8/7/25

1. Write a c++ program for addition of two numbers.

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, sum=0;
    cout<< "Enter value of a & b: ";
    cin>> a>>b;
    sum = a+b;
    cout<< "Addition of two numbers = "<<
                                    sum;

    return 0;
}
```

Output:
Enter value of a & b: 10 5
Addition of two numbers = 15

2. Write a c++ program to check wheather the number is even or odd.

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

```
{
int a ~~b~~;
cout<<"Enter value of a:";
cin>>a;
~~cout<<"Enter value of b:";~~
~~cin>>b;~~
if(a%2==0)
{
cout<<"The number is even"}
}
else
{
cout<<"The number is odd";
}
return 0;
}
```

3. C++ code to print 1 to 10 numbers using for loop.

```
#include<iostream>
Using namespace std;
int main()
```

```
{
    int i;
    cout << "The numbers between 1-10 are: " << endl;
for(i=1 ; i <=10 ; i++)
    {
      cout << i << endl;
    }
return 0;
}
```

4. c++ code to print 10-1 numbers using while loop.

```
#include<iostream>
Using namespace std;
int main()
{
    int i;
    i= 10;

while(i>1)
{
    cout<<i;
      i--;
}
return 0;
}
```

Q. C++ code to print the following pattern:

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

```
#include<iostream>
Using namespace std;
int main()
{
  int i,j;
  for (i=1; i<=4; i++)
    { cout<<"   ";
      for (j=1; j<=4-i ; j++)
      {
        cout<<" * ";
      }
    cout<<endl;
  }
return 0;
```

b.

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

```
#include<iostream>
Using namespace std;
int main()
{
 int i,j;
for(i=1; i<=5 ;i++)
   {
    for(j=2 ; j= i+1 ; j++ )
      {
        cout<< i ;
      }
   cout<<endl;
   }
return 0;
}
```

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

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

9/7

Experiment-1

6 Write a c++ program to declare class Student having data members as roll no. & name. Accept & display data for single student

```
#include<iostream>
#include<string>

class Student.
{
   private:
   int roll no;
   string name;

   Public:
   void inputs()
      {
      cout<< "Enter your name: ";
      cin >> name;
      cout<< "Enter your Roll no: ";
      cin >> roll no;
      cout<< student info:"<< endl;
      cout << "Name: "<<name;
      cout<< "Rollno: " << roll no;
      }
};

int main()
{
```

```
Student obj;();
obj.inputs();
 return 0;
}
```

Output:

```
Enter your name: Aditya
Enter your Roll no: 57
Student info:
Name: Aditya.
Roll no: 57
```

7. Write a progam in c++ to create a cl book having data members as b_nam b_price, b_pages. Accept the data for 2 books & display the name of book having greater price.

```
#include<iostream>
#include<string>

class Book..
{
 private:
 int b_pages;
 String b_name;
```

```
Public
    int b_price;
    void takeInputs()
    {
        cout<<"Enter book details:"<<endl;
        cout<<"Enter book name: ";
        cin>>b_name;
        cout<<"Enter book price: ";
        cin>>b_price;
        cout<<"Enter book pages: ";
        cin>>b_pages;
    };
int main()
{
    Book obj1, obj2;
    if(obj1.b_price > obj2.b_price)
    {
        cout<<"Book name: "<<obj.1 b_name<<
                                    endl;
        cout<<"Book price:"<<obj1.b_price<<endl;
        cout<<"Book pages:"<<obj1.b_pages<<endl;
    }
```

```
else
{
cout<<"Book name: "<<obj2.b_name<<"\n";
cout<<"Book price: "<<obj2.b_price<<"\n";
cout<<"Book pages: "<<obj2.b_pages<<"\n";
}

return 0;
}
```

Output:

```
Enter Book details:
Enter book name: Aditya.
Enter book price: 650
Enter book pages: 100

Enter book details:
Enter book name: Aditya2
Enter book price: 500
Enter book pages: 120

Book name: Aditya.
Book price: 650
Book pages: 100
```

8. WAP to declare a class time having data members as H, M & S. Accept data for one object & display total time in seconds.

```
#include<iostream>
using namespace std;
class Time
{
private:
  int h,m,s;

public:
  void inputTime()
  {
    cout<<"Enter hours: ";
    cin>>h;
    cout<<"Enter minutes: ";
    cin>>m;
    cout<<"Enter seconds: ";
    cin>>s;
  }
int seconds()
  {
    return h*3600+m*60+s;
  }
```

```
void displayTotalSeconds()
{
    cout << "Total time in seconds: " <<
    seconds() << endl;
  }
};
int main()
{
Time obj;
obj.inputTime();
obj.displayTotalSeconds();
return 0;
}
```

Output:

```
Enter hours: 12
Enter minutes: 12
Enter seconds: 12
Total time in seconds: 43932
```

Experiment - 2

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

```
#include<iostream>
using namespace std;
class city
{
  Public:
  string name;
  int population;

  Void accept()
  {
    cout<<"Enter city name: ";
    getline(cin, name);
    cout<<"Enter city population";
    cin>>Population;
  }
  void display()
  {
    cout<<"City name: "<<name;
    cout<<" Population: "<<population;
  }
};
```

```
int main()
{
  city c[5];
  int i, max;
  for(i=0 ; i<5 ; i++)
  { cout<<"Enter population of city"<<i<<
    c[i].accept();
  }
max=[0].population;
for(i=0; i<5; i++)
  {
if (c[i].population>max)
  {
    max = c[i].population;
  }
  }
cout<<"City with highest population
    <<max.name;
return 0;
}
```

2. WAP to declare a class Account having data members as account no. and balance. Accept this data for 10 accounts and give interest of 18% where balance is equal or greater than 5000 and display them.

