OBJECT ORIENTED PROGRAMMING LANGUAGE”

Submitted for the degree of bachelor of technology (b.tech) in computer science & engineering

SUBMITTED TO:

MRS.PRIYANKA KAMBOJ

ASST PROF

SUBMITTED BY:

MANU PAL

8520123

2ND YEAR (CSE)

Department of computer science & engineering

JAI PARKASH MUKAND LAL INNOVATIVE ENGINEERING &TECHNOLOGY INSTITUTE

RADAUR-135133(YAMUNA NAGAR)

INDEX….

16. Program to overload incremental and decremental operator using

(a)member function

(b) friend function

17.Program to overload any binary operator using

(a) member function

(b) friend function

18.Program to overload << and >> operator

19.Program to overload = and [] operator

20.Program to compare two string using operator overloading

21.Program to illustrate Dynamic binding(virtual function)

22.Program to illustrate virtual destructor and pure virtual destructor

23.Program to illustrate *this pointer

24.Program to illustrate exception handling

25.Program to illustrate Template class

26.Program to illustrate Template function

27.Program to overload Template

28.Program to write and find the total number of character in a File

29.Program to copy a file in to another File

30.Program to illustrate Nested class

PROGRAM NO-1

program: To find factorial of a number in c++

```
#include <iostream>
using namespace std;
int main()
{
   int i,fact=1,num;
   cout<<"Enter a Number: ";
   cin>>num;
   for(i=1;i<=num;i++)
   {
      fact=fact*i;
   }
            cout<<"Factorial number of " <<num<<" is: "<<fact<<endl;
            return 0;
}
```

output

```
Enter a Number: 6
Factorial number of 6 is: 720
```

PROGRAM NO-2

Program: Program to illustrate namespace

```
#include <iostream>

using namespace std;

namespace First {

    void sayHello()

     {

         cout<<"Hello First "<<endl;

     }

}

namespace Second

 {

       void sayHello()

        {

           cout<<"Hello second"<<endl;

        }

}

int main()

{

    First::sayHello();

    Second::sayHello();

 return 0;

}
```

Output

```
Hello First

Hello second
```

PROGRAM NO-3

Program: program to illustrate c++ structure

```
#include <iostream>

using namespace std;

struct Person

{

   char name[50];

   int age;

   float salary;

};

int main()

{

   Person p1;


   cout << "Enter Full name: ";

   cin.get(p1.name, 50);

   cout << "Enter age: ";

   cin >> p1.age;

   cout << "Enter salary: ";

   cin >> p1.salary;


   cout << "\n Display Information" << endl;

   cout << "Name: " << p1.name << endl;

   cout <<"Age: " << p1.age << endl;

   cout << "Salary: " << p1.salary;

   return 0;
}
```

Output

```
Enter Full name: ABC

Enter age: 23

Enter salary: 67980


 Display Information

Name: ABC

Age: 23

Salary: 67980
```

PROGRAM NO-4

Program: Program to illustrate

(a) call by value

(b)call by Reference

(c) call by address

```
#include<iostream>

using namespace std;

//call by value

swap1(int x,int y)

{

    int a;

    a=x;

    x=y;

    y=a;

}
```

```
//call by refrence

swap2(int *x,int *y)

{

    int a;

    a=*x;

    *x=*y;

    *y=a;

}

//call by address

swap3(int &x,int &y)

{

    int a;

    a=x;

    x=y;

    y=a;

}

int main()

{

    int x=2,y=3;

    swap1(x,y);

    cout<<"After call by value :"<<x<<" "<<y<<endl;

    swap2(&x,&y);

    cout<<"After call by refrence :"<<x<<" "<<y<<endl;

    swap3(x,y);

    cout<<"After call by address :"<<x<<" "<<y<<endl;

}
```

Output

```
After call by value :2 3

After call by refrence :3 2

After call by address :2 3
```

PROGRAM NO-5

Program: Program to illustrate inline function

```
#include <iostream>

using namespace std;

inline int add(int a, int b)

{

   return(a+b);

}

int main()

{


   cout<<"Addition of 'a' and 'b' is:"<< add(4,8);

   return 0;


}
```

Output

```
Addition of 'a' and 'b' is:12
```

PROGRAM NO-6

Program: Program to find sum of Number using default argument

```
#include <iostream>

using namespace std;

int sum(int x, int y, int z = 0)

{

        return (x + y + z );

}

int main()

{

        cout <<"Sum is="<< sum(10, 45) << endl;

        cout <<"Sum is="<< sum(34, 15, 25) << endl;

        return 0;

}
```

