OOP

Course objectives
- To understand basic concepts of OOP.
- To implement real life objects in programming
- Reuse a common logic to reduce development time & ensures a higher level of security.
- To understand the principle of inheritance.
- Solve broad based engineering problems by applying basic knowledge & OOP language.

Unit 1 - Basic concepts of OOP, classes & objects

1) Code 1 :- Addition by C++ program

```
#include<iostream> using mainspace std;
int main()
{
int a,b,c

cout<<"Enter value a & b";
cin>>a>>b;
  c= a+b;

cout<<"Addition ="<<c;
  return 0;
}
```

Output:
Enter value a & b: 15 7
Addition = 22

2) Write a c++ program to perform arithmetic operations using switch case.

```
#include <iostream>
using namespace std;
int main ()
{
double num1, num2
 char op ;

cout << Enter two numbers: ";
 cin >>num1 >>num 2 ;

cout "Enter an operator ( +, -, *, / )";
 cin << op ;

switch (op) {

case '+'
cout << "Result " << (num1 +num2)
 << endl;
 break ;
.
case '-' :
cout << "Result", << (num1 - num 2)
 <<endl;
 break ;
case '*';
cout << "Result, << (num1 / num2)
 <<end l;
 break ;
```

```
case '/';
  if (num2 != 0)
   cout << "Result:" << (num1 / num2)
    else
   cout << "Error !" << endl;
    break;

   default;
   cout << "Invalid operator!" <<endl;
     }

   return 0;
     }
```

output:

Enter two numbers: 10,23

Enter a operator: (+,-,*,/) +

10 + 23 = 33

3 C++ program to check if no is even or odd

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

int num;

cout<<" Enter a no: ";
 cin >> num ;

 if (num % 2 == 0)
cout <<num <<" is even" << endl;
  else
cout <<num <<" is odd " < endl;
}
 return 0;
```

output :

Enter a no: 10

10 is even

4) C++ code to print 1-10 numbers using for loop.

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

For(i=1; i<=10; i++){

cout<<i<<" ";
}
cout<<endl;

return 0;
}
```

output:

1 2 3 4 5 6 7 8 9 10

5) C++ code to print numbers 10-1 using while loop.

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

output:

10 9 8 7 6 5 4 3 2 1

6) C++ code to print code with following pattern.

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

int i, j;

for (i=1; i<=5; i++){

for ( j=1 ; j<=5-i ; j++ ){

cout << " ";

}
for ( j=1;j<=i ;j++)

cout <<" * " ;

}

cout << endl;

}
return 0;
}
```

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

```
#include <iostream>
using namespace std;
 void main ()
 {
  int i,j ;

for(i= 1;i<=5;i++ ){

for(j =1 ;i<= i; j++){

 cout<<i<<"  ";

}

 cout<<endl;

}

return 0;
}
```

```
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;

for (i=1;i<=5;i++) {

for (j=1;j<=i;j++) {

cout<<j<<" ";

}
cout<<endl
}
return 0;
}
```

817125

Practical 1

a) WAP to declare class "Student" having data members as roll_no, name and class, accept & display data for 1 object.

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

class student{

int roll_no;
string name;
string student_class;

public:

void accept(){
cout<<"Enter Name:";
  cin>>name;
cout<<"Enter roll no:";
cin>>roll_no;
cout<<"Enter class";
cin>>class;
}

voiddisplay(){
cout<<"\n---Student Details---"<<endl;
cout<<"Name:"<<name<<endl;
cout<<"Roll no:"<<roll_no<<endl;
cout<<"Class:"<<class<<endl;
}
};
```

```
int main()
{
 student s1;

cout <<"Enter student details :" <<endl;
    s1.accept();

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

Output:

Enter student details:

Enter Name: Shreyash

Enter Roll no 3

Enter Class Sy

---Student Details---

Name: Shreyash

Roll number: 3

Class: Sy

b) WAP to declare "Book" having data members name, price & no. of pages. Accept this data for 2 objects & display name of book having greater price.

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

class Book {
  int b_pages;
  double b_price;
  string b_name;

public;
 void accept () {
 cout << "Enter Book name : ";
 cin.ignore ();
 getline (cin, b_name);
 cout << "Enter price of book : ";
 cin >> b_price;
 cout << "Enter Number of Book pages : ";
 cin >> b_pages;
 }
void display () {
cout << "\n ---Book Details" << endl;
cout << " Name of Book : " << b_name << endl;
cout << " Price of Book : " << b_price << endl;
cout << " No. of pages : " << b_pages << endl;
}
double getPrice () const {
 return b_price;
}
};
```

```
int main(){
    Book b1, b2;
    b1.accept();
    b2.accept();

    if (b1.getPrice() > b2.getPrice()){
        b1.display();
    }else{
        b2.display();

    }

    return 0;
```

output:

