Exp 1

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

```
#include <iostream>

using namespace std;

class student
{
  Private :
     String name;
     int rollNumber;

  Public :
   void accept ()
{
     cout << "Enter name:";
     getline (cin, name);

       cout << "Enter roll number: ";
       cin >> rollNumber;
    }
     void display ()
     {
        cout << "\nstudent Details: \n";
        cout << "Name:" << name << endl;
        cout << "Roll Number: " << RollNumber << endl;
      }
};
   int main () {
       Student s1
       s1. accept ();
       s1. display (); }
```

Output

```
Enter name: Soham
Enter roll no. 51

Student Details:
Name: Soham
Roll number: 51
```

① WAP to declare a class book, having data member as id, name, price. Accept data for 2 books & display data of book having greater price.

```
#include <iostream>

using namespace std;

class Book
{
  private
  int id;
  string name;
  float price;

Public:
 void input()
 {
   cout << "Enter book ID:";
   cin >> id;
   cin.ignore();
   cout << "Enter book name:";
    getline(cin, name);
    cout << Enter book price:";
    cin >> price;
 }
 void display() const
  {
    cout << "Book ID:" << id << endl;
    cout << "Bookname:" << name << endl;
    cout << "Book price:" << price << endl;
  }
```

```
    float getPrice() const
    {
        return price;
    }
};

int main()
{
    Book book1, book2;
    cout << "Enter details for book 1:\n";
    book1.input();
    cout << "\nEnter details for book 2:\n";
    book2.input();
    cout << "\nBook with higher price:\n";
    if (book1.getPrice() > book2.getPrice())
        book1.display();
    else if (book2.getPrice() > book1.getPrice())
        book2.display();
    else
    {
        cout << "Both books have the same price.\n";
        book1.display();
        book2.display();
    }
}
```

Output:

Enter details for Book 1 :
Enter book ID: 542
Enter book name : marvel
Enter book price : 699

Enter details for book 2:
Enter book ID : 566
Enter book name : DC
Enter book price : 799.

Book with higher price:
Book ID : 566
Book Name: DC
Book price : 799

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

```
#include <iostream>
using namespace std;
class Time
{
    int h, m, s;
    Public:
    void time ()
    {
        cout << "Enter";
        cin >> h >> m >> s;
    }
    void display ()
    {
        int sum = (h * 3600) + (m * 60) + s;
        cout << "Total time = " << sum;
    }
};
int main ()
{
    Time t;
    t.time ();
    t.display ();
}
```

Output:
Total time = 25627

Experiment-2.

Q- 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 {
public:
    String name;
    int population;

    void input() {
        cout << "Enter city name: ";
        cin >> name;
        cout << "Enter population: ";
        cin >> population;
    }
};

int main() {
    city c[5];
    int maxIndex = 0;

    for (int i=0; i< 5; i++) {
        cout << "In city " << i+1 << ": \n";
        c[i].input();
        if (c[i].population > c[maxIndex].
                                population) {
```

```
        maxIndex = i;
    }
}

cout << "In City with highest population: "
     << c[maxIndex].name
     << endl;

return 0;
}
```

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

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

class Account {
public:
    int accNo;
    float balance;

    void input() {
        cout << "Enter account number: ";
        cin >> accNo;
        cout << "Enter balance: ";
        cin >> balance;
    }
    void giveInterest() {
        if (balance >= 5000) {
            balance += balance * 0.10;
        }
    }

    void display() {
        cout << "Account No: " << accNo << " Balance: "
             << balance << endl;
```

```
};

int main() {
  Account a[10];

  for (int i = 0; i < 10; i++) {
    cout << "In Account " << (i+1) << ":\n";
    a[i].input();
    a[i].giveInterest();
  }

  cout << "\nIn Accounts with balance >= 5000 after interest :\n";
  for (int i = 0; i < 10; i++) {
    if (a[i]. >= 5000)
      a[i].display();
  }
  return 0;
}
```

Q3) WAP to declare a class 'staff' having data member as name & post. Accept this data for 5 staff and display name of staff whose post is 'HOD'.

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

class Staff {
    string name, post;

public:
    void input() {
        cout << "Enter Name: ";
        cin << "Enter Post: ";
        cin >> post;
    }
    void display If HOD() {
        if (post == "HOD" || post == "hod") {
            cout << "HOD Name " << name << endl;
        }
    }
};

int main()
{
    Staff s[5];
    for (int i=0; i<5; i++) {
        cout << "\n Enter details of staff " <<
            s[i].input();
    }
```

```
cout << "\n list of MODs : \n"
 for (int i=0; i<5; i++) {
    s[i].display IF MOD();
}
return 0;
}
```

29/7/23

Experiment-3

1) WAP. to declare a class 'book' containing data members as book-title, author-name & price. Accept & display the info one are object using a point* to that object

```
#include <iostream>
using namespace std;
class book
{
  Public :
  String book-title;
   int price;
   String author_name;

void accept ()
{
   cout << " enter book name: ";
   cin >> book-title;
   cout << "enter the name of author: ";
   cin >> author-name;
   cout << " price of the book: ";
   Cin >> price;
}
void display ()
{
   cout << " book name is : " << book-title
   cout << " \n author name : " << author_nam
   cout << " \n price is : " << price;
}
};
```

