Object Oriented Programming

Practical File

Submitted to:
Dr. Charu Gupta
Assistant Professor
Department of Computer Science and Engineering

Submitted by:
Omansh Dhawan
CSE-B
20220802719

INDEX

Exp -1

Introduction to C++ :

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

C++ is an object-oriented programming language.

- Everything in C++ is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an **object**. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.

- Attributes and methods are basically **variables** and **functions** that belongs to the class. These are often referred to as "class members".

- A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.

```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
```

Abstraction

Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Consider a real life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car.

Encapsulation

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, you must declare class variables/attributes as private (cannot be accessed from outside the class). If you want others to read or modify the value of a private member, you can provide public **get** and **set** methods.

- It is considered good practice to declare your class attributes as private (as often as you can). Encapsulation ensures better control of your data, because you (or others) can change one part of the code without affecting other parts
- Increased security of data

Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Like we specified in the previous chapter; Inheritance lets us inherit attributes and methods from another class. **Polymorphism** uses those methods to perform different tasks. This allows us to perform a single action in different ways.

For example, think of a base class called Animal that has a method called animalSound(). Derived classes of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.)

Inheritance

In C++, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- **derived class** (child) - the class that inherits from another class
- **base class** (parent) - the class being inherited from

To inherit from a class, use the : symbol.

```
// Base class
class Vehicle {
  public:
    string brand = "Ford";
    void honk() {
      cout << "Tuut, tuut! \n" ; }
};
// Derived class
class Car: public Vehicle {
  public:
    string model = "Mustang";
};
```

Difference between :

A) C and C++

B) Linker and Loader

A)

SN0.	C	C++
1.	C was developed by Dennis Ritchie in around 1969 at AT&T Bell Labs.	C++ was developed by Bjarne Stroustrup in 1979.
2.	C is procedural programming.	On the other hand, C++ supports both procedural and object-oriented programming paradigms.
3.	As C does not support the OOPs concept so it has no support for polymorphism, encapsulation, and inheritance.	C++ has support for polymorphism, encapsulation, and inheritance as it is being an object-oriented programming language
4.	As C does not support encapsulation so data behave as a free entity and can be manipulated by outside code.	On another hand in the case of C++ encapsulation hides the data to ensure that data structures and operators are used as intended.
5.	C does not support function and operator overloading also do not have namespace feature and reference variable functionality.	On the other hand, C++ supports both function and operator overloading also have namespace feature and reference variable functionality.

B)

SN0.	LINKER	LOADER
1.	The main function of Linker is to generate executable files.	Whereas main objective of Loader is to load executable files to main memory.
2.	The linker takes input of object code generated by compiler/assembler.	And the loader takes input of executable files generated by linker.
3.	Linking can be defined as process of combining various pieces of codes and source code to obtain executable code.	Loading can be defined as process of loading executable codes to main memory for further execution.
4.	Another use of linker is to combine all object modules.	It helps in allocating the address to executable codes/files.
5.	Linker is also responsible for arranging objects in program's address space.	Loader is also responsible for adjusting references which are used within the program.

Experiment-2

Objective : Given a polynomial of the form $c_nx^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_1x + c_0$ and a value of x, find the value of polynomial for a given value of x. Here c_n, c_{n-1}, .. are integers (may be negative) and n is a positive integer.

Algorithm :

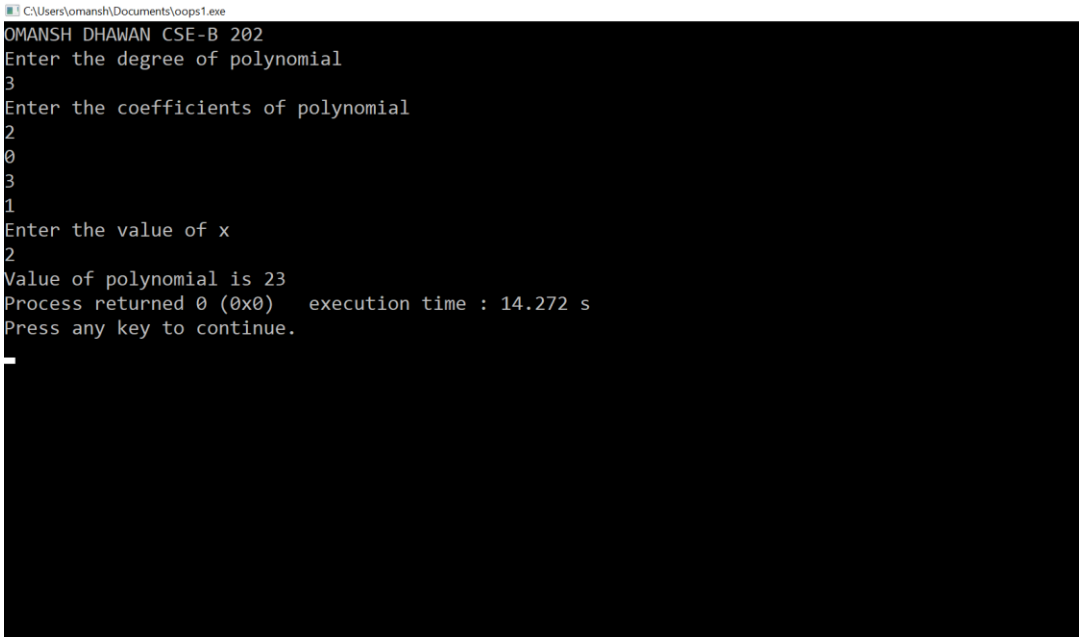
1. Input d,v.
2. Create an array poly of size d+1.
3. Result=poly[0]
4. Start a loop from I=0 to I=d+1`
5. Result=result*x+poly[I]
6. End of loop
7. Return result