Output

```
Sum is=55

Sum is=74
```

PROGRAM NO-7

Program: program to illustrate function overloading

```
#include <iostream>

using namespace std;

void print(int i)

{

    cout << "Interger value is=" << i << endl;

}

void print(double f)

{

    cout << " Floating value is= " << f << endl;

}

void print(char const *c)

{

    cout << "character is= " << c << endl;

}

int main() {

    print(106);

    print(23.56);

    print("Hello");

    return 0;

}
```

Output

```
        Interger value is=106

        Floating value is= 23.56

        character is= Hello
```

PROGRAM NO-8

program: Program to illustrate friend function

(a) friend function

(b) friend member function

(c) friend class

```
#include<iostream>

using namespace std;

class derived;

class demo

{

   public:

   int x;

   demo(int a)

   {

      x=a;

   }


   friend class derived;

   friend void show(demo);

};

class derived

{

   public:

    display(demo &o3)

   {

      cout<<"x :"<<o3.x<<endl;
    }
};
void show(demo o1)
{
   cout<<"x :"<<o1.x<<endl;
}
int main()
{
    demo o1(3);
    derived o2;
    cout<<"Using Friend class :"<<endl;
    o2.display(o1);
    cout<<"Using Friend function :"<<endl;
    show(o1);
}
```

Output

```
Using Friend class :
x :3
Using Friend function :
x :3
```

PROGRAM NO-9

Program to find sum of complex number using constructor overloading (parameterized,copy constructor and default constructor)

```
using namespace std;

class demo

{

private:

   int imag,real;

public:

   demo(int x=0,int y=0)

   {

      imag=x;

      real=y;

   }

   demo(demo &obj1)

   {

      real=obj1.real;

      imag=obj1.imag;

   }

   void sum()

   {

      cout<<"Complex number :"<<real<<"+i"<<imag<<endl;

   }

};

int main()
```

```
{
   demo o1;
   demo o2(2,3);
   demo o3(o2);
   cout<<"complex number using parametrized constructor"<<endl;
   o2.sum();
   cout<<"complex number using copy constructor"<<endl;
   o3.sum();
}
```

Output

```
complex number using parametrized constructor
Complex number :3+i2
complex number using copy constructor
Complex number :3+i2
```

PROGRAM NO-10

Program to illustrate constructor and destructor

```
#include <iostream>
using namespace std;
class print
{
public:
        // constructor
        print()
        {
                cout<<"Constructor called"<<endl;
        }
        // destructor
        ~print()
        {
                cout<<"Destructor called"<<endl;
        }
};
int main()
{
        print obj1;
       return 0;
}
```

Output

```
Constructor called
Destructor called
```

PROGRAM NO-11

Program to illustrate static members (static data member and static member function)

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

class Box {
  public:
    static int objectCount;

    // Constructor definition
    Box(double l = 2.0, double b = 2.0, double h = 2.0) {
      cout <<"Constructor called." << endl;
      length = l;
      breadth = b;
      height = h;

      // Increase every time object is created
      objectCount++;
    }
    double Volume() {
      return length * breadth * height;
    }
    static int getCount() {
      return objectCount;
      }

   private:
      double length;     // Length of a box
      double breadth;    // Breadth of a box
      double height;     // Height of a box
};

// Initialize static member of class Box
int Box::objectCount = 0;

int main(void) {
   // Print total number of objects before creating object.
   cout << "Inital Stage Count: " << Box::getCount() << endl;

   Box Box1(3.3, 1.2, 1.5);    // Declare box1
   Box Box2(8.5, 6.0, 2.0);    // Declare box2

   // Print total number of objects after creating object.
   cout << "Final Stage Count: " << Box::getCount() << endl;

   return 0;
}
```

Output

```
Inital Stage Count: 0
Constructor called.
Constructor called.
Final Stage Count: 2
```

PROGRAM NO-12

Program to find the result of a student using hybrid inheritance.

```
#include<iostream>
using namespace std;
class student
{
   protected:
   int roll_no;
   public:
   void getdata1(int i)
   {
      roll_no=i;
   }
};
class marks : public student
{
   protected:
   int num1,num2;
   public:
   void getdata2(int i, int j)
   {
      num1=i;
```

```
        num2=j;
    }
};
class result : public marks
{
    int r;
    public:
    void total()
    {
        r=num1+num2;
    }
    void display()
    {
        cout<<"Roll no:"<< roll_no << endl;
        cout<<"Marks1="<< num1 << endl;
        cout<<"Marks2="<< num2 << endl;
        cout<<"Total marks="<< r <<endl;
    }
};
 main()
{
    result obj;
    obj.getdata1(101);
    obj.getdata2(45,65);
    obj.total();
    obj.display();
```

```
}
```

Output

```
Roll no:101
Marks1=45
Marks2=65
Total marks=110
```

PROGRAM NO-13

Program to create basic calculator using hierarchical inheritance.

