Prerequisites

1. Write a c++ program to add two numbers.
2. Arithmetic operations (+, -, /, *) using switch.
3. Check no. is even or odd
4. Print 1 to 10 nos. using 'for' loop
5. -11- using 'while' loop
6. Print below patterns

a)
```
*
* *
* * *
* * * *
```

b)
```
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
```

1. Ans
```
#include <iostream>
int main() {
~~int main() {~~
int a, b, c;
std:: cout << "enter values of a and b:";
std:: cin >> a >> b;
c = a+b;
std:: cout << "addition is " << c;
return 0;
}
```

* Output
enter values of a and b: 12 1
addition is 13

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

```
char operation;
double num1, num2;

cout << "enter first number : ";
cin >> num1;
cout << "enter an operator (+,-,/,*) : ";
cin >> operator;
cout << "enter second number : ";
cin >> num2;

switch (operator) {
   case '+' :
      cout << "result = " << num1 + num2 << end1;
      break;
   case '-' :
      cout << "result = " << num1 - num2 << end1;
break → case '*' :
      cout << "result = " << num1 * num2 << end1;
      break;
   case '/' :
      cout << "result = " << num1 / num2 << end1;
      break;
   default;
      cout << "invalid operator ! " << end1;
 }
 return 0;
 }
```

* Output

```
enter first number : 12
enter an operator (+,-,*,/) : /
enter second number : 3
```

result = 4

3.
```
#include<iostream>
using namespace std;
int main(){
  int num1;
  cout<<" enter any number:";
  cin>>num1;
  if (num1%.2==0){
    cout<<"number is even";
  }
  else{
    cout<<"number is odd";
  }
return 0;}
```

* ~~Operato~~ Output
enter number: 5221
number is odd.

4.
```
#include<iostream>
int using namespace std;
 int main(){
   int i;
   for(i=1; i<=10; i++){
     cout<<i<<"\t";
   }
return 0;}
```

* Output
1 2 3 4 5 6 7 8 9 10

5.
```
#include<iostream>
using namespace std;
int main(){
  int i;
  while (i<=10){
    cout<<i<<"\t";
    i++;
  }
```

* Output

1 2 3 4 5 6 7 8 9 10

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

* Output

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

b)
```
#include<iostream>
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)
```
#include<iostream>
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
```

3117

Experiment 1.

Q2. WAP to declare a class Book having data members as book price, book name, book pages. Accept data for 2 books & display name of book having greater price.

```
#include <iostream>
#include <string>
using namespace std;
class Book {
  public:
    int price;
    int pages;
    string name;
    void accept(){
      cout << "enter book name: ";
      cin >> name;
      cout << "enter price";
      cin >> price;
      cout << "enter pages : ";
      cin >> pages;
    }
    void display(){
      cout << "book name is :" << name;
      cout << "Book price is :" << price;
      cout << "pages of book are :" << pages;
    }
    int get_price(){
    return price; }
};
int main(){
  book b1, b2;
  b1.accept();
  b2.accept();
```

```
if (b1.get_price() > b2.get_price()) {
    cout << "book1 has greater price\n";
    b1.display();
}
else {
    cout << "book2 has higher price\n";
    b2.display();
}
return 0;}
```

* Output

```
enter book name : programming_in_C
enter price     : 120
enter pages     : 230
enter book name : Matilda
enter price     : 230
enter pages     : 120
book 2 has higher price
book name is : Matilda
book price is : 230
pages of book are : 120
```

Q1. WAP to declare a class student having data members as roll, name. Accept & display data for one object.

```
# include <iostream>
# include <string>
using namespace std;
class student {
public:
  int roll_no;
  string name;
```

```
void display (){
  cout << "name is :" << name << endl;
  cout << "roll no. is :" << roll_no << endl;
}};

int main(){
  student s1;
  s1.name = "Aanya";
  s1.roll_no = 1;
  s1.display();
  return 0;
}
```

* Output
name is Aanya
roll no is 1

Q3. Write a program to declare a class ~~student~~ ~~havi~~ time. Accept time in HH:MM:SS & display ~~total~~ time in seconds.