CODE:

```
#include <iostream>

using namespace std;


int horner(int poly[], int n, int x)

{

int result = poly[0];

for (int i=1; i<n; i++)

result = result*x + poly[i];

return result;

}


int main()

{   int d,x;

cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
```

```
cout<<"Enter the degree of polynomial"<<endl;
cin>>d;
int poly[d+1];
cout<<"Enter the coefficients of polynomial"<<endl;
for(int i=0;i<=d;i++){
cin>>poly[i];
}
cout<<"Enter the value of x"<<endl;
cin>>x;
cout << "Value of polynomial is " << horner(poly, d+1, x);
return 0;
}
```

OUTPUT:



Experiment-3 Debugging And Programming Exercise

Debugging:

3.1)

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4. int i=0;
5. i=i+1;
6. cout<< i << " ";
7. /*comment\*//i=i+1;    (7 error : '/' should be removed)
8. cout<< i;
9. }
```

3.2)

```
1. #include<iostream>
2. using namespace std;
3. int main()
4. {
5. short i=2500,j=30000;
6. cout>>"i+j="">>-(i+j);   (6 error : '>>' should be replaced by '<<')
7. }
```

3.3)

```
1. #include<iostream>
2. using namespace std;
3. int main(){
4. int i=10,j=5;
5. int modResult =0;
6. int divResult =0;
7. modResult = i%j;
8. cout<<modResult<<" ";
9. divResult = i/modResult;
10. cout<<divResult;
11. }
```

```
0
Process returned -1073741676 (0xC0000094)   execution time : 0.269 s
Press any key to continue.
```

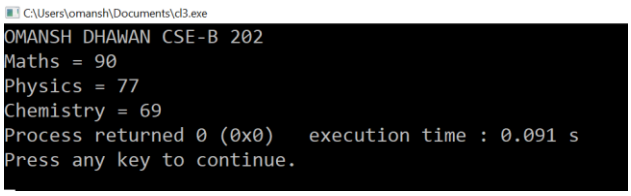
3.4)

1. cout<<"x="x; (corrected statement : cout<<”x=”<<x;)
2. m=5;//n=10;//s=m+n; (corrected statement : int 5; //n=10;//s=m+n;)
3. cin>>x;>>y; ; (corrected statement : cin>>x>>y;)
4. cout<<\n"Name:"<<name; (corrected statement : cout<<”\nName:"<<name;)
5. cout<<"Enter value:";cin>>x; (cin>>x; sho;uld be in next line)
6. /*Addition*/z=x+y; (“z” should have a datatype)

Programming Exercises

3.1)

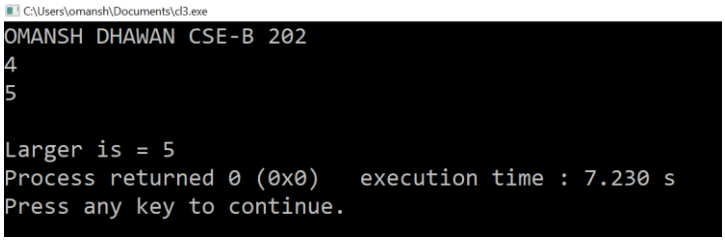
```
#include<iostream>
using namespace std;
int main()
{
system("cls");
cout<<"OMANSH DHAWAN CSE-B 202";
cout<<"\nMaths = 90\n"<<"Physics = 77\n"<<"Chemistry = 69";
return 0;
}
```



3.2)

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

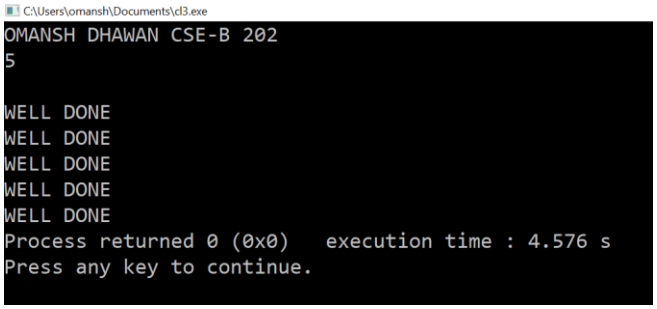
```
int main()
{
system("cls");
cout<<"OMANSH DHAWAN CSE-B 202";
int a,b;
cin>>a>>b;
if(a>b){
   cout<<"\nLarger is = "<<a;
}else{
   cout<<"\nLarger is = "<<b;
}
return 0;
}
```



3.3)

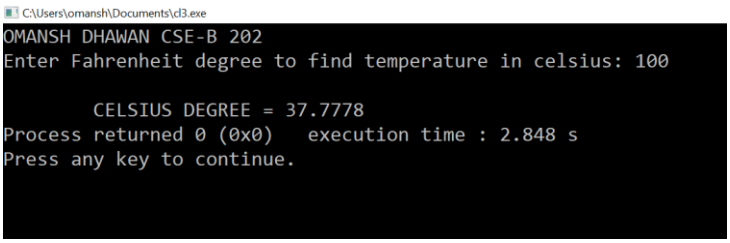
```
#include<iostream>
using namespace std;
int main()
{
system("cls");
cout<<"OMANSH DHAWAN CSE-B 202";
int a;
cin>>a;
for(int i=0;i<a;i++){
   cout<<"\nWELL DONE";
}
```

