=0; i<10; i++) { add= add + arr[i]; }
  return add;
}
int main () {
  float arr[] = {12, 1, 3, 5, 6, 3, 4, 9, 10, 14, 15};
  cout << "\n sum of array : " << sum (arr);
  return 0;
}
```

2) WAP of square function using template specialization.

```
#include <iostream>
using namespace std;
template <class t> square (num);
{
  return num * num;
}
template <> string square (string &str)
{
  return str + str;
}
int main () { int n= 5; string t = "Hello";
  cout << " square of no. : " << square (n);
  cout << " square of string : " << square (t);
  return 0;
}
```

3) WAP to built simple calculator using class template.

→ # include <iostream>
using namespace std;
template <class T> class c
{
public : T num1=0 ; T num2=0;
void accept () { cout << " enter nos :" ;
cin >> num1 >> num2 ;
}
void acc () { cout << " sum : " << num1+num2;}
void sub () { cout << " sub : " << num1 - num2;}
void mult() { cout << " multi: " << num1 * num2;}
void div () { cout << " div : " << num1/num2;}
};
int main () {
c <int> obj ;
obj . acc (); add (); obj . sub (); obj . mult ();
obj . div ();
return 0;
}

—

4) # include <iostream>
using namespace std ;
template <class T > class stack {
private : T * arr ; int top ; int capacity ;
public : stack (int size) {
arr = new [(size)] ; capacity = size ; top = -1 ;
}
stack () { delete [] arr ; }
void push (T value) {
if (top == capacity -1) {
cout << " Stack overflow ! Cannot push " << value << endl ;
return ;
}

```
arr [++top] = value;
cout << value << " pushed into stack \n";
}

T. pop () {
  if (top == -1) {
    cout << "stack underflow ! \n";
    return ();
  }
  cout << arr [top] << " poped from stack \n";
  return arr [top--];
}
bool is empty () { return top == -1; }
void display () { if (is Empty ()) {
    cout << "stack is Empty \n";
    return ;
  }
  cout << " stack is elements :";
  for (int i=0; i<= top; i++)
    cout << arr[i] << " ";
  cout << endl;
}
};

int main () {
  stack <int> int stack (5);
  intstack . push (10);
  intstack . push (20);
  intstack . push (30);
  intstack . display ();
  intstack . pop (); intstack . display ();
  stack <string> string stack (5);
  string stack . push ("Apple");
  string stack . push ("Banana");
  string stack . display ();
  string stack . pop ();
  string stack . display ();
  return 0;
}
```

exp. 11

Q) Without Iterator :-

→
```
#include <iostream>
#include <vector>
#include <cctype>
using namespace std;
int main () {
  vector <char> v(0); unsigned int i;
  cout << "size = " << v.size () << endl;
  for (i=0 ; i<10 ; i++) { v[i] = i+'0'; }
  cout << " current content :\n";
  for (i=0 ; i<10 ; i++) { v.push_back (i+10, +'a'); }
    cout << "size now = " << v.size () << endl;
    cout << " current content :\n";
  for (i=0 ; i<v.size () ; i++) { cout << v[i] << " "; }
    cout << "\n\n";
  for (i=0 ; i<N.size () ; i++) { v[i] = toupper (v[i]); }
    cout << " modified contents :\n";
  for (i=0 ; i<N.size(); i++) { cout << v[i] << " "; }
    return 0;
}
```

With Iterators

```
#include <iostream>
#include <vector>
using namespace std;
int main () {
    vector <char> v(10);
    vector <char> :: iterator p;
```

```
int i;
p = v.begin();
i = 0;
while (p != v.end()) {
    *p = i + 'a';
    p++;
    i++;
}
cout << "original content:\n";
p = v.begin();
while (p != v.end()) {
    cout << *p << " ";
    p++;
}
```

Exp. 12

8.1) Implement stack

```
#include <bits/stdc++.h>
using namespace std;
int main () {
    stack <char> cars;
    cars.push ("BMW"); cars.push ("Audi");
    car.push ("Mercedes"); car.push ("Ferrari");
Cout << "Top element is:" << cars.top () << endl;
Cout << "Size of stack is :" << cars.size () << endl;
 cars.pop (); cars.pop ();
 while (! cars.empty ()) {
    Cout << "elements in stack are:" << cars.top () << endl;
    cars.pop ();
  }
 return 0;
}
```

Output :-

Top element : Ferrari
Size of stack is : 4
Elements in stack are : Audi
" _____ " : BMW

Q.2) Implement Queue :-

```
#include <bits/stdc++.h>
using namespace std;
int main (){
  queue <int> age;
  age.push(21); age.push(22); age.push(23); age.push(24);
  cout <<" front element is :" <<age.front() << endl;
  Cout <<" Book element is :" << age.book() << endl;
  age.pop(); age.pop();
  while(! age.empty());
  {
    cout <<" element in queue are :" <<age front() << endl;
   age.pop(); }
   return 0;
}
```

Output :-

front element is : 21
back element is : 24
Elements in queue are : 23
Elements in queue are : 24