```
#include<iostream>
using namespace std;
class time{
  public:
  int hours, mins, secs, totaltime;
  char col1, col2;
  void input(){
  cout << "enter time in HH:MM:SS format:";
  cin >> hours >> col1 >> mins >> col2 >> secs;
  }

  void calculate(){
```

```
totaltime = (hours * 3600) + (mins * 60) + secs;
}
void display() {
  cout << "total time in seconds : " << totaltime;
}};

int main() {
Time t;
t.input();
t.calculate();
t.display(); }
```

* Output

enter time in HH:MM:ss format:
2:30:15
total time in seconds : 9015

classmate
Date
Page

Experiment 2

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

```
#include<iostream>
using namespace std;
class city{
  private:
    string name;
  public:
    int population;
    void accept(){
      cout<<"enter name:";
      cin>>name;
      cout<<"enter population:";
      cin>>population;
}
    void display(){
      cout<<"name:"<<name<<", population"<<
population<<endl;
}};

int main(){
city c[5];
  int i,max;
    for(i=0;i<5;i++){
      c[i].accept();
      c[i].display();
}
max=c[0].population;
for(i=1;i<5;i++){
    if(c[i].population>max){
      max=c[i].population;
```

}}
cout << "max population is :" << max << endl;
return 0; }

* Output
enter name: pune
enter population: 12000
name: pune, population : 12000
enter name : mumbai
enter population: 150000
enter name : mumbai, population: 150000
enter name : nagpur
enter population : 3400
name: nagpur, population : 3400
enter name : alibaug
enter population : 34000
enter name : alibaug, population : 34000
enter name : nashik
enter population : 1000
name: nashik, population : 2000
max population: 150000

b) 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 or greater than 5000 & display them.

```
#include <iostream>
using namespace std;
class Account {
  private:
    int Account_no, balance;
```

```
public:
  void accept(){
    cout<<"enter account no: ";
    cin >> Account_no;
    cout<<"enter balance :";
    cin>> balance;
  }
  void check(){
    if (balance <= 5000){
      balance += balance * 0.5; }
  }
  void display(){
    if (balance <= 5250){
      cout <<"account no:"<< Account_no <<endl;
      cout <<"balance :" << balance<<endl; }
  }
};
int main(){
Account c[10];
int i;
for(i=0; i<10; i++){
  cout<<"enter details for account :"<<i<<endl;
c[i].accept();
}
for (i=0; i<10; i++){
  c[i].check();
}
for (i=0; i<10; i++){
  c[i].display();
}
return 0;
}
```

* Output

enter details for account : 0
enter account no : 1
enter balance : 1200
-11- details for account : 1
-11- account no : 2
-11- balance : 2000
-11- details for account : 2
-11- account no : 3
-11- balance : 3400
-11- details for account : 3
-11- account no : 4
-11- balance : 5000
-11- details for account : 4
-11- account no : 5
-11- balance : 5000
-11- details for account : 5
-11- account no : 6
-11- balance : 4000
-11- details for account : 6
-11- account no : 7
-11- balance : 10000
-11- details for account : 7
-11- account no : 8
-11- balance : 6000
-11- details for account : 8
-11- account no : 9
-11- details for balance : 1000
-11- details for account : 9
-11- account no : 10
-11- balance : 23000

account no: 1
bal balance : 1260
account no: 2
balance : 2100
account no: 3
balance: 3570
account no: 4
balance : 5250
account no: 5
balance : 5250
account no: 6
balance : 4200
account no: 9
balance : 1050

c) WAP to declare a class 'staff' having data members as name & post. Accept this data for 5 staff & display names of staff who are 'HOD.'

```
#include <iostream>
#include <string>
using namespace std;
class staff {
 private:
   string name, post;
   public:
      void accept() {
         cout << "enter name :"; // enter without spaces.
         cin >> name;
         cout << "enter post :";
         cin >> post; // enter teacher, lecturer or HOD
}
```

```
void check(){
    if(post=="HOD" || post=="head"){
    cout<<" head staff name : "name<<endl;
    }
}};
int main(){
Staff c[5];
int i;
for(i=0; i<5; i++){
   c[i].accept();
}
for(i=0; i<5; i++){
   c[i].check();
}return 0;
}
```

* Output

```
enter name : Mrunal_Aware
-11-  post : HOD
-11-  name : Pallavi_Nehete
-11-  post : HOD
-11-  name : Vilas_Rathod
-11-  post : lecturer
-11-  name : Megha_Dit Dhotay
-11-  post : teacher
-11-  name : Shilpa_Budhavale
-11-  post : HOD
head staff name : Mrunal_Aware
   -11-         : Pallavi_Nehete
   -11-         : Shilpa_Budhavale
```

3/11/

Experiment 3

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

```
#include<iostream>
#include<string>
using namespace std;
class Book{
  private:
    string b_name, a_name;
    int price;
  public:
    void accept(){
      cout<<"enter book name, author's name and price:"; // type without spaces
      cin>>b_name>>a_name>>price;
    }

    void display(){
      cout<<"Book name:"<<b_name<<endl;
      cout<<"author's name:"<<a_name<<endl;
      cout<<"price:"<<price;
    }
};
int main(){
  Book b1;
  Book *p;
  p = &b1;
  p->accept();
  p->display();
  return 0;
}
```

* Output
enter book name, author's name & price: Anna_Karenina
Leo Tolstoy 300
book name: Anna_Karenina
author's name: Leo_Tolstoy
price: 300

b) WAP to declare a class 'student' having data members roll no, percentage. Using 'this' pointer invoke member functions to accept & display this data for one object of the class.