```
return 0;
}
```



3.5)

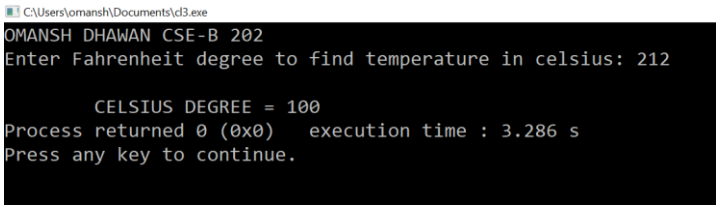
```
#include<iostream>
using namespace std;
int main()
{
float f,c;
system("cls");
cout<<"OMANSH DHAWAN CSE-B 202";
cout<<"\nEnter Fahrenheit degree to find temperature in celsius: ";
cin>>f;
c = (f-32)/1.8;
cout<<"\n\tCELSIUS DEGREE = "<<c;
return 0;
}
```



3.6)

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

```
class temp{
private:
   float f;
   float c;

public:
   void calTemp(){
      cout<<"\nEnter Fahrenheit degree to find temperature in celsius: ";
      cin>>f;
      c = (f-32)/1.8;
      cout<<"\n\tCELSIUS DEGREE = "<<c;
  }
};
int main()
{
class temp t;
system("cls");
cout<<"OMANSH DHAWAN CSE-B 202";
t.calTemp();
return 0;
}
```



Experiment – 4

a) Write a program to implement linear search.

Source code :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int lsearch(int[],int,int);
void main()
{
 int arr[10];
 system("cls");
 int s,ch;
 printf("@OMANSH DHAWAN 202 CSE-B\n");
 printf("enter the number of elements in an array:\n");
 scanf("%d",&s);
 printf("enter %d elements in an array:\n",s);
 for(int i=0;i<s;i++)
 scanf("%d",&arr[i]);
    int n;
            printf("enter the element to be found:");
            scanf("%d",&n);
            int a;
            a=lsearch(arr,n,s);
            if (a!=-1)
            printf("element found at index %d\n",a);
            else
            printf("element not found\n");
 getch();     }
 int lsearch(int arr[],int x,int s)
 {   for(int i=0;i<s;i++)
```

```
{  if(arr[i]==x)
   return i; }
   return -1; }
```

Output :

```
@OMANSH DHAWAN 202 CSE-B
enter the number of elements in an array:
4
enter 4 elements in an array:
1
2
3
4
enter the element to be found:3
element found at index 2
```

```
@OMANSH DHAWAN 202 CSE-B
enter the number of elements in an array:
4
enter 4 elements in an array:
1
2
3
4
enter the element to be found:5
element not found
```

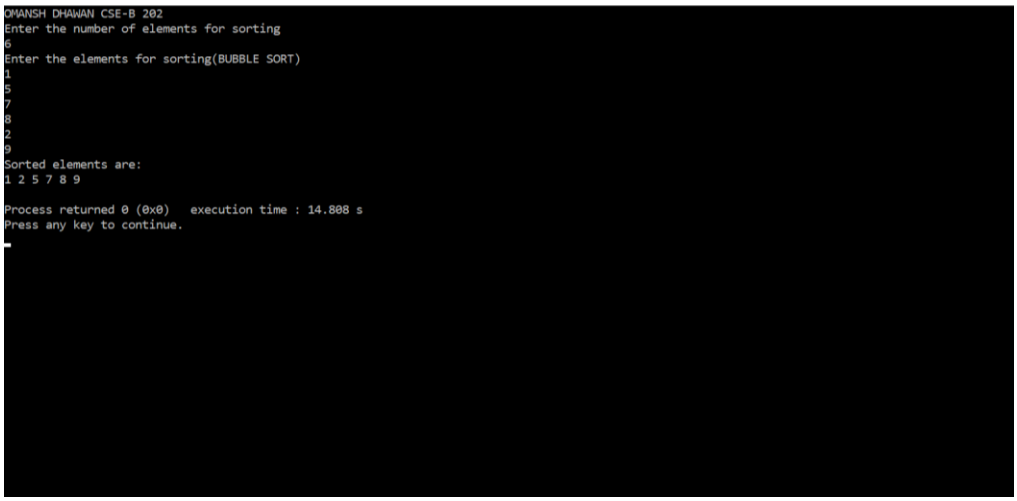
b) Write a program to implement Bubble Sort.