Enter Book Name : It Ends With Us
Enter Price of Book: 499
Enter Number of Book pages: 189
Enter Book Name : It Starts With Us
Enter Price of Book: 399
Enter Number of Book pages: 179

--- Book Details
Name of Book: It Ends With Us
Price of Book: 499
No. of pages: 189

c) WAP to declare a class 'time', accept time in HH:MM format convert it into total seconds & display them.

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

class Time {
public:
 int hours;
 int minutes;
 int seconds;

public:
 void accept () {
 cout<<" Enter hours: ";
  cin >> hours;

  cout<<" Enter minutes: ";
  cin >> minutes;
  cout<<" Enter seconds: ";
   cin >> seconds;
 }
 int toSeconds () {
    return (hours * 3600) + (minutes * 60) + seconds;
 }
void displayseconds () {
cout<<" Total seconds: " << toSeconds() << endl;
}
};
```

```
int main ()
{
    Time t;
    t.accept ();
    t.displaySeconds();
    return 0;
}
```

output :

```
Enter hours : 5
Enter minutes: 10
Enter seconds: 5
Total seconds: 18605
```

15/7

Experiment 2

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

```
#include<iostream>
using namespace std;
class city
{public:
 string name;
 int population;
  public:
 void accept()
 {
 cout<<"Enter name of city: "<<endl;
 cin >>name;
 cout "Enter population:" <<endl;
 cin >>population;
 }
 void display()
 {
cout<<"Name: "<<name<<endl;
cout<<"Population: "<<population<<endl;
}
};
city c[5];
int main()
{
int i;
```

```
for(i=0;i<  5;i++)
{
c[i].accept()
}
int max = 0;
 for (i=0; i< 5;i++)
{
if( c[i].population > c[max].population)
 {
 max =i;
}
}
cout <<" The city with maximum population:"<<endl;
c[max].display();
return 0;
}
```

2) WAP to delare a class 'Account' having data members as Accout no. & balance. Accept this data for 10 accounts & give interest for 70% where balance is equal or greater than 5000 & display.

```
#include<iostream>
using namespace std;
class account;
{
int acc no;
int bal;
public:
void accept()
}
cout<<"Enter account number:"endl;
cin >> acc no;
cout<<"Enter balance:"<<endl;
cin >>bal;
}
void display()
{
cout<<"Account number:"<< acc no <<"Balance:"<< bal <<endl;
}
void addinterest()
{
if(bal >= 5000)
{
```

```
int interest = bal * 0.10;
    bal += interest;
cout<<" Interest of 10% added :"<< bal <<endl;
 }
 else
}
cout<<" Balance is less than 5000, no interest
is added "<<endl;
}
}
int getBalance())
{
return balance;
}
}
 int main ()
{
 account accounts[10];
  int numAccounts;

cout<<" Enter number of accounts (max 10): "
 cin >> numAccounts;

for(int=0; i< num accounts;

for (int i=0; i< num acounts; i++)
{
cout("\n Acount <<[i+1]<<" endl;
accounts(i).accept.
```

```
cout<<("\n Account display with 10% interest"<<
endl;
for(i=0; sumeric neon
for(int i=0; i<name a ) i++)
{
cout<<"In account (c(i+1)-<< ":"<< caroli.

{
cout"In account (liv' with 100%. we had
fun!!
<cout "\n Account(cc(i+1)<<":"erdpride.
accout [i] display(i);
accout [i](accept)admision)
}
return 0;
```

3) WAP to declare a class 'staff' having data members of post staff, [redacted] & price. Accept & display info for obj using pointer.