```
#include<iostream>
#using namespace std;
class Student {
  private:
    int rollno;
    float percentage;
  public:
    void accept(){
      cout<<"enter student's roll no & percentage:";
      cin>>rollno>>percentage;
}
    void display(){
      this->accept();
      cout<<"roll no:"<<rollno<<endl;
      cout<<"percentage:"<<percentage<<endl;
};
int main(){
  Student s1;
  s1.display();
  return 0;
}
```

enter student's roll no & percentage : 31 98.7
roll no : 31
percentage : 98.7

c) write a program to demonstrate use of nested class.

```
#include<iostream>
using namespace std;
class student{
  int roll;
string ~~char~~ name;
  public:
  void accept(){ cout<<"enter roll no: ";
    cin>>roll;
    ~~cin>>name;~~ cout<<"enter name :";
  } cin>>name; }

    void display(){
    cout<<"name :"<<name<<endl;
    cout<<"roll no: "<<roll<<endl;
}

~~public~~:
class Marks { public:
int m1, m2, avg;
void accept(){ cout<<"enter m1 :";
cin>>m1;
~~cin>>m2;~~ cout<<"enter m2 :";
} cin>>m2; }

void calculate_avg(){
avg = (m1+m2)/2;
return avg;
}
```

```
void display1() {
cout << "average of marks is : " << avg;
}};
};

int main () {
student s1;
s1.accept ();
Student :: Marks s2;
s2. accept1 ();
s2. display1 ();      s1. display ();
s2. calculate_avg ();
return 0;      s2. display ();
Return 0; }
```

* Output

```
enter roll no: 1
enter name : Aarya
enter m1 : 22
enter m2 : 22
name : Aarya
roll no: 1
average of marks is : 22
```

Experiment 4.

a) WAP to create two classes result1 & result2 which stores the marks of students. Read the values of marks for both class objects & compute the average of two results.

```
#include <iostream>
using namespace std;
class result1 {
public:
  int m1, m2;
void read (){
  cout<<"enter marks for result1 :";
  cin>> m1 >> m2;
}
void float avg () {
  return (m1 + m2) / 2;
}
};

class result2 {
public:
  int m1, m2;
void read(){
  cout <<"enter marks for result2 :";
  cin>> m1 >> m2;
}
float avg(){
  return (m1 + m2)/2;
}
};
```

```
void compute_avg (result1 r1, result2 r2) {
    cout << "average of result1 : " << r1.avg () << endl;
    cout << "average of result2 : " << r2. avg () << endl;
}
int main(){
result1 r1,
result2 r2;

r1. read ();
r2. read ();
compute_avg(r1, r2);
return 0; }
```

* Output

```
enter marks for result1 : 12 23
enter marks for result2 : 12 23
Average of result1 : 17.5
Average of result2 : 17.5
```

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

```
#include <iostream>
using namespace std;
class class2; // forward declaration
class class1 {
 public:
   int n1, n2, greatest;
   void accept () {
    cout << "enter two numbers: ";
     cin >> n1 >> n2;
}
```

```
void display(){
  cout<<"greatest of two numbers is :"<<greatest<<endl;
}

friend void find_greatest(class1 &c1, class2 &c2);
};

class class2{
public:
int n1, n2, greatest;
void accept(){
  cout<<"enter two numbers : ";
  cin>>n1>>n2;
}
void display(){
  cout<<"greatest of two numbers is :"<<greatest<<endl;
}
friend void find_greatest(class1 &c1, class2 &c2);
};

void find_greatest(class1 &c1, class2 &c2){
c1.greatest = (c1.n1 > c1.n2) ? c1.n1 : c1.n2;
c2.greatest = (c2.n1 > c2.n2) ? c2.n1 : c2.n2;
}

int main(){
class1 c1;
class2 c2;
c1.accept();
c2.accept();
find_greatest(c1,c2);
```

```
c1.display();
c2.display();
return 0;
}
```

* Output

```
Enter two numbers : 23  45
Enter two numbers : 12  78
The greatest of two numbers : 45
        -11-                : 78
```

c) WAP for swapping contents of two variables of same class using friend function.

```
#include<iostream>
using namespace std;
class temp{
int x, y, q;
public:
 void input(){
   cout<<"enter two numbers :";
   cin>> x>>y;
 }
 void display(){
   cout<<"after swap, x : "<< x;
   cout<<"after swap, y : "<< y;
 }friend void swap(temp &t);
 };

void swap(temp &t) {
t.q = t.x;
t.x = t.y;
t.y = t.q; }
```

```
int main(){
temp t1;
t1.input();
t1.dr swap(t1);
t1.display()
return 0;
}
```

* Output

enter two numbers : 1 18
after swap, x : 18
after swap, y : 1

d) WAP to swap two numbers from same class using object as function argument. Write swap function as member function.