Source code :

```
#include <stdio.h>
#include<stdlib.h>
void swap(int *x,int *y)
{
 int temp=*x;
 *x=*y;
 *y=temp;
}
void Bubble(int A[],int n)
{
 int i,j,flag=0;

 for(i=0;i<n-1;i++)
 {
 flag=0;
 for(j=0;j<n-i-1;j++)
 {
 if(A[j]>A[j+1])
 {
 swap(&A[j],&A[j+1]);
 flag=1;
 }
 }
 if(flag==0)
 break;
 }

}
int main()
```

```
{
 int A[10],n,i;
 printf("OMANSH DHAWAN CSE-B 202\n");
 printf("Enter the number of elements for sorting\n");
 scanf("%d",&n);
 printf("Enter the elements for sorting(BUBBLE SORT)\n");
 for(i=0;i<n;i++)
    scanf("%d",&A[i]);

 Bubble(A,n);
 printf("Sorted elements are:\n");
 for(i=0;i<n;i++)
 printf("%d ",A[i]);
 printf("\n");
 return 0;
}
```

Output :



c) Write a program for matrix multiplication

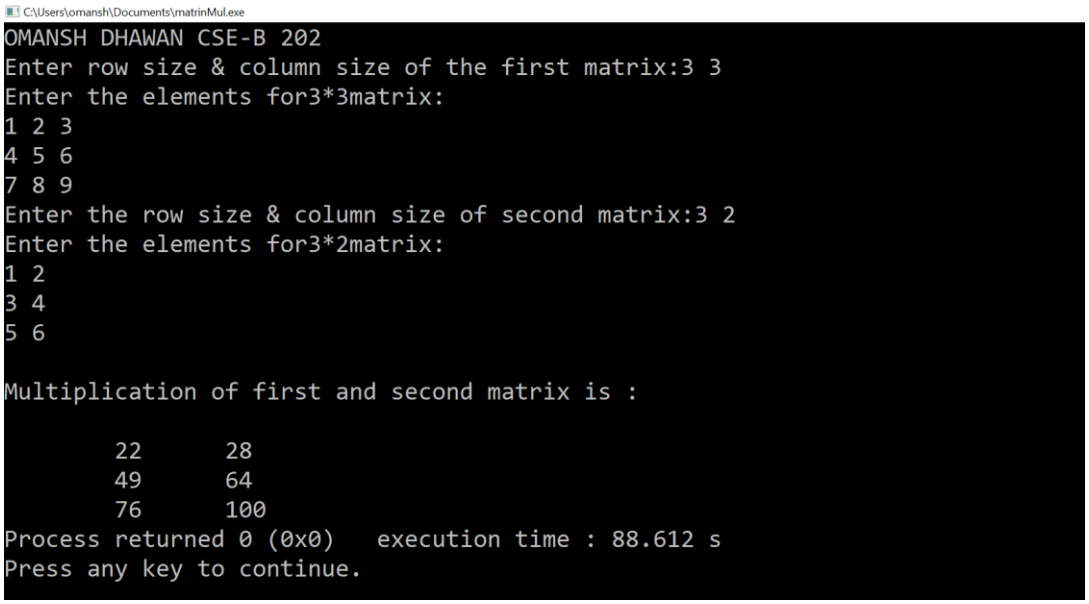
Source code :

```
#include<iostream>
#include<conio.h>
using namespace std;
class matrix
{
int i,j,a[10][10],b[10][10],c[10][10],row1,row2,col1,col2,k;
public:
void getdata();
void putdata();
void sum();
void multiply();
void transpose();
};
void matrix::getdata()
{
cout<<"Enter row size & column size of the first matrix:";
cin>>row1>>col1;
cout<<"Enter the elements for"<<row1<<"*"<<col1<<"matrix:";
for(i=0;i<row1;i++)
{
for(j=0;j<col1;j++)
{
cin>>a[i][j];
}}
cout<<"Enter the row size & column size of second matrix:";
cin>>row2>>col2;
if(col1==row2)
{
cout<<"Enter the elements for"<<row2<<"*"<<col2<<"matrix:";
```

```
for(i=0;i<row2;i++)
{
for(j=0;j<col2;j++)
{
cin>>b[i][j];
}}
}
else
cout<<"matrix algebra is not possible";
}
void matrix::multiply()
{
   cout<<"\nMultiplication of first and second matrix is : "<<endl;
for(i = 0; i < row1; ++i)
       for(j = 0; j < col2; ++j)
       {
          c[i][j]=0;
       }

   for(i = 0; i < row1; ++i)
       for(j = 0; j < col2; ++j)
          for(k = 0; k < col1; ++k)
          {
             c[i][j] += a[i][k] * b[k][j];
          }
          }

void matrix::putdata()
{
for(i=0;i<row1;i++)
{
cout<<"\n";
```

```
for(j=0;j<col2;j++)
{
cout<<"\t"<<c[i][j];
}}}
int main()
{
system("cls");
cout<<"OMANSH DHAWAN CSE-B 202\n";
matrix obj;
obj.getdata();
obj.multiply();
obj.putdata();
return 0;
}
```

Output :