```
#include<iostream>
using namespace std;
class staff
{
string name;
string post;
public:
void accept()
{
cout<<"Enter name:");
 cin>>name;
cout<<" Enter post:";
  cin>>post;
}
void display()
{
cout<<"Name:"<<name<<","Post:"<<post<<endl;
}
String getPost()
{
return post;
}
string getName()
{
return name;
}
};
```

```
int main ()
{
staff s[5];
bool found HOD = false;

for (int i=0; i< 5 ;i++)
 {
cout<<"staff member"<<(i+1)<<":"<<endl;
 s[i].accept();
}
for (int i=0; i<5; i++)
{
if (s[i].getPost()=="HOD")
{
cout<<"HOD is :"<< s[i].accept()<<endl;
 found HOD = True;
break ;
}
}
if (found HOD)
cout<<"HOD not found "<<endl;
return 0;
}.
```

Q1/
2018

Experiment 3

1) WAP to declare class 'book' having data members as book-title, author_name & price. Accept & display this info for obj using pointer.

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

class bookz {
string book_name;
string author_name;
int book_price;

public:
void accept (){
 cout<<" Enter book name:";
  getline(cin, this -> book_name);

 cout<<" Enter author name:";
 getline (cin, this->author_name);

 cout<<"Enter book price:";
   cin >> book_price;
 }
```

```
void display (){
cout <<"\n Name of book:"<< book_name;
cout<<"\n Name of author:"<<author_name;
cout<<"\n Book price :"<<book_price;
}
};

int main (){
book * b1 = & b;
  b1 ->accept();
  b1 -> display ();

return 0;
}
```

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

```
#include<iostream>
#include<string>
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 <<"\n Student roll no :"<<roll_no;
 cout << "\n Student percentage:"<<percentage;
}
};
```

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

  return 0;
  }
```

3) WAP to demonstrate the use of nested class.

```
#include<iostream>
using namespace std;
class student {
 public:
 int m1, m2;
 void accept(){
 cout<<"Enter marks for m1;"
   cin>>m1;
 cout<<"Enter marks for m2";
    cin>>m2;
   }
  void display (){
 cout<<"marks m1:"<<m1<<endl;
 cout<<" marks m2:" <<m2<<endl;
}
};
};
  int main (){
   student::marks m;
     m.accept();
     m.display ();

     return 0;
     }
```

12/8

Experiment 4

1) WAP to swap two no's from some class using objects as function arguement write swap func as member function.

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

class Number{
    int a,b;
public:
    void Data (int x, int y){
        a=x;
        b=y;
    }
    void display(){
    cout<<"a="<<a<<"b"<<b<<endl;
    }
void swap(Number &obj)
    int temp;

    temp = a;
    a = obj.a;
    obj.a = temp;

    temp = b;
    b = obj.b;
    obj.b = temp;
}
}
```

```
int main () {
Number obj 1, obj 2;
 obj 1 Data (10,20);
 obj 2 Data (30,40);

 cout<<"obj 1 :";
 obj 1. display () ;
 cout<<" obj 2 :";
  obj 2. display ();

 obj 1. swap (obj 2);

 Cout<<" After swap :", endl;
 rout<<"obj 1 :";
 Obj 1. display ();
 rout<<"obj 2.:";
 obj 2. display ();
 return 0;
 }
```

output

obj 1: a=10, b=20

obj 2: a=30, b=40

After swap

obj 1 : a=30, b=40

obj 2: a=10, b=20

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

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

class number{
 int value;
  public:
 void value(int v){
   value=v;
 }
 void display(){
  cout<<"value":<<value<<endl;
 }
friend void swap value(Number &N1, number &N2);
};
 void swapvalues(Number &N1, Number& N2){
    int temp= N1.value;
    N1.value = N2.value
    N2.value = temp;
 }
   int main (){
   Number A, B;
    A.value(10);
    B.value(20);

A display();
B display();
```

```
swap values (A,B);

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

O/P:

A=10

B=20

New values:

A=20

B=10

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

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

class classB;
class classA {
  int numA;
public:
  void value (int a){
    numA=a;
  }
  void display (){
  cout<<"class A value=" <<numA <<endl;
  }
  friend void swapvalues (classA&, classB&);
};

class classB {
  int num();
  public:
  void values (int b){
  numB = b;
  }
  void display (){
cout <<" class B value=" << numB <<endl;
}
friend void swap values (class A &, class B &);
};
```

```
void swapvalues(class A &a, class B &b
    int temp = a.numA;
    a.numA = b.numB;
    b.numB = temp;
}

int main (){
    class A A;
    class B B;

    A.value(20);
    B.value(30);
    cout << " values: " <<endl;
    A.display();
    B.display();
    Swapvalues (A, B);
    cout <<"swapped values: " <<endl;
    A.display();
    B.display();
    return 0;
}
```

4) WAP to create 2 classes Result, Result2 which stores the marks. Read values of both classes object & compute the avg of 2 results.