```
#include <iostream>
using namespace std;
class swap{
  int a;
public:
  void input(int x){
    a=x;
  }
  void display(){
    cout<<" value :"<<a<<endl;
  }
  void swap(swap &s){
    int t=a;
    a=s.a;
    s.a=t;
  };
```

```
int main(){
swap s1, s2;
s1.input(5);
s2.input(10);
cout << "before swap:" << endl;
s1.display();
s2.display();

s1.swap(s2);
cout << "after swap:" << endl;
s1.display();
s2.display();
return 0;
}
```

* Output

```
Before swap:
value: 5
value: 10
After swap:
value: 10
value: 5
```

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

```
#include <iostream>
using namespace std;
class ClassB;
class ClassA {
private:
  int n1;
```

```
public:
  void set(int a){
    n1 = a;
  }
  void display(){
  cout << "class_A number:" << n1 << endl;
  }
  friend void swap(class_A &, classB &);
};

class classB{
private:
int n2;
public:
  void set(int b){
    n2 = b;
  }
  void display(){
  cout << "classB number:" << n2 << endl;
  }
  friend void swap(class_A &, classB &);
};

void swap(class_A &a, class_B &b){
  int temp = a.n1;
  a.n1 = b.n2;
  b.n2 = temp;
}
int main(){
class_A a;
class_B b;
```

```
a.$et (5);
b.set(10);
cout <<" before swapping :" <<endl;
a.display();
b.display();
swap(a,b);
cout <<"after swapping :"<<endl;
a.display();
b.display();
return 0;
}
```

* Output

Before swapping:
class A number : 5
class - B number : 10
after swapping:
class - A number : 10
class - B number : 5

Qn

26 B

Experiment No. 5

Q1. a) Write a program to find sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.
(Default Constructor)

```
#include<iostream>
using namespace std;
class Sum {
int num, s;
public:
 Sum() {
  s = 0;
  num = 5;
 for(int i=1; i<num; i++) {
 s = s+i; }}

 void display () {
    cout << "Sum = " << s;
 }
 ~Sum() {
    cout << "Object destroyed :";
 }
};
int main() {
Sum s1;
s1.display();
return 0;
}
```

* Output

Sum = 10
Object destroyed

b) Parameterized Constructor

```
#include <iostream>
using namespace std;
class Sum {
  int num;
public:
  sum(int n) {
    num = n;
    int s = 0;
    for(int i = 1; i < sum; i++) {
      s += i;
    }
  ~sum() {
      cout << "object destroyed :";
}};
int main() {
  Sum s1(5);
  return 0;
}
```

* Output

sum = 10
Object destroyed.

c) (copy constructor)

```
#include <iostream>
using namespace std;
class Sum {
int n, total;
public:
sum(int num) {
n = num;
```

```
total = 0;
for(int i=1; i<=n; i++) {
    total += i;
}}
sum(const sum & obj) {
    n = obj.n;
    total = obj.total;
}
void display() {
cout << "sum = " << total;
}
~sum() {
    cout << "object destroyed:";
}};
int main() {
  int num;
  cout << "enter a number ";
  cin >> num;
  sum s1(num);
  sum s2 = s1;
  s2.display();
  return 0;
}
```

* Output

sum = 10
object destroyed
object destroyed.

Q2. a) WAP to declare a class 'student' having data members as name & percentage. Write a constructor to initialize these data members. Accept & display

data for one student.

```
#include <iostream>
using namespace std;
class student {
string name;
    float percentage;
public:
  student () {
    percentage = 70;
    name = "Aarya";
}
void display () {
  cout << "name = " << name << endl;
  cout << "percentage = " << percentage << endl;
}
~ student () {
  cout << "object destroyed";
  }};
int main () {
  student s1;
  s1.display ();
  return 0;
}
```

* Output

name = Aarya
percentage = 70
object destroyed.

Q3. Define class 'college' number as roll_no, name, course. WAP using constructor with default value as "computer engineering" for course. Accept & display data for 2 objects.

```
#include <iostream>
using namespace std;
class college {
  int roll;
  string name, course;
public:
college (int r, string n, string c = "computer engineering") {
  roll = r;
  name = n;
  course = c;
}
void display () {
  cout<<" name, roll_no & course is :"<< name<<
  roll<< course;
}
};
int main(){
  college c1 (1, "Aarya", "CSE");
  college c2 (1, "Aditya");
  c1.display ();
  c2.display ();
  return 0; }
```

Output

name, roll-no & course : Ac Aarya, 1 & CSE
name, roll_no & course : Aarya, 1

Q4. WAP to demonstrate constructor overloading.