```
#include<iostream>
using namespace std;
class base
{
   protected:
   int x,y;
   public:
   void getdata() {
       cout<<"Enter value of x and y:\n"<< endl;
       cin >> x >> y;
   }
};
class derived1 : public base
{
   public:

   int sum()

   {

      cout<<"\nSum of="<< x+y;

   }

   int sub()

   {

      cout<<"\n substraction of="<< x-y << endl;

   }

};

class derived2 : public base

{

   public:

   int mul()

   {

      cout<<"\n product of="<< x*y;

   }

   int div()

   {

        cout<<"\n Division of="<< x/y;

   }

};

int main()

{

   derived1 obj1;

   derived2 obj2;
   obj1.getdata();

   obj1.sum();

   obj1.sub();

   obj2.getdata();

   obj2.mul();

   obj2.div();

   return 0;

}
```

Output

```
Enter value of x and y:

12

2

Sum of=14

substraction of=10

Enter value of x and y:

12

4

 product of=48

 Division of=3
```

PROGRAM NO-14

Program to illustrate virtual class

```
#include <iostream>

using namespace std;

class A
```

```
 {
        public:
        int a;
        A(){
    a = 10;
  }
};
class B : public virtual A {
};
class C : public virtual A {
};
class D : public B, public C {
};
int main(){
  //creating class D object
  D object;
  cout << " value of a = " << object.a << endl;
  return 0;
}
```

Output

```
value of a = 10
```

```
}
```

PROGRAM NO-15

Program to find of sum of two complex Number using abstract class.

```
using namespace std;


class Complex {


   public:
      int real, imaginary;
   Complex(int tempReal = 0, int tempImaginary = 0)
   {
      real = tempReal;
      imaginary = tempImaginary;
   }
   Complex addComp(Complex C1, Complex C2)
   {
      Complex temp;
      temp.real = C1.real + C2.real;
      temp.imaginary = C1.imaginary + C2.imaginary;
      return temp;
   }
};
int main()
```

```
{
   Complex C1(3, 2);
   cout<<"Complex number 1 : "<< C1.real<< " + i"<<
C1.imaginary<<endl;
   Complex C2(9, 5);
   cout<<"Complex number 2 : "<< C2.real<< " + i"<<
C2.imaginary<<endl;
   Complex C3;
   C3 = C3.addComp(C1, C2);
   cout<<"Sum of complex number : "<< C3.real << " + i"<<
C3.imaginary;
}
```

Output

```
Complex number 1 : 3 + i2
Complex number 2 : 9 + i5
Sum of complex number : 12 + i7
```

PROGRAM NO-16

Program to overload incremental and decremental operator using

(a)member function

(b) friend function

```
#include<iostream>
using namespace std;
class demo
{
```

```
public:
    int x;
    demo() {}
    demo(int x1)    {x=x1;}
    void operator ++()
    {
        x++;
    }
    void operator --()
    {
        x--;
    }
    void show()
    {
        cout<<"x :"<<x<<endl;
    }
    friend operator +(demo &);
    friend operator -(demo &);
};
operator +(demo& o1)
{
    o1.x=o1.x+1;
}
 operator -(demo& o2)
{
    o2.x--;
```

```
}
int main()
{
   demo obj1(2);
   cout<<"Starting value of obj1"<<endl;
   obj1.show();
   ++obj1;
   cout<<"Incrementing value using member function of obj1"<<endl;
   obj1.show();
   --obj1;
   cout<<"Decrementing value using member function of
obj1"<<endl;
   obj1.show();
   demo obj2(10);
   cout<<"Starting value of obj2"<<endl;
   obj2.show();
    +(obj2);
   cout<<"Incrementing value using Friend function of obj2"<<endl;
   obj2.show();
   -(obj2);
   cout<<"Decrementing value using Friend function of obj2"<<endl;
   obj2.show();
}
```

Output

```
Starting value of obj1
x :2
Incrementing value using member function of obj1
```

```
x :3
Decrementing value using member function of obj1
x :2
Starting value of obj2
x :10
Incrementing value using Friend function of obj2
x :11
Decrementing value using Friend function of obj2
x
```

PROGRAM NO-17

Program to overload any binary operator using

(a) member function

(b) friend function