```
#include <iostream>
using namespace std;
class Result2;
class Result1;{
float marks1;
public:
void readmarks(){
cout<<"Enter marks for result1:";
cin>>marks1;
}
friend float computeAverage(Result1 r1, Result2 r2);
};
class Result2{
float marks2;
public:
void readmarks(){
cout<<"Enter marks for Result2:";
cin>>marks2;
}
friend float computeaverage(Result r1, Result2 r2);
};
return(r1.mark1 + r2.marks2)/2;
}
```

```
int main (){
 Result 1  A1;
 Result 2  A2;

 A1 . readmarks ();
 A2 . readmarks ();

 float avg = compute Average (A1, A2);
 cout << "\n Average of two results : " << avg << endl;
 return 0;
}
```

O/P :-

Enter marks for Result 1 : 10

Enter marks for Result 2 : 20

~~Average~~ Average of two results : 15

5) WAP to find the greatest number among two numbers from different classes using friend fun

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

```
void find greatest (class A a, class B b) {
 if (a.numA > b.numB) {
 cout<<"Greatest number is from class A"<<a.numA<<endl;

 }else if (b.numB > a.numA) {
 cout << "Greatest number is from class B"<<b.numB<<endl;

 }else {
 cout<<"Both are equal "<<endl;
 }
 }
 int main () {
 class A A;
 class B B;

 A.accept (35);
 B.accept (40);
 A.display ();
 B.display ();

 find greatest (A,B);
 return 0;
 }
```

O/P:-

Class A Number: 35

Class B Number: 40

Greatest number is from class B: 40

- Practice codes

)

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

class ClassB;
class ClassA {
private:
 int valueA;

public
 void input(){
cout<<"Enter value for class A:";
  cin >>valueA;
 }
 friend int sum(classA a, ClassB b);
};
class ClassB{
 private:
 int valueB;

public:
 void input(){
cout<<"Enter value for class B:";
  cin >>valueB;
 }};
```

```
friend int sum (class A a, class B b);
};
int sum (class A a, class B b){
 return a.valueA + b.valueB;
}
int main (){
class A obj A;
class B obj B;

obj A.input();
obj B.input();

cout<<"Sum:"<< sum(obj A, obj B) <<endl;
return 0;
}
```

2)

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

class Number {
private:
 int value;
public:
 void accept() {
  cin>>value;
 }
 void display() {
  cout<<value<<endl;
 }
 friend void swapNumbers (Number n1, Number n2);
};
void swapNumbers (Number n1, Number n2){
 int temp = n1.value;
 n1.value = n2.value;
 n2.value = temp ;
}
int main (){
Number num1, num2;

Cout<< "Enter value for num1:";
num1.accept();
```

```
cout<<"Enter value for num 2:";
num2.accept();

cout<<"\n Before swapping :"<<endl;
cout<<"num 1 = ";
num1.display();
cout<<"num 2 = ";
num2.display();

swapNumbers(num1, num2);
cout<<"\n After swapping:"<<endl;
cout<<"num 1 = ";
num1.display();
cout<<"num 2 = ";
num2.display();
return 0;
}
```

3)

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

class cube;
class box {
 private:
  int volume;
  public:
  void accept()
  {
cout<<"Enter volume of box:";
     cin>>volume
}
friend void findgreater(box, cube);
};
class cube {
private:
 int volume;
public:
void accept()
{
cout<<"Enter volume of cube:";
 cin >>volume;
}
friend void findgreater (box, cube);
};
void findgreater (box b, cube c)
{
```

```
if(b.volume> c.volume)
{
cout<<" Volume of box is greater";
}
else if(b.volume == c.volume)
{
(out<<" Volume of cube is greater");
}}
int main (){
box h;
cube g;
h.accept()
g.accept()
find greater(h,g);
return 0;
}
```

4)

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

class complex {
private:
float real;
float imag;

public:
void accept(){
cout<<"Enter real part:";
   cin >>real;
   cout<<"Enter imaginary part:";
    cin>>imag;
    }
    void display(){
    cout<<real <<" +"<<imag <<"i" <<endl;
    }
friend complex addComplex(complex c1, complex c2);
};
complex addComplex (complex c1, complex c2){
complex temp;
  temp.real = c1.real +c2.real
   temp.imag= c1.imag + c2.imag;
  return temp;
  }
```