```
#include <iostream>
using namespace std;
class rectangle {
    int l, w;
public:
    Rectangle () {
        l = 2;
        w = 1;
    }
    Rectangle (int a) {
        l = a;
        w = b;
    }
    void calculator () {
        int a;
        a = l * w;
        cout << "Area = " << a; }
};
int main () {
    Rectangle r1;
    Rectangle r2(5);
    Rectangle r3(4,5);
    r1.calculate ();
    r2.calculate ();
    r3.calculate ();
    return 0;
}
```

26/9/25

* Output

Area = 2 Area = 25 Area = 20

Experiment No. 6

1. Single Inheritance.

Q. Create a base class called Person with name & age. Derive a class student that adds attribute roll no. Write functions to display all details of the student.

```
#include <iostream>
using namespace std;
class Person {
protected:
string name;
int age;
};
class student : protected person {
private:
int roll;
public:
void accept() {
cout << "enter name:";
cin >> name;
cout << "enter age:";
cin >> age;
cout << "enter roll no:";
cin >> roll;
}
void display() {
cout << "name:" << name;
cout << "age:" << age;
cout << "roll no:" << roll;
}
};
```

```
int main(){
 student s1;
 s1.accept();
 s1.display();
 return 0;
 }
```

2. Multiple Inheritance.

Q. Create two base classes Academic & Sports:

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

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

```
#include <iostream>
using namespace std;
class Academic {
protected:
 int marks;
};
class Sports {
protected:
 int score;
};
class result : protected Academic, protected sports {
private:
int totalscore;
public:
void accept(){
 cout<<" enter student marks: ";
 cin>> marks;
 cout<<"enter sports score : ";
```

```
cin>> score;
}
void calculate() {
total score = marks + score;
}
void display () {
cout<<" student marks = "<< marks <<endl;
cout<<" sports score = "<< score <<endl;
cout<<" Total score = "<< totalscore;
};
int main() {
result r1;
r1.accept();
r1.~~accep~~ calculate();
r1. display ();
return 0;
}
```

* Output

```
Enter student marks : 98
Enter sports score : 89
Student marks = 98
Sports score = 89
Total score = ~~189~~ 187
```

* Multilevel Inheritance

Q. Create a class Vehicle with attributes like brand & model.
Derive a class car from vehicle which adds an attribute type eg (suv, sedan) ~~two~~ further derive a class electric car which adds battery capacity. Write functions to display all the details.

```
#include <iostream>
using namespace std;
class Vehicle {
public:
string brand;
string model;
};
class car : public vehicle {
public:
string type;
};
class electric_car : protected car {
private:
int battery_capacity;
public:
void accept () {
cout << "enter vehicle brand: ";
cin >> brand;
cout << "enter vehicle model ";
cin >> model;
cout << "enter cartype : ";
cin >> type;
cout << "enter battery capacity : ";
cin >> battery_capacity;
}
void display () {
cout << "vehicles brand : " << brand;
cout << " vehicle model : " << model;
cout << "car type :" << type;
cout << "battery capacity :" << battery_capacity;
}};
```

```
int main(){
electric car electriccar ec1;
ec1.accept();
ec1.display();
return 0;
}
```

* Output

```
Enter vehicle brand: Tesla
Enter vehicle model: 3
Enter car type: sedan
Enter battery capacity: 57.5
Vehicle brand: Tesla vehicle. model: 3
cartype: sedan                battery capacity: 57.5
```

* Hierarchical Inheritance:

Create base class Employee with empID & name. Derive two classes manager & developer from Employee. Manager with department, developer with programming language. Write functions to display details for both.

```
#include <iostream>
using namespace std;
class Employee {
  Protected:
  string name;
  int empID;
};
class Manager: protect Employee {
private:
string dept;
public
void accept() {
```

```
cout<<"enter emp name :";
cin>> name;
Cout<< "enter empID :";
cin>> empID;
cout<<" enter department :";
cin>> dept;
}
void display (){
cout<<" emp name = "<< name;
cout<<" emp id = "<< empID;
cout<<"department = "<< dept;
}};
class Developer : protected depar Employee {
private:
string programming_lang;
public:
void accept (){
cout<<"enter programming language :";
cin>> programming_lang;
}
void display(){
cout<<" programming language = "<< programming_lang;
}};
int main(){
  manager m1;
  m1.accept();
  m1.display ();
 Developer d1;
 d1.accept ();
 d1.display ();
 return 0;
}
```

* Output
Enter name = John
Enter empID = 101
Enter dept = HR
Name = John EmpID = 101 Dept = HR
Enter programming language = C++
Programming lang = C++

Q5. * Hybrid Inheritance:
Q Combine Multilevel & multiple inheritance. Create a base class person with name & age. Derive class Student from person. Create two classes sports & Academics. Derive class Result from Student & Sports.