```
#include <iostream>
using namespace std;
class Arith_num
{
    int x, y;
    public:
        void input()
        {
            cout << " Enter the first number: ";
            cin >> x;
        }
        void input2()
    {
      cout << " Enter the second number: ";
      cin >> y;
    }
    Arith_num operator + (Arith_num &ob)
    {
      Arith_num A;
      A.x = x + ob.x;
      return (A);
    }
    void print()
    {
      cout << "The sum of two numbers is: " <<x;
    }
};
int main ()
{
 Arith_num x1, y1, res;
  x1.input();
  y1.input();
  res = x1 + y1;
  res.print();
  return 0;
}
```

Output

```
Enter the first number: 5
Enter the second number: 6

The sum of two numbers is: 11
```

PROGRAM NO-18

Program to overload << and >> operator

```
#include<iostream>

#include<conio.h>

using namespace std;

class demo

{

   public:

   int x,y;

   demo(){}

   demo(int x1,int y1)

   {

      x=x1;

      y=y1;

   }

   friend ostream & operator<<( ostream &out, demo&obj );

   friend istream & operator>>( istream &in, demo&obj );

};

ostream & operator << (ostream &out, demo&obj)

{

   out<<obj.x<<endl;
```

```
    out<<obj.y<<endl;
    return out;
}
istream & operator >> (istream &in,   demo&obj)
{
    cout<<"Enter first number";
    in>>obj.x;
    cout<<"Enter second number";
    in>>obj.y;
    return in;
}
int main()
{
    demo obj2;
    cin>>obj2;
    cout<<obj2;
    return 0;
}
```

Output

```
Enter first number 12
Enter second number 34
12
34
```

PROGRAM NO-19

Program to overload = and [] operator

```
#include<iostream>
using namespace std;
class greater1
{
   int x;
public:
greater1() {}
greater1(int i)
{
        x = i;
}
void display()
{
 cout << x << endl;
}
 void operator = (greater1 i)
{
    x = i.x;
    cout << x << endl;
}
};

int main()
```

```
{
   greater1 O1(2), O2;
   O2 = O1; // O2.operator=(O1);
   return 0;
}
```

Output

```
2
```

PROGRAM NO-20

Program to compare two string using operator overloading

```
#include<iostream>
#include<stdio.h>
#include<string.h>
using namespace std;
class String
{
      char str[20];
      public:
        void getdata()
      {
          gets(str);
      }
       int operator ==(String s)
      {
            if(!strcmp(str,s.str))
             return 1;
```

```
                return 0;
        }
};
int main()
{
        String s1,s2;

        cout<<"Enter first string :: ";
        s1.getdata();
        cout<<"\nEnter second string :: ";
        s2.getdata();
        if(s1==s2)
        {
            cout<<"\nStrigs are Equal\n";
        }
        else
        {
            cout<<"\nStrings are Not Equal\n";
        }
        return 0;
}
```

Output

```
Enter first string ::  hello world
Enter second string :: Hello World
Strings are Not Equal
```

PROGRAM NO-21

Program to illustrate Dynamic binding(virtual function)

```
#include<iostream>

using namespace std;

class base {

public:

    void print()

    {

        cout << "print base class\n";

    }

     virtual void show()

    {

        cout << "show base class\n";

    }

};

class derived : public base

{


    void print()

    {

        cout << "print derived class\n";

    }

    void show()

    {

        cout << "show derived class\n";
    }
};

int main()
{
    base *bptr;
    derived d;
    bptr = &d;

    // Virtual function, binded at runtime
    bptr->print();

    // Non-virtual function, binded at compile time
    bptr->show();
    return 0;
}
```

Output

```
print base class
show derived class
```

PROGRAM NO-22

Program to illustrate virtual destructor and pure virtual destructor

(a) Virtual destructor

```
include <iostream>

using namespace std;

class base {

public:

         base()

         {

        cout << "Constructing base\n";

    }

         ~base()

         {

        cout<< "Destructing base\n";

    }

};

class derived: public base {

public:

         derived()

         {

         cout << "Constructing derived\n";

    }

         ~derived()

         {
```

```
        cout << "Destructing derived\n";
    }
};

int main()
{
derived *d = new derived();
base *b = d;
delete b;
getchar();
return 0;
}
```

Output

```
Constructing base
Constructing derived
Destructing base
```

PROGRAM NO-23

Program to illustrate *this pointer

```
#include<iostream>
using namespace std;
class Test
{
private:
  int x;
public:
  void setX (int x)
  {
     // The 'this' pointer is used to retrieve the object's x
     // hidden by the local variable 'x'
     this->x = x;
  }
  void print() { cout << "x = " << x << endl; }
};
int main() {
   Test obj;
   int x = 20;
  obj.setX(x);
  obj.print();
  return 0;
}
```

Output