```
int main (){
complex num1, num2, sum;

cout<<"Enter first complex number:"<<endl;
    num1.accept();

cout<<"\nEnter second complex number:"<<endl;
    num2.accept();

sum = addComplex(num1, num2);

cout<<"\nSum of complex numbers:";

    sum.display();
    return 0;
}
```

5)

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

class Student {
private:
string name;
float mark1, mark2, mark3;

public:
 void accept() {
 cout<< "Enter student name:";
 cin>>name;
 cout<<"Enter marks in 3 subjects:";
cin>>mark1>>mark2>>mark3; }
friend void calculateAverage(Students);
 };
void calculateAverage(Students s) {
float avg = (s.mark1+s.mark2+s.mark3)/3;
cout<<"\n Student name:"<<s.name<<endl;
cout<<"Average marks:"<<avg<<endl;
}
int main (){
Student stu;
 stu.accept();
calculateAverage(stu);
return 0;
}
```

6)

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

class Beta;
class Gamma;

class alpha {
private:
  int a;
public:
  void accept() {
  cout << " Enter value for alpha:";
    cin >> a;
  }
  friend void sum Values(Alpha, Beta, Gamma);
  };
class Beta {
  private :
  int b;
  public:
  void accept (){
   cout<<" Enter value for Beta:";
      cin >> b;
      }
friend void sum Values (Alpha, Beta, Gamma);
};
```

```
class Gamma {
  private:
    int c;
  public:
  void accept() {
  cout<<"Enter value for Gamma:";
    cin>>c;
  }
  friend void sumValues(Alpha, Beta, Gamma);
  };
void sumValues(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 A;
Beta obj B;
Gamma obj C;

objA.accept();
obj B. accept();
obj C. accept();

sumValues (objA, obj B, obj C);

return 0;
}
```

7)

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

class Point {
private:
float x;
float y;

public:
void accept(){
cout<<"Enter coordinates (xy): ";
cin >>x>>y;
}
friend float calcDistance (Point, Point);
};
float calcDistance (Point p1, point p2){
float dx = p2.x - p1.x;
float dy = p2.y - p1.y;
return sqrt(dx* dx+dy * dy);
}
int main (){
Point A, B;
A.accept ();
B.accept();
float distance = calc Distance (A, B);
cout<<"The distance between 2 pt's is:"<<distance<<en
return 0;
}
```

8)

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

class BankAccount;

class Audit {
 public:
 friend void
auditBalance(BankAccount);
};
class BankAccount{
 private:
 int balance;
 public:
BankAccount(int b){balance=b;}
 friend void
auditBalance(BankAccount);
};
void auditBalance(BankAccount acc){
cout<<"Balance:"<<acc.balance;
}
```

```
int main (){
BankAccount acc(5000);
auditbalance(acc)
 return 0;
 }.
```

Q1
16/9

Experiment 5

1) WAP to find sum of number between 1 to n with a constructor where 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;
}
```

o/p :-

sum : 55

b) WAP to declare class student, having data members name and %. Write a constructor to initialize these data members.

```
#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("Shreyash", 92.8);
s1.display();
return 0;
}
```

o/p :-

[redacted]

Name - Shreyash
Percentage - 92.8
Student Name - Shreyash
Student percentage : 92.8

c) Define a class college members variables as roll_no, name, course, use constructor with default value as "Computer Engineering" for course. Accept this data for two objects of class & display data.

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

class college {
int roll no;
string name;
string course;

public:
college(int r, string n, string c = Comp Sci
{
roll_no = r;
```

```
name = n;
course = c;
}
void disp()
{
cout<<"Roll No:"<<roll_no<<endl;
cout<<"Name:"<<name<<endl;
cout<<"Course:"<<course<<endl;
}
};
int main()
{
int r1, r2;
string n1, n2;

cout<<"Enter roll no & name first:";
cin>>r1>>r2

college c1(r1, n1);
college c2(r2, n2);

cout<<"Details of student:");
s1.display();
s2.display();
return 0;
}
```

4) WAP to demonstrate constructor overloading

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

class number {
int x, y;
public:
number(int a)
{
 x = a;
 y = 0;
}
Number(int a, int b)
 {
   x = a;
   y = b;
   }
void disp(){
cout<<"x"<<x<<"y="<<y<<endl;
}
};

int main(){
number n1(5);
number n2(10,20);
 n1.display()
 n2.display()
 return 0;
 }
```