```
#include <iostream>
using namespace std;
class Person {
Protected:
String name;
int age;
public:
void get_data() {
cout << "enter name ";
cin >> name;
cout << " enter age ";
cin >> age;
}
void show_data() {
cout << "name " << name << endl;
cout << "age :" << age << endl;
};
class student : public person {
protected:
```

```
int roll_no;
public:
void get_student_data() {
  cout << "enter roll_no:";
  cin >> roll_no;
}
void show_student_data(){
  cout << "roll no : " << roll_no << endl;
};
class sports {
protected:
  int sports_marks;
public:
  void get_sports_marks() {
    cout << "enter sports marks :";
    cin >> sports_marks;
}
  void show_sports_marks(){
    cout << "sports marks: " << sports_marks << endl;
};};

class Academics {
protected:
int academic_m;
public:
void get_academic_m() {
  cout << "enter academic marks :";
  cin >> academic marks;
}
  void show_academic_m() {
    cout << "academic marks :" academic_m << endl;
};};
```

```
class Result : public student, public sports, public
Academics {
int total;
public:
  void calculate_result() {
    total = sports + academic_marks;
}
void display_result() {
  show_data();
  show_student_data();
  show_sports_data
                marks()

  show_academic_marks();
  cout << " Total marks : " << total << endl;
}};
int main() {
 Result r;
 r.get_data();
 r.get_student_data();
 r.get_sports_marks();
 r.get_academic_marks();

 r.calculate_result();
 cout << " Result: ";
 r.display_result();
 return 0;
}
```

* Output

```
enter name: Aarya
enter age: 16
enter roll no: 67
```

Sports marks : 98
enter academic marks : 100

Result :
Name : Aarya
Age : 16
Roll no. 67
Sports : 98
Academic : 100
Total marks : 198

26/9/25

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 {
public:
  int area (int side) {
    return side * side;
  }
  int area (int length, int breadth) {
    return length * breadth;
  }
};
int main() {
  Area a1;
  int side, length, breadth;
  cout << "enter side of classroom : ";
  cin >> side;
  cout << "enter length & breadth of laboratory :";
  in >> length >> breadth;
  cout << " area of laboratory : " << a1.area ( length,
  breadth) << endl;
  cout << " area of classroom: " << a1.area (side) << endl;
  return 0;
}
```

* Output

Enter side of classroom: 2
Enter length & breadth of laboratory: 18 2
Area of laboratory: 36
Area of classroom: 4

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

```
# Include <iostream>
using namespace std;
class sum_calculate {
    public:
    int sum (int a, int b, int c, int d, int e, int f, int g,
int h, int i, int j) {
    return a+b+c+d+e+f+g+h+i+j;
}
    float sum (float a, float b, float c, float d, float e){
    return a+b+c+d+e;
}};
int main(){
  Sum_calculate s;
  int s1 = s.num (1,1,1,1,1,1,1,1,1,1);
  float s2 = s.num (1.2f, 1.2f, 1.2f, 1.2f, 1.2f);
  cout<<"sum of 10 integer values :"<<s1<<endl;
  cout<<"sum of 5 float values :"<<s2<<endl;
  return 0;
}
```

* Output

sum of 10 integer values : 10
sum of 5 float values : 6

c) Write a program to implement Unary operator when used with the object so that the numeric data is negated.

```
#include<iostream>
using namespace std;
class Number{
 int a;
 public:
  void accept(){
   cout<<"enter a: ";
   cin>>a;
}
  void display(){
   cout<<"value of a is: "<<a;
}
  void operator-(){
   a=-a;
};
int main(){
 Number n1;
 n1.accept();
 -n1;
 n1.display();
 return 0;
}
```

~~WAP to impl~~ Output

```
enter a:22
value of a is:-22
```

n++ ++n

d) WAP to implement the Unary ++ operator (for pre increment & post increment) when used with object so that the numeric data member of class is incremented.

```
#include<iostream>
using namespace std;
class Number{
  int n=10;
  public:
  void display(){
    cout<<"value of n is :"<<n;
  }
  friend void operator(Number &x);
};
int main(){
  Number n1;
}
void operator++(Number &x){
  x.n = x.n++;
}

int main(){
  Number n1;
  n1++;
  n1.display();
}
```

* Output
value of n is : 10

24/9/25

Experiment - 8

Write a program to overload the '+' operator so that two strings can be concatenated.