```
x = 20
```

PROGRAM NO-24

Program to illustrate exception handling

```
#include <iostream>
using namespace std;
int main()
{
int x = -1;
   cout << "Before try \n";
try {
        cout << "Inside try \n";
        if (x < 0) {
                throw x;
                cout << "After throw (Never executed) \n";
        }
}
catch (int x ) {
        cout << "Exception Caught \n";
}
    cout << "After catch (Will be executed) \n";
    return 0;
}
```

Output

```
Before try
Inside try
Exception Caught
After catch (Will be executed)
```

PROGRAM NO-25

Program to illustrate Template class

```
#include <iostream>
using namespace std;
template<class T>
class A
{
    public:
```

```
    T num1 = 13;

    T num2 = 6;

    void add()

    {

        std::cout << "Addition of num1 and num2 : " <<
num1+num2<<std::endl;

    }

};

int main()

{

    A<int> d;

    d.add();

    return 0;

}
```

Output

```
Addition of num1 and num2 : 19
```

PROGRAM NO-26

Program to illustrate template function

```
#include <iostream>

using namespace std;

template<class T> T add(T &a,T &b)

{

    T result = a+b;

    return result;
}
int main()
{
  int i =6;
  int j =3;
  float m = 2.3;
  float n = 1.2;
  cout<<"Addition of i and j is :"<<add(i,j);
  cout<<'\n';
  cout<<"Addition of m and n is :"<<add(m,n);
  return 0;
}
```

Output

```
Addition of i and j is :80
Addition of m and n is :3.5
```

PROGRAM NO-27

Program to overload Template

```
#include<iostream>

using namespace std;

template <class T>

void display(T t1)

{

        cout << "Displaying Template: "

                << t1 << "\n";

}

void display(int t1)

{

        cout << "Explicitly display: "

                << t1 << "\n";

}

// Driver Code

int main() {

        display(200);

        display(12.40);

        display('Hello');

        return 0;

}
```

Output

```
Explicitly display: 200

Displaying Template: 12.4

Explicitly display: 1701604463
```

PROGRAM NO-28

Program to write and find the total number of character in a File.

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
	ifstream  fin("read.txt");
	char ch;
	int i, c=0, sp=0;
	while(fin)
	{
		fin.get(ch);
		i=ch;
		if((i > 63 && i < 91) || (i > 96 && i < 123))
			c++;
		else
			if(ch== ' ')
				sp++;
	}
	cout<<"\n No. of Characters in a File : "<<c;
	cout<<"\n Space between the Words     : "<<sp;
	return 0;
}
```

output

PROGRAM NO -29

Program to copy a file in to another File

```
#include<iostream>
using namespace std;
int main()
{
   char ch, sourceFile[20], targetFile[20];
   FILE *fs, *ft;
   cout<<"Enter the Name of Source File: ";
   cin>>sourceFile;
   fs = fopen(sourceFile, "r");
   if(fs == NULL)
   {
      cout<<"\nError Occurred!";
      return 0;
   }
   cout<<"\nEnter the Name of Target File: ";
   cin>>targetFile;
   ft = fopen(targetFile, "w");
   if(ft == NULL)
   {
      cout<<"\nError Occurred!";
      return 0;
   }
   ch = fgetc(fs);
   while(ch != EOF)
   {
       fputc(ch, ft);
       ch = fgetc(fs);
   }
   cout<<"\nFile copied successfully.";
   fclose(fs);
   fclose(ft);
   cout<<endl;
   return 0;
}
```

Output

```
 Enter the Name of Source File: mayank.txt
Enter the Name of Target File: ..txt
File copied successfully.
```

PROGRAM NO -30

Program to illustrate Nested class

```
#include<iostream;
using namespace std;
class enclose
{
private:
     int x;
     class nest
     {
private :
      int y;
public:
      int z;
void prn()
{
      y=3;z=2;
     cout<<"\n The product of"<<y<<'*'<<z<<"= "<<y*z<<"\n";
}
}; //inner class definition over
nest n1;
public:
nest n2;
void square()
{
n2.prn(); //inner class member function is called by its object x=2;
      n2.z=4;
      cout<<"\n The product of " <<n2.z<<'*'<<n2.z<<"=
"<<n2.z*n2.z<<"\n";
      cout<<"\n The product of " <<x<<'*'<<x<<"= "<<x*x; }
};   //outer class definition over
int main()
{
enclose e;
      e.square();  //outer class member function is called
}
```

Output

```
The product of3*2=6
```

The product of 4*4= 16

The product of 8*8= 64