Experiment – 5

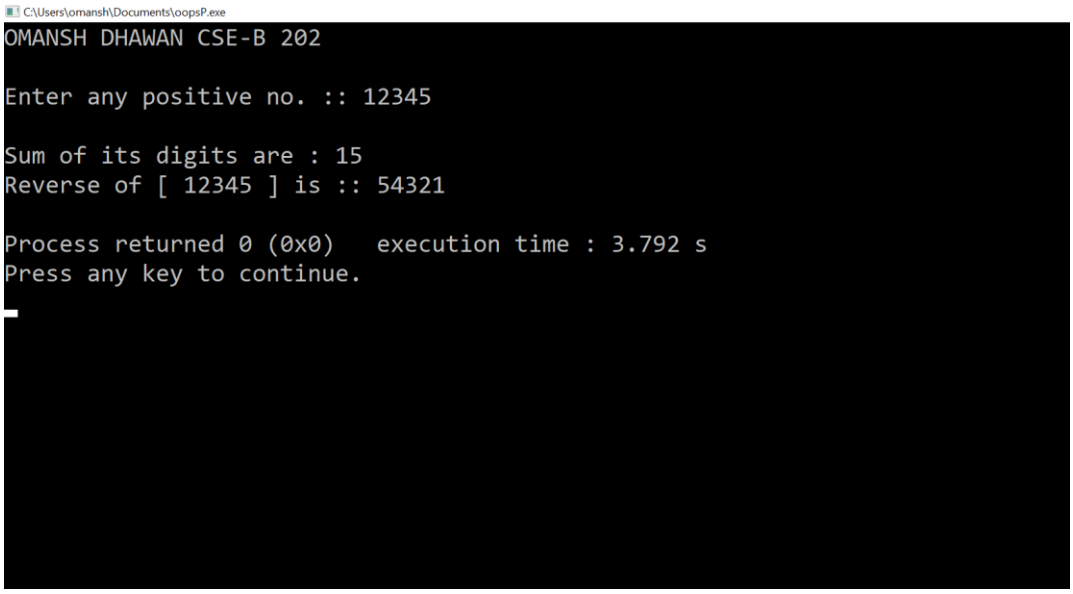
a) Write a program to reverse a given number and find its sum using classes.

Source code :

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

class rev
{
	private:
		int n,n1,rn=0,d;
	public:
		void input();
		void calc();
		void display();
};
void rev::input()
{
	cout<<"\nEnter any positive no. :: ";
	cin>>n;
}
void rev::calc()
{
	n1=n;
	int sum=0;
	while(n>0)
	{
		d=n%10;
		rn=(rn*10)+d;
		sum+=d;
		n/=10;
```

```
    }
    cout<<"\nSum of its digits are : "<<sum;
}
void rev::display()
{
        cout<<"\nReverse of [ "<<n1<<" ] is :: "<<rn<<"\n";
}
int main ()
{
    rev r;
    cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
    r.input();
    r.calc();
    r.display();
    return 0;
}
```

Output :



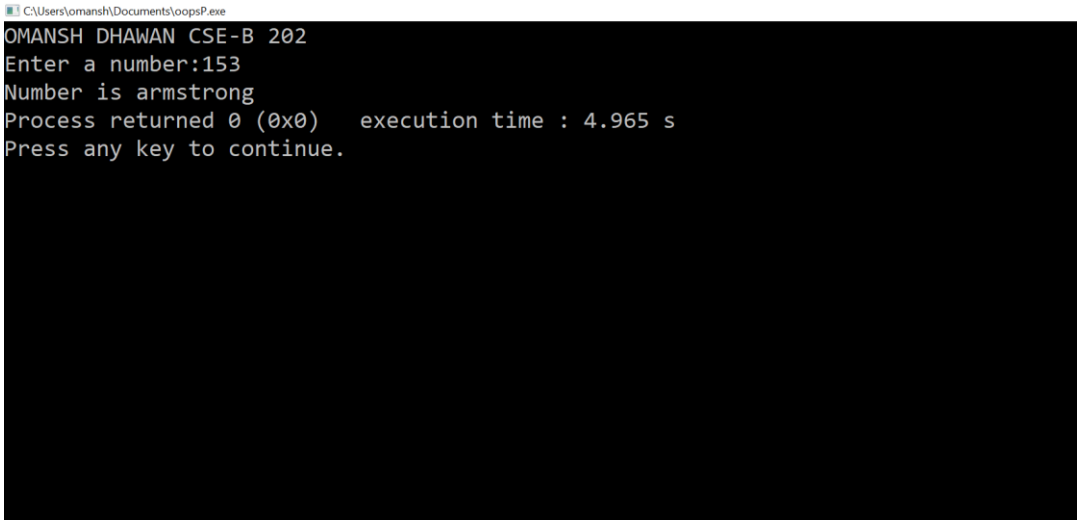
b) Write a program to check whether given number is Armstrong or not using classes.

Source code :

```
#include<iostream>
using namespace std;
class Test {
public:

   int checkArmstrong(int x) {
       int r, num = 0;

       while (x > 0) {
           r = x % 10;
           num = num + r * r* r;
           x = x / 10;
       }
       return num;
   }
};

int main() {

   int x, arm;
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   cout << "Enter a number:";
   cin >> x;

   Test obj;
   arm = obj.checkArmstrong(x);
```

```
    if (arm == x) {
        cout << "Number is armstrong";
    } else {
        cout << "Number is not armstrong";
    }

    return 0;
}
```

Output :

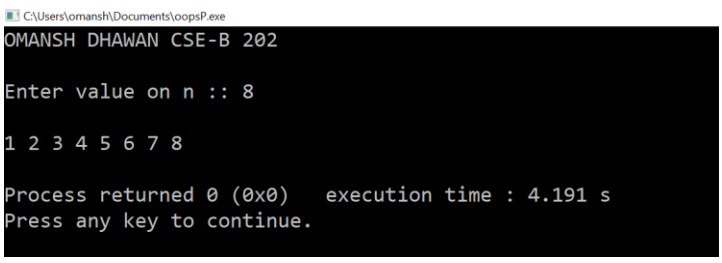


c) Write a program to print all numbers upto given numbers using classes.

Source Code :