```
int main()
{
  book b1;
  book p;
  b1.accept();
  b1.display();

  return 0;
}
```

Output :-

```
Enter book name: Python
Enter name of author: Rush
Price of the book: 999
Book name is Python
Author name is Rush
Price is: 999
```

3) WAP to check a class 'Student' having data members roll_no & % using 'this' pointer & write member functions to accept & display the for one object of the class.

→
```
#include <iostream>
using namespace std;
class student
{
  int roll_no;
  float percentage;
  Public:

→ void accept()
  {
    cout << "Enter student roll_no: ";
    cin >> this -> roll_no;
    cout << "Enter student percentage:";
    cin >> this -> percentage;
  }
  void display()
  {
    cout << "number of student: " << roll_no;
    cout << "\n percentage is: " << percentage;
  }
};
  int main()
{
  student S1;
  Student *p;
  P = &S1;
```

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

return 0;
}
```

Output:

```
Enter the student roll_number : 39
Enter the student percentage : 93
Roll number of student is =39
Percentage is = % 93.
```

3) WAP in demonstrate use of nested class

->

```
#include <iostream>
using namespace std;
class student
{
public:
int num;
class operations
{
public:
int num
class operations
{
public:
int square(int n)
{
return n*n;
}
};
public:
void accept().
{
cout <<"Enter on number:";
cin>> num;
}
void display()
{
operations op;
cout <<" square of:"<<"<<op.square(num)<<
endl;
cout <<"cube of :" <<" = " <<op.cube(num)<<endl;
```

```
};
int main()
{
number n1;
n1.accept();
n1.display();
return 0;
}
```

output:

Enter a number: 2
Square of 2: 4
Cube of 2: 8

OOP ASSIGNMENT - 2

* Member functions - Passing Argument:
- Member function - A function that belongs to a class & operates on its objects & it can access the class's data members

* Call by value
- When the value is changed, it changes the value of that copy, the actual value remains the same.
- The actual value that is passed as argument is not changed after performing some operation on it.

- Code:

```
#include <iostream>
using namespace std;
void my_function (int x)
{
x=50
cout << " value of x from my_function : " << x <<
endl;
}
int main()
{
x=10
my_function (x)
cout << "value of x from Main function:
<< x << endl;
}
```

output: value of x from my function : 50
value of x from main function : 10

* Call by Reference :

→ In call by Reference the actual value that is passed as argument is changed after performing some operations on it.
So when the value is changed the value of actual variables

Code:
```
#include <iostream>
using namespace std;
void my-function (int &x) {
x= 50;
cout << " value of x from my-function:"
  << x<<endl;
}
int main () {
int x =10 ;
my-function (x)
cout << "value of x from my-function
 : " << x << endl;
}
int main () {
 int x =10
 my-function (x);
 cout << " value of x from main function:
                          <<x<<endl;
}
```

Output : value of x from my-function: 50
value of x from main function : 50

* Difference bet" call by Ref & call by Value

Call by Reference	Call by value
• Requires more memory	• Requires less memory
• Takes more time	• Takes less time
• Makes copy of the actual parameter.	• Address of actual Parameter passes to fn.
• changes made inside, do not reflect in original value.	• changes made inside, to reflection original value

* Default Argument :

• we Assign default value to these argument at time of function declaration & the default values assign to the function parameter called default argument.

- Syntax : return_type function_name (Parameter1 = default_value1, parameter2 = default_value 2,

- Code :
```
# include <iostream>
using namespace std;
int sum (int x, int y, int z=4, int w=0),
{
 return (x+y+z+w);
}
int main()
{
 cout << sum (10,15)<< endl;
 cout << sum (10,15,25) << endl;
 cout << sum (10,15,25,30) << endl;
}
```

- Output : 19
50
80

* Object as function Argument
- The objed of a class can be passed as argument to the member function. So well as non-member function.

This can be done in 2ways.
. A copy of entire object is passed to the functio
. Only the address of objed is transferred to the function

* Passing object by Value :
- Syntax : return_type function name (class name obj);

* Inline functions;
it is a function that is expanded in line when it is invoked.
. The compiler replaces the function call with corresponding function code.

- Syntax : inline return-type function_name (Parameter list) { // function body }

- Code : # include <iostream>
using namespace std;
inline int square (int x) {
return x* x;
}

int main() {
cout << " Square of 5 : " << square(5) << endl;
cout << " Square of 8 : " << square (8) << endl;
return 0;
}

Output : Square of 5: 25
Square of 8: 64

* Nesting of Member function
It is defined as calling one member function of a class from another member function of same class.

- Syntax : class classname {
// data members.
void private function ();
Public:
void outer function ();
};
void class Name:: Outer function () {
private function ();
}.

- Code : # include <iostream>
using namespace std;
class Sample {
int data;
void read (){
cout << "Enter Value: ";
cin >> data;
}
Public :

```
void display () {
 read ();
 cout << " you Entered : " << data;
    }
 };

 int main () {
 sample s;
 s.display ();
 return 0;
}
```

output; Enter value : 42
you entered 42.