6/10

Expt 6

a) WAP to implement multilevel inheritance

```
#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;
```

```
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;
}
```

b) WAP to implement multiple inheritance

```
#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 marks of student:"
cin>>marks;
cout<<"Enter the sports score of student
cin >> score;
}
void calculate()
{
total_score = marks+score;
```

```
cout<<"The marks of 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;
}
```

3) WAP to implement hierarchial inheritance

```
#include<iostream>
using namespace std;
class vertical
{
protected:
string brand;
int model;
};

class car:protected vehicle
{
protected:
string type;
};
class Electric car: protected car
{
int battery capacity;
public:
void accept()
{
cout<<" Enter brand of car:";
cin>>brand;
cout<<"Enter model of car:";
cin>>model;
cout<<" Enter type of car:";
cin>>type;
}
```

```
void disp()
{
cout<<"\n The brand of car: <<brand<<endl;
cout<<" The model of car:"<<model<<endl;
cout <<" The type of car:" <<type<<endl;

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

d) WAP to implement hybrid inheritance

```
#include<iostream>
#include<string>
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:
void showinternDetails()
{
cout<<"\n--Intern Details--\n";
student::showpersondetails();
showEmployeeDetails();
}
};
int main()
{
internit;
it.student:getpersondetails();
it.getdetails()
it.getemployeeDetails();
it.showinternDetails();
return 0;
}
```

8/10

Expt 7

a) WAP using function overloading

```
#include<iostream>
using namespace std;
class Area;
{
private:
float l, b;
public:
void area (float l)
  {
  float area;
  area = l * b
cout<<"Area of square:"<<area<<endl;
  }
void area (float l, float b)
  {
  float area;
  area = l * b;
cout<<"Area of rectangle:"<<area;
 }
};
int main ()
{
  Area a1;
  a1.area(8);
  a1.area (9,12);
  return 0;
 }
```

b) - using function overloading

```
#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 float no's: "<<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<<endl;
}
};
```

```
int main ()
{
    sum s1;
    s1.sum (1.2, 2.3, 3.4, 5.5, 4.4);
    s1.sum( 2, 1, 3, 4, 5, 6, 7, 8, 9, 10);
    return 0;
}
```

c) WAP to implement unary operator...

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

```
number n1
n1.accept();
 -n1;
n1.disp();
 return 0;
}
```

d) WAP to implement unary ++ operator

```
#include<iostream>
 using namespace std;
 class number
  {
  private: int x, int accent;
    public:void accent()
     {
  cout<<"Enter a no:";
    cin >> x;
    temp = x;
    }
    void operator++()
      {
      x = ++x;
    }
    void reset()
    {
    x = temp;
    }
    void operator++(int)
      {
      x = x++;
```

```
void disp ()
  {
  cout<<"(Pre) The no is:"<<x<<endl;
      }

void disp ()
  {
  cout<<"(Post) The number is:"<<x;
     }
     };
  int main ()
  {
  number n1;
   n1.accept();
    ++n1;
    n1.disp();
  n1.reset();
    n1++;
     n1.disp2();
    return 0;
   }
```

7611

Expt 8

a) WAP to overload '+' operator

```
#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 disp()
{
cout<<str<<endl;
}
};
int main()
{
mystring s1,s2,s3;
s1.str="xyz"
s2.str ="pqr";
s3 = s1+s2
cout<<"Concatenated string";
s3.disp();
return 0;
}
```

6

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

class membership login: public login
    {
string membership ID;
    public:
    void accept()
    override;
    {
    login: accept()
    override
    {
    Login: accept()
cout<<"Enter membership ID:";
    cin >> membershipID;
        }
    void display()
        {
    cout<<"\n - Membership login details -
            \n";
cout<<" Name: "<<name<<"\n password: "<<
password<<"\n Membership ID: "<<membership ID<<endl
    }
};
```

```
int main ()
{
Email login e;
membership login m;
  e.accept();
 m.accept();
  e.display();
  m.display();
   return 0;
   }
```

Q.

16/10

Experiment 9

* Creating / opening file using open() function.