```
#include<iostream>
using namespace std;
class Accounts
{
Public:
    int accountNumber;
    float balance;

void acceptData()
  {
    cout<<"Enter account number:";
    cin>>accountNumber;
    cout<<"Enter balance:";
    cin>>balance;
  }
};

int main()
{
  Accounts Obj[10];
  int i;
  float interest;
```

```
for(i=0; i<10; i++)
  {
    cout << "Enter details for account: "
    << i+1 << endl;
    obj[i].accountData();
  }

for(i=0; i<10; i++)
{
  if(obj[i].balance >= 5000
  ~~(i=0; i<10; i++)~~ )
  {
    interest = obj[i].balance * 0.10;
    obj[i].balance += interest;
    cout << "Interested amount for
account " << obj[i].accountNumber <<
    is: " << obj[i].balance << endl;
  }
}

return 0;
}
```

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

```
#include<iostream>
#include<string>
using namespace std;
class Staff
{
  public:
  string name, post;
  void getData ()
  {
    cout<<"Enter staff name: ";
    getline(cin, name);
    cout<<"Enter staff post: ";
    cin>>post;
    cin.ignore();
  }

void displayData()
  {
    cout<<"Staff name: "<<name<<endl;
    cout<<"Staff post: "<<post<<endl;
  }
};
```

```
int main()
{
    Staff s1[5];
    for(int i=0 ; i<5 ; i++)
    {
        cout<<"Enter details for staff mem
            <<(i+1)<<" : "<<endl;
        s1[i].getData();
    }
for(int i=0; i<5 ; i++)
{
    if (s1[i].post == "HOD" || s1[i].post=
                    "hod")
    {
    cout<<"Details of staff member"<<(i+
    <<" : "<<endl;
    s1[i].displayData();
      break;
    }
    else
    {
cout<<"No HOD found in the list."
                <<endl;
    }
}
return 0;
}
```

Experiment - 3.

1. WAP to declare a class 'Book' containing data members as book_title, author_name and price. Accept & display information for one object using a pointer to that object.

```
#include<iostream>
#include<string>
using namespace std;
class Book
{
    private:
    string title; author;
    float price;

    public:

    void getData()
    {
      cout<<"Enter book title: ";
      getline(cin, title);
      cout<<"Enter book author: ";
      getline(cin, author);
      cout<<"Enter book price: ";
      cin>>price;
    } cin.ignore();
    }
```

```
void displayData()
{
cout<<"Book Title : "<<title<<endl;
cout<<"Book Author : "<<author<<endl;
cout<<"Book price : "<<price<<endl;
}
};
int main()
{
Book b1;
Book *ptr;
ptr= &b1;
cout<<"Enter details of the book: "
<<endl;
ptr->getData();
cout<<"Details of the book: "<<endl;
ptr->displayData();
return 0;
}
```

Output:

```
Enter details of the book:
Enter book title: Harry Potter
Enter book author: JK Rowling.
Enter book price: 1000,50
Details of the book:
Book title: Harry potter.
Book author: JK Rowling
Book Price : 1000,50Rs.
```

2. WAP to declare a class 'student' having data members roll_no; and percentage. Using this pointer invoke member functions to accept & display this data for one object of the class.

```
#include<iostream>
#include<string>
{
  private:
  string name;
   int rollno;
   float percentage;
   public:
   void getData()
   {
     cout<<"Enter student name: ";
     getline(cin, this->name);
```

```
cout << "Enter student roll number: ";
cin >> this->rollno;
cout << "Enter student percentage: ";
cin >> this->percentage;
cin.ignore();
}
void displayData()
{
this->getData();
cout << "Student Details: " << endl;
cout << "Student Name: " << this->name << endl;
cout << "Roll Number: " << this->rollno << endl;
cout << "Percentage: " << this->percentage << "%" << endl;
}
};
int main()
{
Student s1;
s1.displayData();
return 0;
}
```

Output:
Enter student name: Aditya Marake
Enter student roll number: 57
Enter student percentage: 91
Student Details:
Student name: Aditya Marake
Roll Number: 57.
Percentage: 91%.

3. Write a program to demonstrate the use of nested classes.

```
#include<iostream>
#include<string>
using namespace std;
class Student
{
  Public:
  string name;
  int rollno;
  void getData()
  {
    cout<<"Enter student name: ";
    getline(cin,name);
    cout<<"Enter student roll number: ";
  } cin>>rollno;
```

```
void displayData()
{
cout<<" Student Name: "<<name<<
cout<<" Student Roll Number: "<<roll
                              endl;
}

class Marks
{
 Public:
 int m1, m2;
 void getMarks()
  {
   Student obj1;
   obj1.getData();
   cout<<"Enter marks of OOP: ";
   cin>>m1;
   cout<<"Enter marks of ADS: ";
   cin>>m2;
  }
void displayMarks()
  {
   Student obj1;
   obj1.displayData();
   cout<<" Marks in OOP: "<<m1<<endl
   cout<<" Marks in ADS: "<<m2<<endl;
  }
};};
```

```
int main()
{
    Student :: Marks obj2;
    obj2.getMarks();
    obj2.displayMarks();
} return 0;
```

Output:

```
Enter student name: Aditya Manake
Enter student roll number: 57
Enter marks of OOP: 25
Enter marks of ADS: 32
Student Name: Aditya Manake
Student Roll Number: 57
Marks in OOP : 25
Marks in ADS : 32
```

29

Experiment - 4.

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

```
#include<iostream>
using namespace std;
class Result1
{
  private public:
    int m1, m2;
    void getData()
    {
      cout<< "Enter marks for subject 1:";
      cin>>m1;
      cout<< Enter marks for subject 2: ";
      cin>>m2;
    }

int display()
{
  return m1 + m2;
}
};
```

```
class Result2
{
 Public:
 int m3, m4;

  Void accept ()
  {
    cout<< "Enter marks for subject 2 ";
     cin >> m3;
    cout<< "Enter marks for subject 4: ";
     cin >> m4;

  }

 int displayData ()
 {
   return m3+m4;
 }
};

int main()
{
  Result1 obj1;
  Result2 obj2;
 obj1.getData();
 obj2.accept();
```

```
float avg=((obj).display()+obj2.display())/4.0;
cout<<"Average marks: "<<avg;
return 0;
}
```

Output:
Enter marks for subject 1: 65
Enter marks for subject 2: 72
Enter marks for subject 3: 89
Enter marks for subject 4: 95
Average marks = 80.25

2. WAP to find the greatest number among two numbers from two diff classes using friend function.

```
#include<iostream>
Using namespace std;
class Greatest2;
class Greatest
{
public:
int a;
void getData()
{
cout<<"Enter the first number:";
cin>>a;
}
friend void compare(Greatest &a,
                    Greatest2 &b);
```

```
};
class Greatest2
{
public:
  int b;
  void accept()
  {
    cout<<"Enter the second number:";
    cin>>b;
  }

  friend void compare(Greatest &n1,
                      Greatest2 &n2);
};

void compare(Greatest &n1, Greatest2 &n2)
{
  if (n1.a > n2.b)
    cout<<"The greatest no is:"<<n1.a;
  else
    cout<<"The greatest no. is:"<<n2.b;
}

int main()
{
  Greatest obj1;
  Greatest2 obj2;
  obj1.getData();
  obj2.accept();
  compare(obj1, obj2);
  return 0;
}
```

```
float avg=((obj).display()+obj2.displayData())/4.0;
cout<<"Average marks: "<<avg;
return 0;
}
```

Output:
Enter marks for subject 1: 65
Enter marks for subject 2: 72
Enter marks for subject 3: 89
Enter marks for subject 4: 95
Average marks=80.25

2. WAP to find the greatest number among two numbers from two diff classes using friend function.

```
#include<iostream>
Using namespace std;
class Greatest2;
class Greatest
{
public:
int a;
void getData()
{
cout<<"Enter the first number:";
cin>>a;
}
friend void compare(Greatest &a,
                    Greatest2 &b);
```

```
};
class Greatest2
{
public:
 int b;
 void accept()
 {
  cout<<"Enter the second number:";
  cin>>b;
 }

 friend void compare(Greatest &n1,
                     Greatest &n2);
};

void compare(Greatest &n1, Greatest &n2)
{
  if (n1.a > n2.b)
     cout<<"The greatest no is: "<<n1.a;
  else
  cout<<"The greatest no. is: "<<n2.b;
}
int main()
{
  Greatest obj1;
  Greatest2 obj2;
  obj1.getData();
  obj2.accept();
  compare(obj1, obj2);
  return 0;
}
```

Output:
Enter the first number: 5
Enter the second number: 10
The greatest number is: 10.

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

```
#include<iostream>
using namespace std;
class Swapping
{
private:
    int value;

Public:
    void getData()
    {
        cout << "Enter a number: ";
        cin>>value;
    }
void display()
    {
        cout <<value <<endl;
    }

friend void swapValues(Swapping &x,
                       Swapping &y);
};
```

```
void swapValues(Swapping &n1, Swapping &n2)
{
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp;
}

int main()
{
    Swapping obj1, obj2;
    obj1.getData();
    obj2.getData();

    swapvalues(obj1, obj2);

    cout<<"After swapping :\n";
    obj1.display();
    obj2.display();

    return 0;
}
```

Output:

```
Enter a number:5
Enter a number:10
After swapping:
10
5
```

4. 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 Number
{
  int value;

Public:
  Number (int v=0)
  {
    value=v;
  }

void display ()
{
  cout<< value<<endl;
}

void swap (Number &obj)
{
  int temp=value;
  value=obj.value;
  obj.value=temp;
}
};
```

```
int mai
{
  Num
  cout
  cou
  cou
}
```

```
int main()
{
    Number n1(10), n2(20);
    cout << "Before Swap: " << endl;
    cout << "n1 = " << n1.display();
    cout << "n2 = " << n2.display();

    n1.swap(n2);

    cout << "\n After swap: " << endl;
    cout << "n1= " << n1.display();
    cout << "n2 = " << n1.display();
    return 0;
}
```

Output:

Before Swap:
n1 = 10
n2 = 20

After swap:
n1 = 20
n2 = 10.