* Returning values from function.
- A function means sending data back to the caller using the return statement. The value can be a basic type a reference or even a object

Syntax :

→ returning By Value:

```
return. type function-name (parameters){
    // statement:
    return value;
  }
```

→ returning By Reference:

```
return. type of function-name (parameters){
   // statements
   return variable;
  }
```

→ returning an Object

```
class Name function_name () {
   class Name obj;
   return obj;
}
```

code → Returning By Value

Output →
Square of 6 = 36

```
# include <iostream>
 using namespace std;
 int Square (int n) {
    return n * n
 }
  int main () {
   int num = 6;
   int result = square (num);
   cout << " Square of " << num << " = " << result;
   return 0;
   }
```

code → Returning pass :

```
    # include <iostream>
     using namespace std;
     int & larger (int &a, int &b) {
       return (a>b) ? a : b;
     }
        int main () {
        int x=10, y = 25
        larger (x,y) = 100;
        cout << " x = " << x << " y = " << y;
        return 0;
        }
```

- Returning an object.

```
# include <iostream>
using namespace std;
class box {
Public:
 int length;
 Box (int l=0) : length (l) { }
};
    Box create Box (int size) {
        Box b (size);
        return b;
}
   int main() {
    Box b, = create Box (50)
    cout << " Box length = " << b1. length;
    return 0;
   }
```

* Friend function:

it is a special function that is not a number of class but has permission to access its private & protected members

- it is declared inside the class with the keyword friend.
- friendship is not inherited. & not transitive.
- It is useful for operator overloading.
- work on multiple classes.

- Syntax ->

```
class Name {
// data members
friend return_type function_name
                    (class name obj);
};
```

Code ->

```
# include <iostream>
using namespace std;
class sample {
int data
Public:
Sample (int x) : data (x) { }
Sample friend void display (Sample s);
};
void display (Sample s) {
    cout << " Private data = " << s.data<<
                                 endl;
}
int main() {
    Sample ob, (4);
    display ob);
    return 0;
}
```

Output -> Private data = 4;

* pointer to object :
→ when pointer variable is pointing to memory Address of object then such concepts is called as pointer to object..

→ object access the class member using ( ) operator.
→ In Concept of pointer to object can be access class member using an arrow ( ) operator.

- Syntax : int main ( ) {
  class Name * pointer_name;
  pointer_name = & object_name;
  Pointer_name → member function ( );

- Code : # include <iostream>
  using namespace std;
  class Item {
  item price;
  Public:
  void set price (int p) { price = p; }
  void showprice ( ) { cout << "price" << Price << endl; }
  };
  int main ( ) {
  Item obj;
  Item * ptr;
  ptr = & obj;
  ptr → set price (150);
  ptr → show price ( );
  return 0;
  }
  Output : Price = 150

* "This" pointer :
- Every object in C++ has access to its own address through an important pointer is called 'this pointer'.

→ 'This' pointer holds adres of current object that is 'This' pointer too current object that is this
- It is used to differentiate bet" data member & parameter
- friend fun do not this pointer because friends are not member fun.

⇒ Code -> # include <iostream>
  using namespace std;
  class student
  {
  int roll;
  string name;
  Public
  void accept ( );
  {
  cout << " Enter Rollno: ";
  cin >> This -> roll;
  cout << " Enter Name of student";
  cin >> this -> name;
  }
  void disp ( )
  {
  this -> accept ( );

```
    cout<< "Roll no = " << roll;
    cout << "Name = " << name;
  }
}
int main()
{
  student s1;
  s1. display()
  return 0;
}
```

→ Output

```
Enter Roll no = 24
Enter Name of student = Soham
Roll no = 24
Name : Soham
```

Nested / Inner class

→ A class declared inside another class. It helps group related classes together & limits the scope of the inner class to the enclosing class

Syntax →

```
class Outer {
Public:
  class Inner {
    Public
      void show();
  };
};
```

code: →

```
# include <iostream>
using namespace std;
class Outer {
Public:   void display_outer() {
            cout << "This is Outer class";
          }
  class Inner {
    Public:
    void display_Inner() {
      cout << "This is Inner class";
    }
  };
};
int main() {
  Outer o;
  o.displayOuter();
```

```
Outer : Inner i;
  i.display Inner();
  return0;
}
```

Output → This is Outer class
This is Inner class.

Q/
19/9

Exp-9

Q- Create 2 classes, ClassA and class B, each with a private integer write a friend function Sum() that can access Private data from both classes and return Sum.

```
# include <iostream>
using name space std;

  class B;

Class A {
private:
   int x;

Public:
     A(int val) {
        x = val;
     }
     friend int Sum(A, B);
};
Class B {
private:
   int y;
Public:
     B(int val) {
        y = va;
     }
     friend Int Sum(A, B);
};
```

```
int Sum (A a, B b) {
    return a.x + b.y;
}
int main ()
    A obj1 (5);
    B obj2 (7);

    cout << "Sum: " << sum(obj1, obj2) << endl;

    return 0;
}
```

→ Output
Sum: 12

Q: WAP with a class Number that contains a private integer. Use a friend function swapNumbers (Number, Number) to swap the private values of two Number objects.

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

class Number {
private
    int value;
Public:
    Number (int val) {
        value = val;
    }
    void show () {
        cout << "value : " << value << endl;
    }

    friend void swapNumbers
                (Number&, Number& );
};
void swapNumbers (Number& n1, Number& n2){
    int temp = n1.value;
    n1. Value = n2.value;
    n2.value = temp;
}
int main () {
    Number a (10);
    Number b (20);
cout << "Before swap:" << endl;
```

```
a.show();
b.show();

SwapNumbers (a,b);

cout << "After swap: " << endl;
    a.show();
    b.show();

    return 0;
}
```

Output:

```
Before swap
value : 10
value : 20
After Swap
value : 20
value : 10
```

Q3) Define 2 classes Box and Cube, each having a private volume. Write a friend function findGreater (Box, Cube) that determines which object has a larger volume.

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

class Cube;

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

public:
    Cube (int v) {
        volume = v;
    }
    friend void findGreater (Box, Cube);
};
void findGreater (Box b, Cube c) {
    if (b.volume > c.volume) {
```

```
    cout << "Box has greater volume: " <<
} else if (volume > b.volume) {cout<<
cout << "Cube has greater volume: "
    << volume << endl;

} else {
    cout << "Both have equal volume: "
        << b.volume << endl;
}
}

int main () {
Box box1 (100);
Cube box2 (80);

find Greater (box1, cube1);
    return 0;
}
```

Output
Box has greater volume : 100

04) Create a class Complex with real and imaginary parts as private member. Use a friend fun to add two complex numbers and return the result as a new Complex object.