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

int main () {
fstream new-file;
new_file.open("new file.txt", ios::out);
if (!new file) {
cout<<" File creation failed"<<endl;
}else{
cout<<" New file created"<<endl;
new file.close();
}
return 0;
}
```

copy content from one file to another

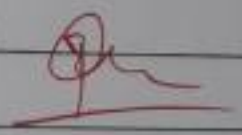
```
#include<iostream>
#include<fstream>
using namespace std;

int main(){
fstream *file("file1.txt",ios::app);
  if(!file1){
  cout<<"Error opening file.txt"<<endl;
  return 1;
  }
file1<<"welcome to MIT"<<endl;
  file1.close();

ifstream file1_read("file1.txt");
fstream file2("file2.txt");

  if(!file1_read){
  cout<<"Error opening file 1.txt for reading<<endl;

  return 1;
}
  if(!file2){
  cout<<"Error opening file2.txt for writing"<<endl;
  return 1;
  }

char ch;
while(file1_read.get(ch)){
```

```
file2.put(ch);
}
cout << "Content copied successfully from file1.txt
     to file2.txt" << endl;

file1.read.close();
file2.close();
return 0;
}
```

* Count * number of digits of spaces

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

int main () {
ifstream inputFile ("Input.txt");
if(!inputFile) {
cout<<"Cannot open input.txt" <<endl;

return 1;
}

char ch;
int spaceCount = 0;
   digitCount = 0;

while (inputFile.get (ch)) {
if (ch == ' ') or {isspace (ch)
   SpaceCount ++;
```

```
        else if (isdigit(ch));
                digitcount++;
    }
    inputfile.close();

    cout<<"Total spaces:"<<spacecount;
    cout<<"Total digits:"<<digitcount;

    return 0;
}
```



Experiment 10

Find sum of array using function template integer, float, double

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

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

int main(){
const int size=10;
int intArr[size]={1,2,3,4,5,6,7,8,9,10};

cout<<"sum of int array:"<<findsum(Arr,size)<<endl;

return 0;
}
```

```
float floatArr[size]={1.1f,2.2f,3.3f,4.4f,5.5f,6.6f,
                      7.7f,8.8f,9.9f,10.1f};
cout<<"Sum of float array:"<<findSum(float,size<<endl;
```

```
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 double array: "<<findSum(doubleArr,size)<<
                                                    endl;
```

2) Concatination of string

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

template<class T>
T square(T x){
    return x*x;
}

template<>
char* square<char*>(char* str){
    static char result[200];
    strcpy(result,str);
    strcat(result,str);
    return result;
}
int main(){
char str[100]="Hello world";
cout<<"Square of string: "<<square(str)<<endl;

return 0;
}
```

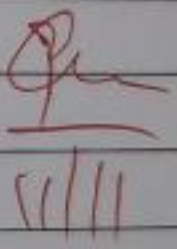
Calculator to perform 10 functions.

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

template <class T> class calculator
{
public:
T num1=0;
T num2=0;

void accept()
cout<<"Enter num1 and num2:"<<endl;

cin>>num1>>num2;
}
void add()
cout<<"Sum:"<<num1+num2;
}
void subtract(){
cout<<"Subtract:"<<num1-num2;
}
void mul(){
cout<<"Multiply:"<<num1*num2;
}
void div(){
cout<<"Division:"<<num1/num2;

void mod(){
cout<<"Modulus:"<<num1%num2;
}
```

```
void sine (){
cout<< "sine"<<sin(num1)<<sin(num2);
 }

void cosine() {
 cout<<"cosine : "<<cos(num1)<<cos(num2);
}
void Tan() {
cout<<"Tan: "<<tan(num1)<<tan(num2);
 }
void power() {
cout <<"power:"<<pow(num1, num2) <<endl;
 }

void squareroot() {
cout<<"Square Root: "<<sqrt(num1)<<sqrt(num2);
 }

void display () {
 cout<<"\n --Calculator Menu--"<<endl;
cout<<" 1. Addition"<<endl;
        2. Subtraction
        3. Multiplication
        4. Division
        6. Modulus
        6. Power
        7. Cosine
        8. Tangent
        9. Sine
        10. Square root
        }
};
```

```
int main (){
int choice;
Calculator <int>calc;

do {
calc.display();
cout<<"\n Enter your choice: ";
cin>>choice;

if (choice >=1&&choice <= 10)
calc.accept()