5. Swap two numbers from different class using friend function.

```
#include<iostream>
using namespace std;
class Num2;
class Num1
{
Public:
  int n1;
  void accept()
  {
   cout<<"Enter first number:";
   cin>>n1;
  }

void display()
{

cout<<"Num1:"<<n1<<endl;
}

friend void Swap(Num1 & n1, Num2 & n2);
};

class Num2
{
Public:
  int n2;
```

```
void accept()
{
 cout<<" enter second number: ";
  cin>>n2;
}

 void display()
 {
  cout<<" Num 2: "<<n2<<endl;
 }

 friend void swap(Num1 &nu1, Num2 &nu2);
};
void swap(num1 & nu1, num2 &nu2)
 {
    int temp= nu1.n1;
    nu1.n1= nu2.n2;
  nu2.n2= temp;
   cout<<" Swapped:\n ";
   cout<<" first : "<<nu1.n1<<endl;
   cout<<" Second : "<<nu2.n2<<endl;
 }
int main()
{
 Num1 nu1;
Num2 nu2;
```

```
num1.accept();
num2.accept();
num1.display();
num2.display();
swap(num1, num2);
return 0;
}
```

Output:

```
Enter first number: 5
Enter second number: 9
Num1: 5
Num2 9
Swapped:
First: 9
Second: 5
```

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

```
#include<iostream>
using namespace std;
class B;
class A
{ int A;
public:
```

```
void accept()
{
 cout << " Enter value for A: ";
 cin >> A;
}
friend int sum(class A & oA, class B & oB)
} n1;
class B {
  int B;
public:
void accept()
{
   cout << "Enter value for B: ";
} n2;
int sum(class A oA, class B oB);
{
   return oA.A + oB.B;
}
int main()
{
   n1.accept();
   n2.accept();
   cout << "Sum: " << sum(n1, n2) << endl;
} return 0;
```

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

```
#include <iostream>
using namespace std;
class Number
{
    int num;
public:
    void accept ()
    {
        cout << "Enter value: ";
        cin >> num;
    }

void display ()
    {
        cout << "Value: " << num << endl;
    }
friend void swapNumbers(Number &
                        num1, Number &num2);
} n1, n2;

void swapNumbers (Number &, Number &)
{ int temp = n1.num;
  n1.num = n2.num;
  n2.num = temp; }
```

```
int main()
{
  n1.accept();
  n2.accept();
  cout<<"Before swap:"<<endl;
  n1.display();
  n2.display();
  swapNumbers(n1, n2);
  cout<<"After swap:"<<endl;
  n1.display();
  n2.display();
  return 0;
}
```

8. Define two classes Box & cube each having a private volume. Write a friend function findgreater(Box, cube) that determines which obj has larger volume.

```
#include<iostream>
using namespace std;
class Cube;
class Box
{
    double V;
  public:
  void accept()
  {
    double l,w,h;
    cout<<"Enter length, width
        & height for box:";
```

```
cin >> l >> w >> h;
V = l * w * h;
}
friend void findGreater(Box b, Cube);
} b;
class Cube {
    double V;
public:
void accept()
{
    double s;
cout << "Enter side length for cube";
cout cin >> s;
    V = s*s*s*;
}
friend void findGreater(Box b, Cube c)
} c;
void findGreater(Box, cube)
{
    String res = (b.V > c.V)? "Box" : ((c.V b.V)
    "cube" : "Both equal");
    cout << "Greater volume :" res << endl;
}
```

```
int main()
{
    b.accept();
    c.accept();
    findtreater(b,c);
    return 0;
}
```

9.
```
#include<iostream>
using namespace std;
class Complex
{
    double r,i;
public:
    void accept()
    {
        cout<<"Enter real and imaginary parts: ";
        cin>>r>>i;
    }

    void display()
    {
        cout<<r<<"+"<<i<<";"<<endl;
    }

friend Complex add(Complex c1, Complex c2);
};c1,c2;
Complex add(Complex, Complex)
{
    Complex sum;
```

```
    sum.r = c1.re + c2.r;
    sum.i = c1.i + c2.i;
    return sum;
}
int main()
{
    c1.accept();
    c2.accept();
    complex sum = add(c1, c2);
    cout << "The sum is: ";
    sum.display();
    return 0;
}
```

10.
```
#include<iostream>
using namespace std;
class Student {
    string n;
    int m[3];

public:
    void accept()
    {
    cout << "Enter student name: ";
    cin >> n;
    cout << "Enter marks for three subjects: ";
    cin >> m[0] >> m[1] >> [illegible] /3.0;
```

```
cout << "Student :" << sn << endl;
cout << "Avg: " << avg << endl;
}
int main()
{
  s.accept();
  calculateAverage(s);
  return 0;
}
```

11.
```
#include<iostream>
using namespace std;
class Beta;
class Gamma;
class Alpha
{
  int n;
  public:
  void accept()
  {
  cout << "Enter value for alpha :";
  cin >> n;
  }
};
class Beta {
  int n;
  public:
```

```
void accept()
{
cout<<"Enter value for Beta:";
  cin>>n;
}
friend int sum(Alpha a, Betab, Gamma c);
}b;
class Gamma
{
int n;
 public:
void accept()
{
 cout<<"Enter value for gamma:";
 cin>>n;
}
friend int sum(Alpha a, Betab,
                    Gamma c);
}c;
int sum(Alpha a, Betab, Gamma c)
{
  return a.n+b.n+c.n;}
int main()
{
  a.accept();
 b.accept();
```

```
c.accept();
cout<< "Sum: " <<sum(a,b,c) <<endl;
return 0;
}
```