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

class Complex {
private:
    int real;
    int img;

Public:
    Complex (int r=0; int i=0) {
        real = r;
        imag = i;
    }
    void show () {
        cout << real << " + " << imag
                    << "i" << 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 = ad(a,b).

    cout << "Sum of complex no:";
    c.show();

    return 0;
}
```

Output
Sum of complex number : 6+8i

Q5) Create a class Student with private data members : name and 3 subject mark. Write a friend function calculate Average (Student) that calculate and displays the average marks.

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

class student {
private:
    string name;
    int mark1 ; mark2, mark3;

public:
    Student (string n, int m1, int m2, int m3)
        name = n;
        mark1 = m1,
        mark2 = m2,
        mark3 = m3,
    }
    friend void calculate Average (student);
};

void calculate Average (Student s) {
    float avg = (s.mark1 + s.mark2 + s.mark3
    cout << "Student name " << s.name << endl;
    cout << "Average marks " << avg << endl;
}
int main() {
    student s1 ("Ani", 85, 90, 95),
    Calculate Average (s1),
    return 0;
}
```

→ Output:
Student Name: Sotara
Average Marks : 90

Q6) Create three classes: Alpha, Beta, and Gamma, each with a private data member, while a single friend function that can access all three and print their sum.

```
#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 totalSum (Alpha, Beta, Gamma);
};

class Gama {
  private:
      int c;

Public:
    Gamma (int val) {
        c = val;
    }

    friend void totalSum (Alpha, Beta, Gamma);
};

void total Sum (Alpha x, Beta y, Gamma z)
    { int Sum = x.a + y.b + z.c;
    cout << "Sum of all values :" <<sum
        << endl;
    }
int main() {
    Alpha obj 1 (5);
    Beta obj 2 (20);
    Gamma obj 3 (30);

    total Sum (obj 1, ob 2, obj 3);
    return 0;
}
```

-> Output:
Sum of all values : 55

Q) Create a class point with private members x and y. Write a friend function that calculate and returns the distance between 2 point objects.

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

class point {
private:
    int x, y;

public:
    Point (int xval, int yval) {
        x = xval;
        y = yval;
    }
    friend double calculateDistance (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;
}
```

Output:
Distance between points

Q3) Create two classes : BankAccount and Audit. BankAccount holds private balance information. Write a friend function in Audit that access and prints balance information for auditing.

```
#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 acc) {
        auditBalance (acc, *this);
    }
```

Investment $

→ Angels take higher risks than VCs cause they invest in early startup which has higher rate of loss.

→ 90 % of startups fail ~~due to~~ in their first 5 years.

→ After the angle deal ~~3 to~~ 2 - 3 moths are required to complete the funding process, So if startups require any funding, they have to start dealing with the angle aprox. one ~~month~~ year prior. ~~of~~.

33
55
165 Lakh

for Stakh.

1.65

3%

friend void auditBalance (BankAccount, Audit)
{};

void auditBalance (BankAccount acc, Audit) {
cout << "Balance: " << acc.Balance << endl; }

int main () {
BankAccount myAcc (50000);
Audit auditor;

auditor.StartAudit (myAcc);

return 0;
}

→ Output
Balance : Rs 50000

Eg) WAP to swap 2 numbers from same class using object as function argument. Write swap function as member function.

#include <iostream>
using namespace std;

class Number {
int num;
Public:
Number (int n) { num = n; }
void swap (Number &n) {

```
        int t = num;
        num = n.num;
        n.num = t;
    }
    void show() { cout << name
};

int main() {
    Number a(10), b(20);
    a.show(); b.show(); cout
    a.Swap(b);
    a.show(); b.show();
}
```

→ Output:

10 20
20 10

WAP to swap 2 numbers from different class using friend function

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

class B;
class A {
    int numA;
public:
    A(int n) { numA = n; }
    friend void swapNumbers(A &x, B &y);
    void show() { cout << numA << " "; }
};
class B {
    int numB;
public:
    B(int n) { numB = n; }
    friend void swapNumbers(A &x, B &y);
    void show() { cout << numB << " "; }
};
void swapNumbers(A &x, B &y) {
    int temp = x.numA;
    x.numA = y.numB;
    y.numB = temp;
}
int main() {
    A obj1(10);
    B obj2(20);
```

obj 1. show(); obj 2. show(); cout
Swap Numbers (obj 1, obj 2);
obj 1. show(); obj 2. show();
}

→ Output

10 20
20 10

Q12) WAP to create 2 classes Result and Result2 which stores the marks of students. Read the value of marks for both the class objects and compute average of the 2 given marks