}
cout<<"Result"<<endl;

switch (choice){

case 1 : calcadd();
break;
case 2 : calc.subtract()
break;
case 3: calc. mul ();
break;
case 4: calc. div ();
break;
case 5: calc.mod();
break;
case 6 : calcpower ();
break;
```

```
case 7: calc.cosine()
        break;
case 8: calc.tane()
        break;
case 9: calc.sineb()
        break;
case 10: calc.squarroot();
        break;
default: cout<<"choice!>10 && choice!=0";
        return 0;
        }
```



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

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

template<class T> class stack{
 T arr[20];
  int top;

 public:
  stack(){
  top=-1;
  }

void push(T value){
  if(top==9)
  cout<<"Stack overflow!"<<endl;
 }else{
   top++;
   arr[top]=value;
cout<<"value pushed into stack"<<endl;
  }
  }
void pop(T value){
      if(top==-1)
cout<<"Stack underflow"<<endl;
      top--;
```

```
void display() {
if (top == -1)
cout<< "Stack is empty"<<endl;
else {
cout<< "Stack elements:";
for(int i=top; i>=0; i--)
cout<< "arr[i] "<< endl;
}
};

int main()
stack <int>s;
int choice, value;
{
cout<< "\n1.Push\n2.Pop\n3.Display
\n4.Exit\nEnter your choice:";
cin>>choice;

switch (choice) {
case 1: cin>>value;
s.push(value);
break;
case 2:
s.pop;
break;
case 3;
s.display();
break;
case 4:
cout<<Exit<<endl;
break;
} while (choice!=0); return 0; }
```

Experiment 11

Create a vector
1) Modify
2) Multiply by scalar
3) Display

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

template <class T>class vector{
    vector<T> v;
  public:
  void createvector(int n){
  cout<<"Enter "<<n<<" elements:\n";
    v.resize(n);
    for(int i=0;i<n;i++)
      cin>>v[i];
  }
void modify(int index, T value){
  if(index>=0 && index<v.size())
    v[index]=value;
  else
  cout<<"Invalid!\n";
  }
void multiply by scalar(T scalar){
  for(int i=0; i<v.size(); i++)
    v[i]=scalar;
  }
```

```
void display () {
    cout<<" (";
for (int i= 0; i< v.size(); i++) {
    cout<< v[i];
    if (i != v.size() - 1)
        cout<<" , ";
}
cout<<" )\n";
}
};

int main () {
vector <int> vec;
int n, index, newValue, scalar;

cout<< "Enter size of vector ";
cin>>n;
vec.createvector(n);

cout<<"Original vector:";
    vec.display();

cout<<"\nEnter index & newvalue to
                modify :";
cin >>index>> newvalue;
vec.modify(index, newvalue);

cout<<"After modification:";
    vec.display ();
```

```
cout<<"\n Enter scalar to multiply:";
cin>>scalar
vec.multiply(scalar);

cout<<"After multiplication:";
vec.display();
return 0;
}
```

using iterator

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

int main (){
vector<int> v= {2,4,6,8,10,12,14,16,18,20};

cout<<"Initial vector :"<<endl;
for(vector<int>::iterator it=v.begin();
    it!=v.end();it++){
    cout<<*it<<" "<<endl;
}
for(vector<int>::iterator it=v.begin();
    it!=v.end();it++){
    *it=(*it)*10;
}
cout<<"New vector :"<<endl;
for(vector<int>::iterator it=v.begin();
    it!=v.end();++it){
    cout<<*it<<" "<<endl;
}

return 0;
}
```

Expt 12

a) WAP using STL to implement data.

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

int main(){
stack<int> s;

cout<<"stack";

s.push(10);
s.push(20);
s.push(30);

cout<< pop() <<endl;
cout << "Display elements.";
stack<int>temp=s;
while (!temp.empty()){
    cout<<temp.top()<<" ";
    temp.pop();
}
cout<<endl;

cout<<top element:<<s.top();

return 0;
}
```

WAP using STL to implement Queue.

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

int main(){
  queue<int>q;

  q.push(10);
  q.push(25);
  q.push(45);

cout<<"Queue elements: \n"<<endl;
   queue<int> temp=q;
  while (!temp.empty()){
    cout<<temp.front()<<" ";
     temp.pop();
  }
  cout<<endl;

  return 0;
  }
```