14.
```
#include<iostream>
#include<math>
using namespace std;
class Point
{
 double x,y;
 public:
 void accept(){
 cout<<"Enter x and y cordinate: ";
 cin>>x>>y;
 }
 friend double dist (Point p1, point p2);
};
p1, p2;

double dist (Point, point)
{
 double dx= p1.x -p2.x;
 double dy= p1.y - p2.y;
 return sqrt ((dx*dx)+(dy*dy));
}
int main()
{
```

```
p1.accept();
p2.accept();
cout<<"Distance "<<dist(p1,p2)<<endl;
return 0;
}
```

13.
```
#include<iostream>
using namespace std;
class BankAccount;
class Audit
{
public:
  void pBal(BankAccount);
};aud;
class BankAccount{
double b;
 public:
  void accept()
 {
  cout<<"Enter bank account
              balance:";
    cin>>b;
 }
friend void Audit::pBd(BankAccount);
{ba;
void Audit::pBal(BankAccount){
  cout<<"Auditing... BankAcount
                balance:₹"<<ba.b<<e
```

```
}
int main()
{
b.accent();
aud.pBrd(ba);
return 0;
}
```

Q1
21g

Exp-5

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

```
#include <iostream>
using namespace std;
class Sum
{
  int n, sum;
  public:
   Sum(int num)
   {
    n=num;
    sum=0;
    for(int i=0; i<n; i++)
    {
      sum+=i;
    }
   cout<<"Sum= "<<sum;
}
};
int main()
{
  Sum s(10);
}return 0;
```

Output:
Sum: 55

b.) WAP to declare a class student having data members as name and percentage. Write a constructor to initialize these data members. Accept & display data for one student.

```
#include<iostream>
#include<string>
using namespace std;
class Student
{
  private:
  string name;
  float percentage;
  public:
Student(string n, float p)
{
  name = n;
  percentage = p;
  cout<< "Name: ";
  getline(cin, name);
  cout<< "Percentage: ";
  cin >> percentage;
}
void display()
{
  cout<< "Student name: " <<name<<endl;
  cout<< "Student percentage: " <<percentage
  <<endl;
};
```

```
int main()
{
    Student S1("Aditya", 85.5);
    S1.display();
    return 0;
}
```

Output:
Name: Aditya.
Percentage: 91
Student Name: Aditya.
Student percentage: 91.

c. Define a class College members variables as roll no, name, course, WAP using constructor with default value as Computer Engineering for course Accept this data for two objects of class and display the data

```
#include<string>
#include<iostream>
using namespace std;
class College
{
private:
string course_name, stud_name;
int roll_no;
public:
college()
{   roll_no=57;
```

```
    stud_name = "Aditya";
    course_name = "CSE";
}
void display()
{
    cout << "Roll no: " << roll_no << endl;
    cout << "Student name: " << stud_name <<
                                endl;
    cout << "Course name: " << course_name
                                << endl;
}
};
int main()
{
    college c1;
    c1.display();
    return 0;
}
```

Output:
Roll no: 57.
Student name: Aditya.
Course name: CSE.

d. Write a program to demonstrate constructor overloading.

```
#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(5,6);
     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

17/10

Experiment - 6

a. WAP to implement multilevel inheritance. Assume suitable data.

```
#include <iostream>
#include <string>
using namespace std;
class Person
{
  protected:
   int age;
   string name;
};

class Student:public Person
{
  private:
   int roll_no;
  public:
void accept()
{
 cout<< "Enter your name: ";
  cin >> name;
 cout<< "Enter your age: ";
  cin >> age;
cout<< "Enter your roll no: ";
cin >>roll_no;
}
```

```
void display()
{
  cout<< "Name:\n" << name;
  cout << "Age:\n" << age;
  cout << "Roll no:\n" <<roll_no;
  }

};

int main ()

{
  Student s1;
s1.accept();
  s1.display();
   return 0;

}
```

Output:

Enter your name: Aditya.
Enter your age: 17.
Enter your roll number: 57.
Name: Aditya.
age: 17.
Roll no: 57.

b. Write a program to implement multiple
Assume suitable data

```
#include<iostream>
using namespace std;
class Academic
{
  protected:
    int marks;
};

class Sports
{
  protected:
    int score;
};

class Result: protected Academic,
              protected Sports
{
  int total_score=0;
  public:
  void accept(){
  cout<<"Enter the marks of the
        student:";
  cin>>marks;
  cout<<"Enter the sports score of
      the student:";
  cin>>score;
}
```

```
void calculate ()
{
  total_score = marks + score;
  cout << "The marks of the student is:
       << marks << endl;
  cout << "The sports score of the
       student is: << score << endl;
  cout << " The total score of the
       student is: " << total_score << endl;
  }
};

int main()
{
  Result r;
  r.accept();
  r.calculate();
  return 0;
}
```

Output:

```
Enter the marks of the student: 99
Enter the sports score of the student: 95
The marks of the student: 99.
The sports score of the Student: 95
The total score of the student: 194.
```

c. WAP to implement hierarchical inheritance. Assume suitable data.