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

class Result2;

class Result1 {
    float marks;
Public:
    void read() {
    cout<<" Enter marks for Result1
    cin >> marks;
    }
    friend void average (Result1, Re
};

class Result2 {
    float marks;
Public:
    void read() {
    cout<< Enter mark for Result2
    cin >> marks;
    }
    friend void average (Result1, Result2);
};

void average (Result1, Result2, r2){
float avg = (r1.marks + r2.marks)/2;
cout << " Average marks = " << avg;
}

int main() {
    Result obj1;
    Result obj2;

    obj1.read();
    obj2.read();

    average (obj1, obj2)
    return 0;
}
```

Output :

Enter marks for Result : 100
Enter marks for Result : 99
Average marks = 99.5

Q5) WAP to find the greatest among 2 numbers from 2 different classes using friend function

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

class B;

class A {
    int numA;

Public:
    void read () {
        cout << "Enter number for class A: ";
        cin >> numA;
    }
    friend void find Greatest (A,B);
};
class B {
    int numB;
Public:
    void read () {
        cout << "Enter number for class B: ";
        cin >> numB;
    }
    friend void find Greatest (A,B);
};
```

```
void find Greatest (A x, B y) {
    if (x.numA > y.numB)
        cout << "Greatest : " << x.numA << endl;
    else if (y.numB > x.numA)
        cout << "Greatest : " << y.numB << endl;
    else
        cout << "Equal " << endl;
}
int main () {
    A obj1;
    B obj2;

    obj1.Read();
    obj2.read();

    find Greatest (obj1, obj2);
    return 0;
}
```

Output:

```
Enter number for class A: 17
Enter number for class B: 23
Greatest : 23
```

19/11

Exp - 5

Q1) WAP to find the Sum of numbers between 1 to n using a constructor where the value.

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

class number {
    int num;

Public:
    number () {
        cout << " Enter a number
        cin >> num;

        int sum=0;
        for (int i=1 ; i< = num ; i++)
            Sum = Sum + i;
        }

        cout << " Sum of number upto <<n
            " is : " << Sum << endl;
        }
    };
int main() {
    number n;

    return 0;
}
```

Output :

```
Enter a number:
The Sum of num
upto 3 is 6.
```

Q2) WAP to declare a class 'student' having data members as name and percentage. write a constructor to include these data members. Accepted data for one student.

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

class Student {
    string name;
    float per;
    Public:
    student (string n, float p) {
        name = n;
        per = p;
    }

void display () {
    cout << "Name : " << name << endl;
    cout << "Percentage :" << per << endl;
}
};
int main() {
    Student s ("Soham", 93.2);
    S.display ();

    return 0;
}
```

Output :

```
Name Soham
Percentage : 93.2
```

Q3) Define a class 'College' [illegible] as roll no, name, course, WAP [illegible] constructor (with default values) [illegible] Engineering' for Cources. Accept [illegible] for 2 objects of class and [illegible]

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

class college {
    int roll;
    string name;
    string course;
    Public:
    College (int r = 00; string n = ;
             string c = "Computer ;
        { roll = r;
          name = n;
          course = c;
        }
    void display () {
        cout << "Roll no : " << roll <<
        cout << "Name : " << name <<
        cout << "course" << Co
};
int main () {
    College s1 (37, "Anushree");
    College s2 (18, "Sahil");

    s1. display ();
    s2. display ();
```

```
    return 0;
}

Output :
Roll number : 1
Name : Soham
Course : Computer Engineering
Roll number 18
Name : Soham 2
Course : Computer Engineering

WAP to demonstrate Constructor
loading
# include <iostream
using namespace std;

class student {
    int Roll;
    String name;
    Public:

    student () {
        name = "Unknown";
        roll = 0;
    }
student (string , int r) {
    name = n;
    roll = r
void display () {
    cout << "Name " << name << endl;
    cout << "Roll Number " << roll << end
}
```

```
int main () {
    student s1;
    student s2 ("[illegible]");
    student s3 ("Sahil");

    s1. display ();
    s2. display ();
    s3 display ();
    return 0;
}
```

Output:

```
Name : Unknown
Roll number : 0.
Name : soham
Roll no: 0
Name : Sahil
Roll no : 18
```

14/11

Exp - 6

Write a program to implement multilevel inheritance. Assume suitable data

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

class department {
protected:
    string dname;
};
class student : protected department {
protected:
    string sname
    int roll;
};
class marks : protected student {
    int m1, m2, percentage;
public:
    void accept () {
        cout << "Enter department : "<<endl;
        cin >> dname;
        cout << "Enter name : "<<endl;
        cin >> name;
        cout << "Enter marks 1 : "<<endl;
        cin >> m1;
        cout << "Enter marks 2 : "<<...
        cin >> m2;
    }
    void calculate () {
        int per = (m1+m2)/2;
```

```
        cout << "Department : " << dname <<
        cout << "Name : " << Sname <<
        cout << "Percentage : " << Per
    }
};
int main () {
    marks m;
    m.accept ();
    m.calculate ();
    return 0;
}
```

-> Angels t
in early

-> 90 %
5

-> After t
required
So
to
~~mail~~

-> output:

```
Enter department :
Soham
Enter marks :
81
Department : CSE
Name Soham
Percentage : 88.
```

WAP to implement multiple inheritance
Assume Suitable data

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

class department {
protected:
    string dname;
};
class student {
protected:
    string sname;
    int roll;
};
class marks : protected department, protected student {
    int m1, m2, percentage;
public:
    void accept () {
        cout << "Enter department" << endl;
        cin >> dname;
        cout << "Enter name : " << endl;
        cin >> Sname;
        cout << "Enter name marks" <<
        endl;
        cin >> m1;
        cout << "Enter marks : " <<
        endl;
        cin >> m2;
```