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

Output

abcxyz

b) Write a program to create a base class Login having data members name & password. Declare accept function virtual. Derive emaillogin & membership-login classes from Login. Display emaillogin details & membership login details of the employee.

```
#include <iostream>
#include <string>
using namespace std;
class Login {
  protected:
    string name;
    string password;
  public:
    virtual void accept() {
      cout << "enter name : ";
      cin >> name;
      cout << "enter password ";
      cin >> password;
    }
    virtual void display() {
      cout << "Name : " << name << endl;
      cout << "Password : " << password << endl;
    }
};
class Emaillogin : public Login {
  private:
    string email_ID;
  public:
    void accept() {
      Login::accept();
      cout << "email ID : ";
      cin >> email_ID;
    }
    void display() {
      Login::display();
      cout << "email_ID" << end;
    }
};
```

```
class membership : public Login {
private:
    string membership_type;
    int ID;
public:
    void accept() {
        Login :: accept();
        cout << "enter membership type :";
        cin >> membership_type;
        cout << "enter membership ID";
        cin >> ID;
    }
    void display () {
        Login::display();
        cout << "membership type : " << membership_type <<
        endl;
        cout << " membership ID: " << ID << endl;
    }
};
int main() {
Email emp_email;
Membership emp_ms;
cout << " enter email details: ";
emp_email.accept();
cout << " enter membership details :";
emp_ms.accept();
cout << "email details :";
emp_email.display();
cout << " membership details ";
emp_ms.display();
return 0;
}
```

* Output

Enter email details:
Enter name : Aarya
-11- password : aarya
-11- email ID : aaryagaikwad1902@gmail.com

Enter membership details:
-11- name : Aarya
-11- password : aarya
-11- membership type : intermediate
-11- ID : 123

Email details:
Name : Aarya
Password : aarya
Email ID : aaryagaikwad1902@gmail.com

Membership details:
Name : Aarya
Password : aarya
Membership type : intermediate
-11- ID : 123

12/11

Experiment No. 9

Q1. Write a program to copy the contents of one file into another. Open "first.txt" in read (ios::in) mode & "second.txt" file in write (ios::out) mode. Copy the contents of 'first.txt' into 'second.txt'. Assume 'first.txt' is already created.

```
#include <fstream>
#include <iostream>
using namespace std;
int main(){
fstream file1, file2;
char ch;
file1.open("first.txt", ios::in);
file2.open("second.txt", ios::out);
while(file1.get(ch)){
    file2.put(ch);
}
cout<<" file copied successfully."<<endl;
file1.close();
file2.close();
return 0;
}
```

* Output

file copied successfully.

Q2. WAP to count digits & spaces using file handling

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

```
int main(){
ifstream 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.~~count~~close()
return 0;
}
```

* Output

Q3. WAP to count words using file handling.

```
#include<ifstream>
#include<fostream>
using namespace std;
```

```
int main(){
    fstream file;
    file.open("file.txt", ios::in);
    if(!file){
     cout<<"unable to open file";
     return 1;
    }
    char ch;
    int word_count=0;
    int prev_type=0;
    while(file.get(ch)){
     int current_type=((ch>='a'&&ch<='z')||
(ch>='A'&&ch<='Z')||(ch>='0'&&ch<='9'))?
     1:0;
     if(current_type==1 && prev_type==0){
        wordcount++;
     }
     prev_type=current_type;
    }
    cout<<"word count:"<<word_count<<endl;
    file.close();
    return 0;
}
```

* Output

12/11

Experiment 10

a) Write a program to find sum of array elements using function template. (eg, pass integer, float & double array of 10 elements)

```
#include <iostream>
using namespace std;
template <class t> t func(t a[], int n){
    t sum=0;
    int i;
    for(i=0, i<n; i++){
        sum = sum + a[i];
}
return sum;
}
int main(){
    int a1[3]={2,0,2};
    float a2[3]={1.2, 2.2, 2.4};
    double a3[3]={10.5, 20.5, 30.5};
    cout<<"sum of integer array is:"<<func(a1,3)
<<endl;
    cout<<"sum of float array is:"<<func(a2,3)
<<endl;
    cout<<"sum of double array is:"<<func(a3,3)
<<endl;
return 0;
}
```

* Output

sum of integer array is : 4
sum of float array is : 4.8

Sum of double array is : 61.5.

2) Write a C++ program of square function using template specialization :
:- calculate the square of integer no. & a string.
:- Write a specialized function for the square of a string.

```
# include <iostream>
using namespace std;
template <class T> T square (T x) {
    T result;
    result = x*x;
    return result;
}
template <> string square <string> (string ss) {
  return (ss+ss);
}
int main() {
  int i=2, ii;
  string www("Aarya");
  ii = square <int> (i);
  cout << i << ":" << ii << endl;
  cout << square <string> (www) << endl;
  return 0;
}
```