```
#include<iostream>
using namespace std;
class Vehical
{
  protected:
   string brand;
   int model;
};
class car:protected Vehical
{
  protected:
    string type;
};
class ElectricCar : protected Car
{
 int batterycapacity;
 public:
void accept()
{
 cout<<"Enter the brand of car:";
 cin>>brand;
cout<<"Enter the model of the
      car:";
 cin>>model;
cout<<"Enter the type of car:";
cin>>type;
```

```
cout << " Enter the battery capacity of
                the car: ";
cin >> batterycapacity;
}

void display ()
{
  cout << "\n The brand of the car: " << brand
                                    << endl;
cout << " The model of the car: " << model <<
                                    endl;
cout << " The type of the car: " << type << endl;
cout << " The battery capacity of the car: " <<
          batterycapacity << endl;

  }
};

int main()
{
  Electriccar e;
  e.accept();
  e.display();
  return 0;
}
```

Output:

```
Enter the brand of the car: BMW
Enter the model of the car: M4
Enter the type of the car: Sedan Sport
Enter the battery capacity : 115.
```

d. WAP to implement hybrid inheritance.

```
#include<string>
#include<iostream>
using namespace std;
class Person
{
  public:
  string name;
  int age;
  void getPersonDetails()
  {
    cout<<"Enter name: ";
    cin>>name;
    cout<<"Enter age: ";
    cin>>age;
  }
  void showDetails()
  {
  cout<<"Name"<<name<<", Age: "<<age<<endl;
  }
};
class Student: public Person
{
  public:
  string course;
  void getDetails()
  {
    cout<<"Enter course: ";
    cin>>course;
```

```
{void showDetails()
{
  cout<<"course: "<<course<<endl;
}
};

class Employee: public Person
{
  public:
    string company;
    {void getEmployeeDetails()
    {
      cout<<"Enter company: ";
    }cin>>company;
    }

{void showEmployeeDetails()
{
}{cout<<"Company: "<<company<<endl;
}
};

class Intern: public Student, public
  Employee{
public:
  {void showInternDetails()
  {
      cout<<"\n---Intern Details---\n";
      Student::showPersonDetails();
      showEmployeeDetails();
  }
};
```

```
int main()
{
    Intern it;
    it.Student::getPersonDetails();
    it.getDetails();
    it.getEmployeeDetails();
    it.ShowInternDetails();

} return 0;
```

Output:
Enter name: Aditya
Enter age: 21
Enter course: ComputerScience
Enter company: Microsoft.

--- Intern Details ---
Name: Aditya, Age: 21
Course: ComputerScience
Company: Microsoft

17/10

Experiment 7

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

```
#include<iostream>
using namespace std;
class Area;
{ private:
  float d, b;
  public:
  void area(float d)
  { float area;
    area = d*d;
  { cout<< "Area of square: "<<area<<endl;
void area (float d, float b)
{
  float area;
  area = d* b;
  { cout<< "Area of rectangle: "<<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 value & 10 int.

```
#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);

  return 0; }
```

Output:

Sum of 5 floating numbers: 20.2
Sum of 10 integers: 55

WAP to implement unary operator when used with the obj so that the numeric data member of the class is negated

```
#include<iostream>
using namespace std;
class Number
{
  private: int x;
  public: void accept()
    {
      cout<<"Enter a number: ";
      cin>>x;
    }
  void operator-()
  {
    x=-x;
  }
  void display()
  {
    cout<<"Negated number: "<<x<<endl;
  }
};
int main()
{
  Number n1;
  n1.accept();
  -n1;
  n1.display();
  return 0;
}
```

Output:

```
Enter a number: 5
Negated number: -5
```

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

```
#include<iostream>
using namespace std;
class Number
{
  private: int x;
  public: void accept();
          {
            cout<<"Enter a number";
            cin>>x;
          }
temp=x;
}
void operator ++ ()
{
  x=++x;
}
Void reset()
{
  x=temp;
}
void operator ++ (int)
{
  x=x++;
}
void display()
{ cout<<"The number is:"<<x<<endl;
}
```

```
void display2()
{
  cout<<" (post) The number is: "<<x;
}
};
int main()
{
  Number n1;
  n1.accept();
  ++n1;
  n1.display();
  n1.reset();
  n1++;
  n1.display2();
  return 0;
}
```

Output:

Enter a number: 5
(pre) The number is: 6
(post) The number is: 5

Qn
17/10

Experiment - 8

1) WAP to overload '+' operator so that two strings can be concatenated. Eg: "X42" + "PQR" = "X42PQR"

```
#include<iostream>
#include<string>
using namespace std;
class MyString {
public: string str;
    MyString operator+ (MyString& other){
        MyString temp;
        temp.str = str + other.str;
        return temp;
    }
    void display()
    {
        cout<<str<<endl;
    }
};
int main()
{
    MyString s1, s2, s3;
    s1.str = "X42";
    s2.str = "pqr";
    s3 = s1 + s2;
    cout<<"Concatenated string: ";
    s3.display();
    return 0;
}
```

Output:
Concatenated string:
X42pqr.

b)
```
#include<string>
#include<iostream>
using namespace std;
class Login {
  protected: string name, password;
  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<<"\n---Email Login Details---\n";
      cout<<"Name:"<<name<<"Password:"<<
    }password<<"Email:"<<email<<endl;
};
class MembershipLogin:public Login {
      string membership ID;
    public:
      void accept() override{
        Login::accept();
        cout<<"Enter membership ID: ";
      } cin>>membershipID;
```

```
    display() {
cout<<"\n--- Membership Login Details---\n";
cout<<" Name: "name<<"\n password:"<<password<<
  "MembershipID: "<< membershipID;<<endl;
 }
};
int main()
{ EmailLogin e;
  MembershipLogin m;
   e.accept();
   m.accept();
   e.display();
   m.display();
 return 0;
}
```

Output:

Enter Name: Aditya.
Enter password: 123
Enter email: abc@gmail.com.
Enter name: Aditya.
password: 745
Enter membership ID: 5745
--- Email Login Details ---
Name: Aditya.
password: 123.
Email: abc@gmail.com.
--- Membership Login Details ---
Name: Aditya.
password: 745
Membership ID: 5745

17/10

Experiment - 9.