```
cout << "Department: " << dname <<
cout << "Name: " << Sname <<
cout << Percentage: " << Per <<
}
int main () {
    marks m;
    m.accept ();
    m.calculate ();
    return 0;
}
```

-> Output:

Enter department:
Soham
Enter marks:
91
Department: CSE
Name Soham
Percentage: 88.

WAP to implement multiple inheritance.
Assume Suitable data

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

class department {
protected:
    string dname;
};
class student {
protected:
    string sname;
    int roll;
};
class marks: protected department, protected student {
    int m1, m2, percentage;
public:
    void accept () {
        cout << "Enter department:";
        cin >> dname;
        cout << "Enter name: " << endl;
        cin >> Sname;
        cout << "Enter marks1:" << endl;
        cin >> m1;
        cout << "Enter marks2: " <<
        endl;
        cin >> m2;
```

→ Angels take
in early star

→ 90 % of
5 yea

→ After the
required
So if
to star
~~month~~ yea

```
    }
    void calculate () {
        int per = (m1+m2)/2;
        cout << "Department : " <<
        cout << "Name : " << sname
        cout << "Percentage : " <<
    }
};
int main () {
    marks m;
    m.accept ();
    m.calculate ();
}
```

Q:) WAP to implement hierarchical inh

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

class Person {
protected :
string name;
int age;
public :
void acceptPer () {
    cout << "Enter name : " << endl;
    cin >> name;
    cout << "Enter age : " << endl;
    cin >> age
};
```

```
class Student : public Person {
    int roll;
    float per;
    Public :
    void accept stud () {
        cout << "Enter roll number" endl;
        cin >> roll;
        cout << "Enter percentage " <<endl

        cin >> per;
    }

void display stud () {
    cout << "Name : " << name << endl;
    cout << "Age : " << age << endl;
    cout << "Roll no : " << roll << endl;
    cout << "Percentage : " << percentage;
}
};

class staff : public person {
    int emp-id;
    string subject;
    Public :
    void accept staff () {
    cout << "Enter employee ID : " <<endl;
    cin >> emp-id;

    cout << "Enter Subject : " << "endl;
    cin >> subject;
}
```

```
void display staff () {
    cout << " Name : " << name
    cout << Age : " << age
    cout << " Emp ID : "
    cout << " Subject taught
  }
};

int main () {
    Student s;
    Staff t;

    s. accept Per ();
    s. accept stud ();
    t. accept Per ();
    t. accept staff ();

    s. display stud ();
    t. display staff ();

    return 0;
}
```

Write a program to implement hybrid inheritance

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

class College {
        protected:
        string cname;
};
  class Employee : Protected College {
            Protected:
            string emp_name;
            int id;
};

class staff : public Employee {
            string name;
            int dept Id;
            public:
        void accept Emp() {
            cout << "Enter clg name" << endl;
            cin >> cname;
            cout << " Enter Emp name " << endl;
            cin >> emp_Name;
            cout << " Enter ID : " << endl;
            cin >> id;
            cout << Enter staff name : " << endl;
            cin >> name;
            cout << "Enter dept id :- " << endl;
            cin >> dept Id;
        }
```

```
void display Emp() {
    cout << " College : " << cname ;
    cout << " Emp name : " << emp_n;
    cout << " ID : " << id << endl;
    cout << " Staff Name : " << s
    cout << " Department ID : " <<
}
};
class Student : protected College {
        string stu_name;
        int roll;
        public:
        void accept Stud() {
          cout<< "Enter college na
          cin >> cname;
          cout<< "Enter name:
          cin >> stu_name;
          cout << "Enter roll numb
          cin >> roll;
        }
    void display Stud() {
      cout << " College : "<< cname<<e
      cout << " Student name : "<< stu

      cout<< " Roll number : " roll <<
    }
int main() {
  Staff staff1;
  staff1.acceptEmp();
  staff1.display Emp();
```

```
  Student stu1;
  stu1.accept stud();
  stu1.display stud();

  return 0;
}
```

WAP to demonstrate virtual base class Name Suitable data with figures.

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

class College Student {
 protected:
     int student_id;
     string ccode;
 Public:
     void accept() {
       cout<< "Enter Student ID: ";
       cin >> student_id;
       cout << "Enter College Code:";
       cin >> ccode;
     }
     void display() {
       cout << "Student ID: "<< student_id<<
       cout << "college Code: "<<ccode<<endl;
     }
};
class Test : virtual public College student {
 protected:
```

```
    float percentage;
public:
    void accept() {
        College Student :: accept();
        cout << "Enter Test percentage";
        cin >> percentage;
    }
    void display() {
        College Student :: display();
        cout << "Test Percentage"
             << percentage << endl;
    }
};

class Sports : virtual public College {
protected:
    char grade;
public:
    void accept() {
        cout << "Enter Sports Grade";
        cin >> grade;
    }
    void display() {
        cout << "Sports Grade:" << grade << endl;
    }
};
```