Output
2:4
AaryaAarya

Q) Write a C++ program to build simple calculator using class templated.

```
#include <iostream>
#include <cmath>
using namespace std;
template <class T>
class Calculator {
private:
  T num1, num2;
public:
  Calculator(T n1, T n2) {
    num1 = n1;
    num2 = n2;
  }
  T add() {
    return num1 + num2;
  }
  T sub() {
    return num1 - num2;
  }
  T mul() {
    return num1 * num2;
  }
  T divide() {
    if (num2 == 0) {
      cout << "error ! " << endl;
      return 0;
    } return num1 / num2;
  }
  void displayNumbers() {
    cout << "Numbers : " << num1 << num2 << endl;
  }
};
```

```
int main(){
  double a,b;
  int choice;
  cout<<"enter 2 numbers:";
   cin>>a>>b;
  calculator<double>calc(a,b);
  cout<<"Choose an operation";
  cout<<"1. Addition\n";
  cout<<"2. Substraction\n";
  cout<<"3. Multiplication\n";
  cout<<"4. Division\n";
  cout<<":
   cin>>choice;
  switch(choice){
  case 1:
     cout<<"Result:"<<calc.add(a,b)<<endl;
     break;
  case 2:
     cout<<"Result:"<<calc.sub(a,b)<<endl;
     break;
  case 3:
     cout<<"Result:"<<calc.mul(a,b)<<endl;
     break;
  case 4:
     cout<<"Result:"<<calc.div(a,b)<<endl;
     break;
  default:
     cout<<"invalid choice"<<endl;
}
return 0;}
```

* Output

enter two numbers : 2 4
Choose an operation : 1.
1. Addition
2. Subtraction
3. Multiplication
4. Division
Result : 6

choose an operation : 3.
1. Addition
2. Subtraction
3. Multiplication
4. Division
Result : 8

d) Write a c++ program to implement push & pop methods from stack using class template.

```
# include <iostream>
# include <vector>
using namespace std;
template <typename T>
class Stack {
private:
    vector<T> elements;
public:
    void push(T item) {
        elements.push_back(item);
        cout << item << "pushed to stack \n";
    }
    void pop() {
```

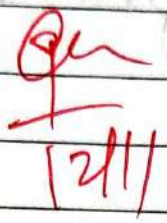
```
if (elements.empty()){
    cout << "stack is empty. Cannot pop.\n";
    return;
}
cout << elements.back() << " popped \n";
elements.pop_back();
}
void display(){
    cout << "stack elements: ";
    for (const T &item : et
    for (auto item : elements)
        cout << item << " ";
    cout << endl;
}};

int main(){
    stack<int> s;
    s.push(10);
    s.push(20);
    s.push(30);
    s.display();
    s.pop();
    s.display();
    return 0;
}
```

* Output

10 pushed to stack
20 pushed to stack
30 pushed to stack
Stack elements: 10 20 30

30 popped
Stack elements 10 20



Experiment 11

Q. Write a C++ program to implement generic vector
a) To modify the value of a given element.

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

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

* Output

0 10 20 30 40

after modification : 0 10 99 30 40

b) To multiply by a scalar value.

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

* Output
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;
}
```

* Output
(10, 20, 30, 40, 50)

Experiment : 12

Q. Write a C++ program using STL.

a) Implement stack

```
#include <iostream>
#include <stack>
using namespace std;
int main (){
stack <int> s;
s.push (10);
s.push (20);
s.push (30);
cout<<"Top element : "<< s.top ()<<endl;
s.pop ();
cout<<" Top element after pop: "<< s.top () << endl;
cout <<"stack size : "<< s. size () << endl;

if (s.empty ()){
    cout<<" stack is empty "<<endl;
}else{
    cout<<" stack is not empty ";
}
return 0;
}
```

* To Output

Top element : 30
Top element after pop : 20
stack size : 2
stack is not empty.

b) Implement queue

```
#include <iostream>
#include <queue>
using namespace std;
int main () {
    queue <int> q;
    q.push(10);
    q.pop(20);
    q.push(30);
    cout << "front elements: " << q.front() << endl;
    cout << "back elements: " << q.back() << endl;
q.pop();

    cout << "after pop operation : " << 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;
}
```

* Output

```
front element : 10
back element : 30
after pop operation :
front element : 20
queue size : 2
queue is not empty
```

4) Implement sorting & searching with user-defined records such as personal record (name, birth-date, telephone no), item record (item code, item name, quantity & cost).

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

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

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

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

    cout<<"sorted records by age: \n";
    for(auto& p: people){
```

```
    cout << p.name << "-" << p.age << "\n";
}

int searchAge = 28;
int foundIndex = -1;
for (int i=0; i<(int) people.size(); i++) {
    if (people[i].age == searchAge ? 1 : 0) {
        foundIndex = i;
        break;
    }
}
if (foundIndex != -1)
    cout << "person with age " << searchAge <<
"found" << people[foundIndex.name << "\n"
else
    cout << "person with age " << searchAge <<
" not found";
return 0;
}
```

* Output

Sorted records by Age:
Bob - 25
David - 28
Alice - 30
Charlie - 35
Person with age 28 found: David

12/11