9.) 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;
    first_file.open("first.txt", ios::in);
    if (!first_file)
    {
        cout<<"Error opening first.txt!"<<endl;
        return 1;
    }
    second_file.open("second.txt", ios::out);
    if (!second_file)
    {
        cout<<"Error opening second.txt!"<<endl;
        return 1;
    }
    char ch;
    while(first_file.get(ch))
    {
        second_file.put(ch);
    }
    first_file.close();
    second_file.close();
}
```

cout<<" File ~~open~~ 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.txt", ios::in);
if(!new_file)
{
cout<<"Error occured while opening!"<<";
endl;
return 1;
}
int digit_count=0;
int spaces_count=0;
char ch;
while(new_file.get(ch)){
{if(isdigit(ch))
{ digits_count++;}
```

Exp - 9.

```
if (isspace(ch)) {
    space_count++;
}
new_file.close();
cout << "Number of digits: " << digits_count << endl;
cout << "Number of spaces: " << spaces_count << endl;
return 0;
}
```

c) WAP to count words using file handeling

```
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;
{
    fstream new_file;
    new_file.open("first.txt", ios::in);
    if(!new_file)
    {
        cout << "error occured!" << endl;
        return 1;
    }
    char ch;
    int words_count = 0;
    bool inWord = false;
    while (new_file.get(ch))
    {
```

```
{ if (isspace(ch))
    inWord = false; }
else if (!inWord)
{
    words_count++; }
    inWord = true; }
}
new_file.close();
cout << "Number of words: " << words_
                    count << endl;
return 0;
}
```

d.) WAP to count occurance of a given word.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{ fstream new_file;
  new_file.open("first.txt", ios::in);
  if(!new_file)
  { cout << "Error in opening first.txt!"
                    << endl;
  } return 1;
```

```
string target = "Aditya";
string word;
int count = 0;
while(new_file >> word)
{
  if (word = target)
  { count++;
  }
new_file.close();
cout<< "The word '" <<target<< "'
 occured "<< count << "times." <<endl;
 return 0;
}
```

17/10

DATE:

Experiment - 10

Q1. WAP 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 findsum(T arr[], int n)
{
    T sum=0;
    {for(int i=0; i<n; ++i)
    {  sum+= arr[i];
    }
    return sum; }
int main()
{
    int size=10;
    int intArr[size]={1,2,3,4,5,6,7,8,9,10};
    float floatArr[size]={1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.10};
    double doubleArr[size]={1.11,2.22,3.33,4.44,5.55,6.66,7.77,8.88,9.99,10.10};
```

```
cout<< "Sum of integer array: " <<
findSum(intArr, size) <<endl;
cout<< "Sum of float array: " <<findSum
(floatArr, size);
cout<< "Sum of doubleArr: " << findSum
(doubleArr, size) <<endl;
} return 0;
```

O/P:
Sum of integer array = 55
Sum of float array = 59.6
Sum of double array = 60.05.

Q2. WAP to sq. fn using template specialization

```
#include<bits/stdc++.h>
using namespace std;
template<class T>
T square(T num) {
    return num*num; }
template<>
string square<string>(string str){
    return str+str; }
```

array: "<<

```
int main()
{
  int intNum=5;
  string str= "Hello";
  cout<<"Square of integer "<<intNum<<
  "= "<<square(intNum)<<endl;
  cout<<"Square of string "<<str<<" :
  "<<square(str)<<endl;
  return 0; }
```

O/P:
square of integer=25
square of string=HelloHello.

3. WAP to build a simple calculator using class template.

```
#include<bits/stdc++.h>
using namespace std;

template<class T>
class Calculator {
  private:
    Tnum1, Tnum2;
  public:
    Calculator(Tn1, Tn2) {
    num1=n1;
    num2=n2; }
```

```
T add() { return num1+num2; }
T substract() { return num1- num2; }
T multiply() { return num1*num2; }
T divide() {
    if (num2!= 0)
        return num1/num2;
    else {
        cout<< "Error! division by zero. "<<
                endl;
        return 0; } }

void displayResults() {
    cout<< "Numbers: "<< num1<< " and "<<
    num2<<endl;
    cout<< "Addition= "<<add()<<endl;
    cout<< "Substraction= "<< substract()<<endl;
    cout<< "Multiplication= "<<multiply()<<endl;
    cout<< "Division= "<<divide()<<endl;
};
int main() {
cout<< "Simple Calculator using class
    Template \n";
Calculator<int> intCalc(10,5);
cout<< "\n Integer calculator:- \n";
    intCalc.displayResults();
Calculator<float> floatCalc(10.5, 2.5);
cout<< "\n--- float calculator---\n";
floatCalc.displayResults();
    return 0; }
```

O/P:
Simple calc
--- Integer
Numbers: 10
Addi
Su

O/P:

```
Simple calculator using class Template
--- Integer Calculator ---
Numbers: 10 and 5
Addition = 15
Substraction = 5
Multiplication = 50
Division = 2
--- Float Calculator ---
Numbers: 10.5 and 2.5
Addition = 13
Substraction = 8
Multiplication = 26.25
Division = 4.2
```