```
class Result : public Test, public Sports {
    float Tot_marks;
public:
    void accept() {
        College Student :: accept();
        cout << "Enter Test percentage";
        cin >> percentage;
        cout << "Enter Sports Grade";
        cin >> grade;
        cout << "Enter Total marks";
        cin >> Tot_marks;
    }
    void display() {
        College Student :: display();
        cout << "Test Percentage" << percentage << endl;
        cout << "Sports Grade:" << grade << endl;
        cout << "Total marks:" << Tot_marks << endl;
    }
};
int main() {
    Result r;
    cout << "\n ---- Enter Student Details ----" << endl;
    r.accept();
    cout << "\n ---- Student Result ----" << endl;
    r.display();
    return 0;
}
```

Exp-7

1) WAP using function overloading to find the area of a laboratory and class room

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

class Area {
public:
    float calculate (float length, float breadth)
    { return length * breadth;
    }
    float calculate (float side) {
        return side * side;
    }
};

int main() {
    Area a;
    cout << "Area of labratory: " << a.calculate(10.5, 20.5) << endl;
    cout << "Area of square: " << a.calculate(5.5) << endl;
```

2) WAP using function overloading to calculate the Sum of 5 float values and sum of 10 integer values.

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

class Sum {
Public:
    int total (int a[], int n) {
        int s=0;
        for (int i=0; i<n; i++)
            s += a[i];
        return s;
    }
    float total (float a[], int n) {
        float s=0;
        for (int i=0; i<n; i++)
            s += a[i];
        return s;
    }
};

int main() {
    Sum s;
    int marks[10] = {45, 56, 67, 78, 89, 50, 76, 88, 92, 85};
    float grades[5] = {9.2, 8.7, 9.5, 8.9, 9.0};
    cout << "Sum of 10 student marks: " <<
        s.total(marks, 10) << endl;
    cout << "Sum of 5 students grade points: "
        << s.total(grades, 5) << endl;
```

```
    return 0;
}
```

Q3) WAP to demonstrate the C... and implement Unary operator (uber... with the object so that the ... data member of the classes ...

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

class Teacher {
    int experience;
Public:
    Teacher (int e) {
        experience = e;
    }
    void display () {
        cout << " Experience " << ex
                " year " << e
    }
    void operator -() {
        experience = - experience;
    }
};
int main () {
    Teacher t1 (10);
    t1.display();
    -t1;
    cout << "After negation:";
    t1.display ();
    return 0;
```

WAP to implement the Unary ++ operator when used with the object so that the numeric data member of the class is Incremented

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

class Student {
    int count;
Public:
    Student (int c=0) {
        count = c;
    }
    void operator ++ () {
        ++ count;
    }
    void operator ++ (int) {
        count ++;
    }
    void display {
        cout << "Student count: " << count << endl;
    }
};
int main () {
    Student S1 (50);
    cout << " Before increment : " << endl;
    S1.display ();
    ++ S1;
    cout << " After pre-increment: " << endl;
    S1.display ();
```

→ After the
required
So if
to str
~~wont~~ ye

```
endl;
<< "After post-incr..."
s1. display ();
return 0;
}
```

14/11

Exp-8

WAP to overload the '+' operator so that 2 strings can be concentinated.

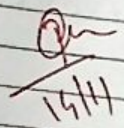
```
# include <iostreams
# include < string>
using namespace std;
class Combine {
    string str;

public : Combine (string s=" ")
        str = s;
    }

combine operator + (commbine obj) {
    return combine (str+ obj.str);
    }
void display () {
    cout << str << endl;
    }
};

int main () {
    combine s1 ("xyz"), s2 ("ipqr"), s3;
    s3 = s1+s2;
    cout << "Concatenated String :";
    s3. display ();
}
```

Q2) Write a program to create a base class I Login having data members name and password. Declare accept() function virtual. Derive Email Login and Membership class from I Login. Display accept() virtual. Derive Email Login and Mem

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

class I login {
protected:
    string name, password;
Public:
    virtual void accept () {
        cout << "Enter name: ";
        cin >> name;
        cout << " Enter passw
        cin >> password;
    }
Virtual void display () {
    cout << " Name : " << Name
    cout << "password : " << passw
}
};
```

```
class Email Login : public I login {
    string email;

Public :
    void accept () override {
        cout << " Enter Email ID : ";
        cin >> email;
        Login :: accept ();
    }
    void display () override {
        cout << "\n --- Email Login in Detail's
                                    << endl;
        cout << " Email ID; " << email
        I login :: display ();
    }
};
class Membership login : public I login {
string member ID;

Public:
    void accept () override {
        cout << " Enter Membership ship ID
        cin >> member ID;
        I login :: accept ();
    }
    void display () override {
        cout << "\n --- Membership login in Detail
                                    --- " << endl;
        cout << " Membership ID : " << member ID
        I Login :: display ();
    }
```

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



Exp - 9

Write a program to copy the content one file into another. Open "first.txt in read mode and "second.text". Assume 'first' text is already created.

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

int main () {
    ifstream infile ("first.text");
    ofstream outfile ("second.text");

    if (!infile) {
        cout<< " Error opening first.text "<<endl;
        return 1;
    }
    char ch;
    while (infile.get (ch)) {
        outfile.put (ch);
    }
    cout << "file copied successfully" << endl;
    infile.close ();
    outfile.close ();
    return 0;
}
```

Q2) WAP to Count Digits and Spaces file handling

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

int main () {
    ifstream file ('first. text');
    if (: file) {
    cout << " Error opening file "<<
    return 1;
    }
char ch;
int digits =0 ; spaces =0;

while (file.get (ch)) {
    if (is digit (ch))
        digits ++
    else if (isspace (ch))
        spaces ++;
    }
cout << " Digits : " << digits <<
cout << " Spaces : " spaces <<

    file . close ();
    return 0;
}
```

WAP to Count words using file Handling.

```
# include <iostream>
# include < fstream>
# Include < string >
using name space std ,

int main () {
    ifstream file (" First . txt ");
    if (! file) {
        cout << " Error " << endl;
        return 1;
    }

String word;
int count =0;

while (file >> word)
    count ++ ;

cout << " Total words . " << count << endl;
file. close ();
return 0;
}
```

Q4) Write a C++ program to count ... of a give word using file ...