```
#include<iostream>
using namespace std;
class Num
{
   public:
      static int i;
      Num()
   {     cout<<i++<<" ";
      }};
int Num::i=1;
int main()
{
   int n;
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   cout<<"\nEnter value on n :: ";
   cin>>n;
   cout<<"\n";
   Num obj[n];
   cout<<"\n";
   return 0;
}
```

Output :



Experiment – 6

a)Write a program to illustrate the concept of array of objects using classes.

Source code :

```
#include <iostream>
using namespace std;
class MyClass {
  int x;
public:
  void setX(int i) { x = i*i; }
  int getX() { return x; }
};
int main()
{
  MyClass obs[4];
  int i;
  cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
    for(i=0; i < 4; i++)
    obs[i].setX(i);
for(i=0; i < 4; i++)
    cout << "obs[" << i << "] = " << obs[i].getX() << "\n";
 return 0;
}
```

Output :

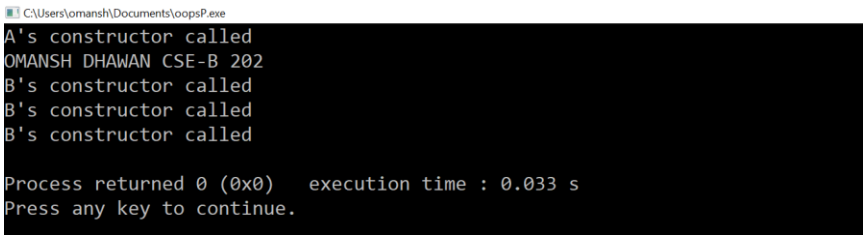
```
C:\Users\omansh\Documents\oopsP.exe
OMANSH DHAWAN CSE-B 202
obs[0] = 0
obs[1] = 1
obs[2] = 4
obs[3] = 9

Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.
```

b) Write a program to illustrate the the concept of static data member.

Source code :

```
#include <iostream>
using namespace std;
class A
{
	int x;
public:
	A() { cout << "A's constructor called " << endl; }
};
class B
{
	static A a;
public:
	B() { cout << "B's constructor called " << endl; }
	static A getA() { return a; }
};
A B::a;
int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
	B b1, b2, b3;
	A a = b1.getA();
	return 0;
}
```

Output :

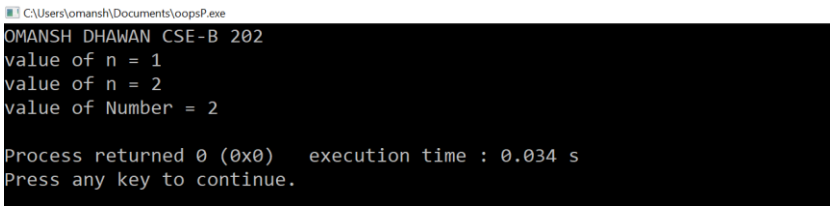


c) Write a program to illustrate the concept of static data functions.

Source code :

```
#include <iostream>
using namespace std;
class Example{
    static int Number;
    int n;
public:
void set_n(){ n = ++Number; }
void show_n(){ cout<<"value of n = "<<n<<endl; }
static void show_Number(){ cout<<"value of Number = "<<Number<<endl; } };
 int Example:: Number;
 int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
    Example example1, example2;
    example1.set_n();
    example2.set_n();
    example1.show_n();
    example2.show_n();
    Example::show_Number();
     return 0;
}
```

Output :



Experiment – 7

a) Write a program to implement the concept of pass by value using TIME class.

Source code :

```
#include <iostream>
using namespace std;
class Time {
public:
        int a;
        void add(Time E)
        {a = a + E.a;
        }};
int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
        Time E1, E2;
        E1.a = 50;
        E2.a = 100;
        cout << "Initial Values \n";
        cout << "Value of object 1: " << E1.a<< "\n& object 2: " << E2.a<< "\n\n";
        E2.add(E1);
        cout << "New values \n";
        cout << "Value of object 1: " << E1.a<< "\n& object 2: " << E2.a<< "\n\n";
            return 0;
}
```

Output :

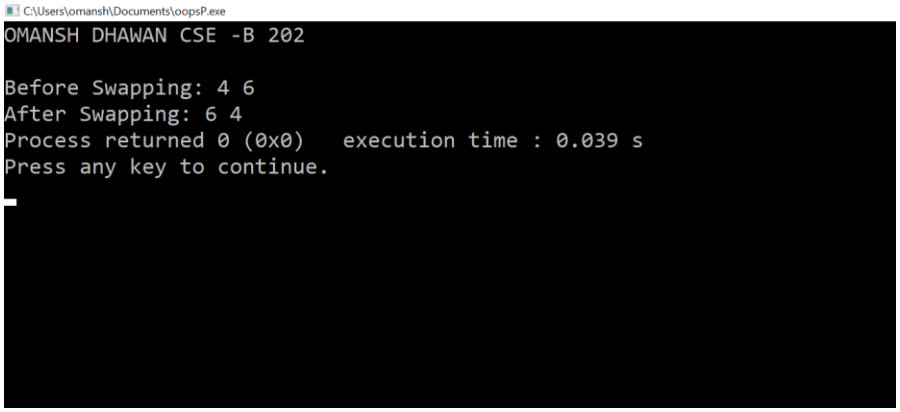
b) Write a program to implement the concept of call by reference using friend function.

Source code :