4. WAP to implement push & pop methods from stack using class template

```
#include<bits/stdc++.h>
using namespace std;
#define SIZE 5
template<class T>
class Stack{
  private:
    T arr[size];
    int top;
  public:
  stack(){
    top=-1;
}
```

DATE:

```
void push(T value){
  if (top == size-1) {
    cout<< "Stack Overflow! cannot push "<<
    value <<endl;
  }else{
    arr[++top] = value;
    cout<<value << "Pushed into stack "<<
                    endl; } }

void pop () {
  if (top==-1){
    cout<< "Stack is empty "<<endl;}
  else{
    cout<< arr[top--] << " popped from
stack "<<endl; } }

void display(){
  if (top == -1) {
    cout<< "Stack is empty "<<endl; }
  else{
    cout<< "Stack elements: ";
    for(int i = top; i >= 0; i--) {
      cout<<arr[i]<<" ";
      Cout <<endl; } } };
int main() {
cout<< "Stack implementation using class
template\n";
```

```
stack<int> intstack;
cout<<"\n--Integer stack Operations--\n";
int stack.push(10);
intStack.push(20);
intstack.push(30);
intstack.display();
intStack.pop();
intstack.display();

stack<string> strstack;
cout<<"\n--String stack Operations--\n";
strstack.push("Apple");
strStack.push("Banana");
strstack.display();
strstack.pop();
strStack.display();
return 0;
}
```

O/P:

--Integer stack operations--
10 pushed into stack
20 pushed into stack
30 pushed into stack.
Stack elements: 30 20 10
20 popped from stack.
stack elements: 20 10
--String stack operations--
Apple pushedinto stack.
Banana pushed into stack.
Stack elements: Banana Apple
Banana popped from stack
Stack elements: Apple.

DATE:

Exp- 11

81. WAP to implement generic vectors. Include following member functions
a.) To modify the value of a given elements.
b.) To multiply by a scalar value
c.) To display the vector in form (10,20,30)

```
#include<bits/stdc++.h>
using namespace std;

template<class T>
class GenericVector{
private:
    vector<T>V;
public:
    void createVector(int size){
    T element;
    cout<<"Enter "<<size<<" elements:\n";
    for(int i=0; i<size ;i++)
    {  cin>>element;
        V.push_back(element);}}

void modifyElement(int index, T newValue){
if(index>=0 && index<V.size()){
    V[index]=newvalue;
cout<<"Element modified sucessfully\n";
}else{cout<<"Invalid index\n";
    }}
```

```
void multiplyByScalar(T scalar) {
    for (int i = 0; i < v.size(); i++) {
        v[i] *= scalar; }
    cout << "Vector multiplied by scalar
            Successfully\n"; }
void displayVector() {
    cout << "(";
{for(int i=0; i<v.size(); i++)
{
    cout << v[i];
    if (i != v.size() - 1)
    { cout << ", "; }
}cout << ")\n";
}
};

int main(){
GenericVector<int> vec;
int size, index, newValue, scalar;
cout << "Enter size of vector: ";
cin >> size;
vec.createVector(size);
cout << "\nInitial vector: ";
vec.displayVector();
cout << "\nEnter index to modify: ";
cin >> index;
cout << "Enter new value: ";
```

DATE:

```
vec.modifyElement(index, newValue);
cout<<"\n After Modification: ";
vec.displayVector();
cout<<"\n Enter Scalar value to multiply: ";
cin>>scalar;
vec.multiplyVector(scalar);
cout<<"\n After multiplying by Scalar: ";
vec.displayVector();
return 0;
```

2. Accessing vector using iterators.

```
#include<bits/stdc++.h>
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='a'+i;
p++;
i++;}
```

```
cout<< "Orignal Content:\n";
  p=v.begin();
while (p!=v.end()){
     cout<< *p<<" ";
      p++; }
```

O/P:

Vector contents:

a b c d e f g h i j

Without iterator:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
  vector<char>v(10);
  unsigned int i;
  cout<< size= <<v.size()<<endl;
for(i=0; i<10; i++)
{   v[i]=i+'a'; }
```

DATE:

```
cout<<"Current Content:";
for(i=0; i<v.size(); i++){
    cout<<v[i]<<" ";}

cout<<"\n\n";
cout<<"Expanding Vector:\n";
{for(i=0; i<10; i++)
    v.push_back(i+10+'a');}
cout<<"Size now = "<<v.size()<<endl;
cout<<"Current contents:\n";
{for(i=0; i<v.size(); i++)
    v[i]=toupper(v[i]);
}

cout<<"Modified Contents:\n";
{for(i=0; i<v.size(); i++)
{
    cout<<v[i]<<" ";
}
cout<<endl;
{return 0;
}
```

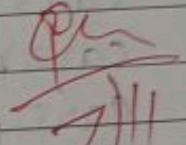
O/P:

DATE:

Current elements in vector are:
a b c d e f g h i j

Expanding Vector:

Modified contents:
A B C D E F G H I J



DATE:

b)

Exp - 12.

a) Implement Stack.

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    stack<char> cars;
    cars.push("BMW");
    cars.push("Audi");
    cars.push("Mercedes");
    cars.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;
}
```

O/P:

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

DATE:

b.) 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<<" Back element is: "<<age.back()<<
                                    endl;
while(!age.empty())
{
   cout<<"Element in queue are: "<<age.
                          front()<<endl;
   age.pop();}
return 0;}
```

O/P:

front element is: 21
Back element is: 24
Elements in queue are: 23
Elements in queue are: 24.