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

int main () {
    ifstream file ("first.text");
    if (!file) {
        cout << "Error " << endl;
        return 1;
    }
    string word, target;
    int count = 0;

    cout << "Enter word to count: "
    cin >> target;

    while (file >> word) {
        if (word = target)
            count ++;
    }
    cout << " occurence of [" " <<
    file.close ();
}
```

Exp-10

```
# include <iostream>
using namespace std;
template < class T>
T Summary (T arr[], int n)
{
    T Sum = 0;
    for (int i = 0; i<n; i++)
        Sum += arr[i];
    return Sum;
}
Int main()
{
    int int arr[5] = {1,2,3,4,5};
    float flo - arr[5] = {1.1, 2.2, 3.3, 5.5, 5
    double dec_arr [5] = {0.5, 1.5, 2.5, 3.5,

cout << " Sum of integer array " <<
        Sum Array (int arr, 5) <<
            endl;

cout << " Sum of float array: " << SUM A
        rray (flo_arr, 5) << endl;

cout << " Sum of double array: "
        Sum Array (dec_arr, 5) <<
            endl;
    return 0;
}
```

→ Angels take in early ...

→ 30 % of 5 ...

→ After the required So if to star ... yea

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

template <typename T>
T square (T num)
{
    return num * num;
}
template <>
string square <string> (string str)
{
    return str + str ;
}
int main ()
{
int num = 5;
double dec_num = 2.5;
string str = "Apple";

cout << " square of integer :" << square
    (num) << endl;
cout << " square of double :" << square
    (dec_num) << endl;
cout << " square of string :" <<
    square (str) << endl;

return 0;
}
```

Angels take h
in early startup

90% of
5 years

After the angl
required to
So if
to start
~~next~~ year

```
    }
    void rem()
    {
    cout << " Reminder: " << x % y << endl;
    }
    void power()
    {
    cout << " x raised to y is: " << pow(x,y)
        << endl;
    }
    void min_num()
    {
    cout << " min of numbers : " << fmin(x,y)
        endl;
    }
    cout << " max of numbers : " << fmax(x,y)
        << endl;
    }
    void sin_x()
    {
    cout << " sin of x : " << sin(x) <<
        endl;
    }
    void cos_y()
    {
    cout << " cos of y : " << cos(y)
    }
};
int main()
```

```
calc <int, int> n(100, 200);
n.sum();
n.diff();
n.pro();
n.quo();
n.rem();
n.power();
n.min_num();
n.max_num();
n.sin_x();
n.cos_y();
}
```

12/11

Exp- 11

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

int main()
{
    vector <int> v = {1,2,3,4,5,6,7,8,9};

    cout << "Initial vector : " << endl;
    for (int i=0 ; i<9; i++)
    {
    cout << v[i] << " " << endl;
    }
    cout << "multiply by 5" <<
        endl;

    for (int i=0; i<9; i++)
    {
        v[i] = v[i] * 5;
    }
    cout << "New vector : " << endl;
    for (int i=0; i<9; i++)
    {
    cout << v[i] << " " << endl;
    }
    return 0;
}
```

Exp-12

Q:) Implement stack using ~~FS~~ STL

```
# include <iostream>
# include < stack>

using namespace std;
  int main ()
{
    stack < int>s;
    int ch , x;

do {
cout<< " \n 1. push \n2 .pop\n 3. top\n4
      Size\n 5. Exit \n";
    cout<< " enter choice:";
      cin >> ch.

  Switch (ch)
{
cas 1 : cout << " enter value : ";
        cin >> x;
      s. push (x);
      break ;
  Case 2;
   if (! S.empty ())
   {
        S. pop ();
   cout << " popped ;;
   else
   {
      cout << " Stack is empty ;")
   }
   break;
   case 3:
      y (! s. empty ())
{
  cout << " Top = " << S.top() <<end;
}
  else
{
  cout << " stack is empty \n";
}
break;
Case 4:
cout << "size : " << s.size () << endl
break;
case 5:
  Cout << " exit";
    break;

default :
   cout << " Invalid";
}}
while (ch! = 5);

    return 0;
}
```

→ Angels t
in early

→ go "4

→ After
requir
So
to

Q:)

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

int main()
{
  queue <int> q;
int ch, x;
do {
cout << "\n1.enqueue \n2. dequeue\n3
      front\n" << "4. Size 5. exit \n";
cout << "enter choice :";
  cin >> ch;

switch (ch)
{
case 1:
  cout<< " enter value :";
  cin >> x;
  q.push()
  break;
case 2:
  if (!q.empty())
  {
     q.pop();
  cout << " dequeue :"
  }
  else
  {
  cout << " Queue empty";
}
break;
Case 3:
if (!q.empty())
{
cout<< " front :" << q.front();
}
else
{
cout<< " queue is empty
}
break;

case 4:
{
 cout << "size :" << q.size()
       break;
    e }
    case 5:
cout << "exit ";
    break;
default : {
   cout << 'Invalid'
}
}
while (ch! =5);

   return 0;
}
```

14/11