```
#include <iostream>

using namespace std;

class Swap {

	int temp, a, b;

public:


	Swap(int a, int b)
	{
		this->a = a;
		this->b = b;
	}


	friend void swap(Swap&);
};

void swap(Swap& s1)
{

	cout << "\nBefore Swapping: " << s1.a << " " << s1.b;
```

```
        s1.temp = s1.a;
        s1.a = s1.b;
        s1.b = s1.temp;
        cout << "\nAfter Swapping: " << s1.a << " " << s1.b;
}
int main()
{   cout<<"OMANSH DHAWAN CSE -B 202"<<endl;
        Swap s(4, 6);
        swap(s);
        return 0;
}
```

Output :



Experiment – 8

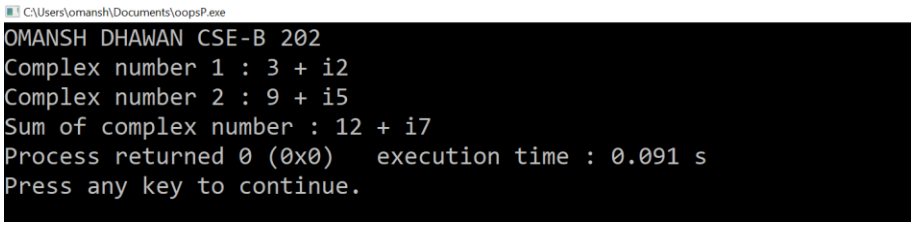
a) Write a program to implement the concept of constructor overloading using complex number class.

Source code :

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

        public:
                int real, imaginary;
        Complex()
        {
        }
  Complex(int tempReal, int tempImaginary)
        {
                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()
{
	cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
	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 :

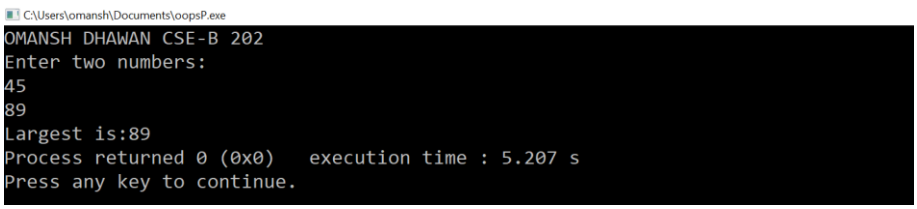


b) Write a program to find the greatest of two numbers using friend function.

Source code :

```
#include<iostream>
using namespace std;
class Test {
private:
   int x, y;
public:
 void input() {
       cout << "Enter two numbers:";
       cin >> x>>y;    }
friend void find(Test t);
};
void find(Test t) {
   if (t.x > t.y) {
       cout << "Largest is:" << t.x;
   } else {
       cout << "Largest is:" << t.y;
   }}
int main() {
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   Test t;
   t.input();
   find(t);
 return 0;  }
```

Output :

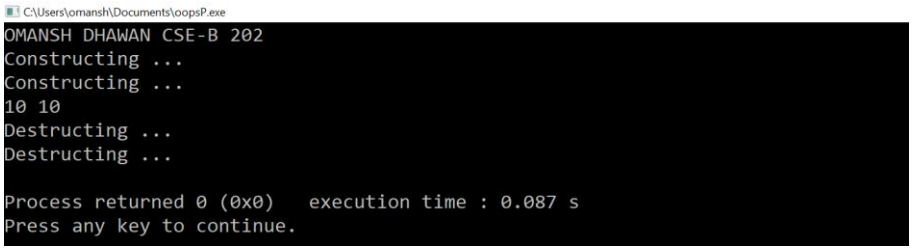


c) Write a program to implement the concept of constructor and destructor.

Source code :

```
#include <iostream>
using namespace std;
 class MyClass
 {
    public:
    int x;
   MyClass();
  ~MyClass();
 };
  MyClass::MyClass()
{   cout<< "Constructing ...\n";
     x = 10;  }
   MyClass::~MyClass()
{ cout<< "Destructing ...\n"; }
 int main()
{    cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
     MyClass ob1;
     MyClass ob2;
     cout <<ob1.x<< " " <<ob2.x<<"\n";
     return 0;   }
```

Output :



Experiment – 9

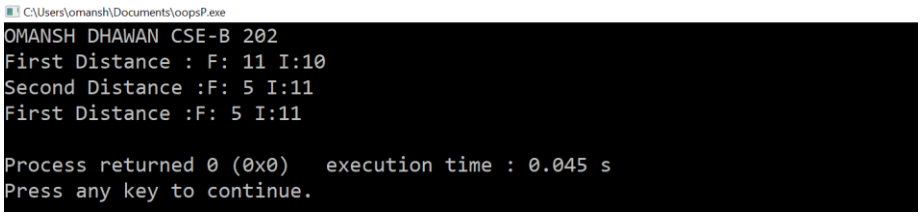
a) Write a program to overload assignment(<=) operator.

Source code :

```
#include <iostream>
using namespace std;
class Distance {
  private:
    int feet;
    int inches;

  public:
    Distance() {
      feet = 0;
      inches = 0;
    }
    Distance(int f, int i) {
      feet = f;
      inches = i;
    }
    void operator = (const Distance &D ) {
      feet = D.feet;
      inches = D.inches;
    }

    void displayDistance() {
      cout << "F: " << feet <<  " I:" <<  inches << endl;
    }
};
```

```
int main()
{

 cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   Distance D1(11, 10), D2(5, 11);

   cout << "First Distance : ";
   D1.displayDistance();

 cout << "Second Distance :";
   D2.displayDistance();

   D1 = D2;

 cout << "First Distance :";
   D1.displayDistance();

   return 0;
}
```

Output :



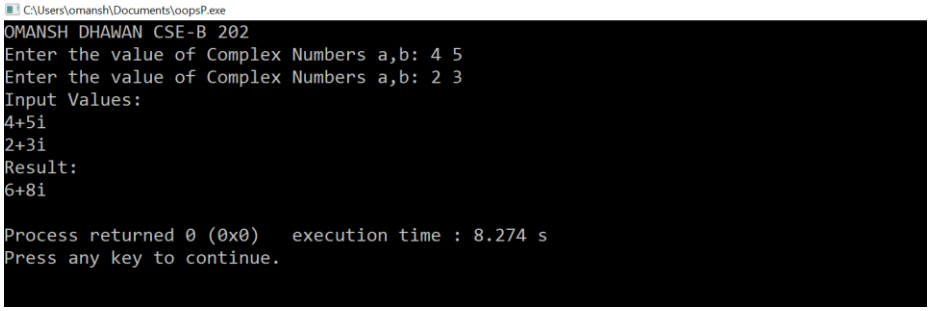
b) Write a program to overload binary(+) operator.

Source code :

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

class complex {
   int a, b;
public:

   void getvalue() {
      cout << "Enter the value of Complex Numbers a,b:";
      cin >> a>>b;
   }

   complex operator+(complex ob) {
      complex t;
      t.a = a + ob.a;
      t.b = b + ob.b;
      return (t);
   }
   void display() {
      cout << a << "+" << b << "i" << "\n";
   }
};
int main() {
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   complex obj1, obj2, result, result1;

   obj1.getvalue();
   obj2.getvalue();
```

```
    result = obj1 + obj2;

    cout << "Input Values:\n";
    obj1.display();
    obj2.display();

    cout << "Result:"<<endl;
    result.display();

    return 0;
}
```

Output :



Experiment – 10

a) Write a program to implement Single inheritance

Source code :

```
#include <iostream>
using namespace std;
 class Account {
   public:
   float salary = 60000;
 };
   class Programmer: public Account {
   public:
   float bonus = 5000;
   };
int main(void) {
     cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
     Programmer p1;
     cout<<"Salary: "<<p1.salary<<endl;
     cout<<"Bonus: "<<p1.bonus<<endl;
    return 0;
}
```

Output :

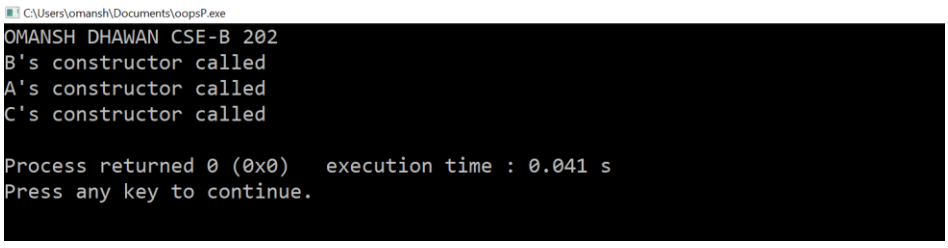
```
C:\Users\omansh\Documents\oopsP.exe
OMANSH DHAWAN CSE-B 202
Salary: 60000
Bonus: 5000

Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

b) Write a program to implement Multiple inheritance

Source code :

```
#include<iostream>
using namespace std;
class A
{
public:
A() { cout << "A's constructor called" << endl; }
};
class B
{
public:
B() { cout << "B's constructor called" << endl; }
};
class C: public B, public A // Note the order
{
public:
C() { cout << "C's constructor called" << endl; }
};
int main()
{       cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
        C c;
        return 0;
}
```

Output :

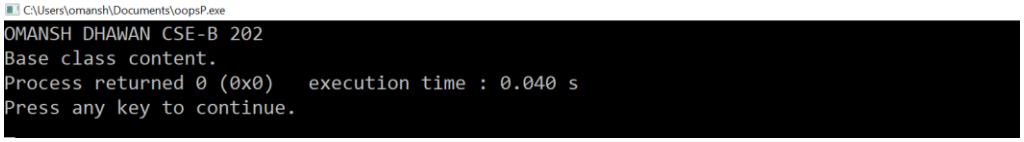


c) Write a program to implement MultiLevel inheritance

Source code :

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

class A {
   public:
     void display() {
        cout<<"Base class content.";
     }
};

class B : public A {};

class C : public B {};

int main() {
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   C obj;
   obj.display();
   return 0;
}
```

Output :



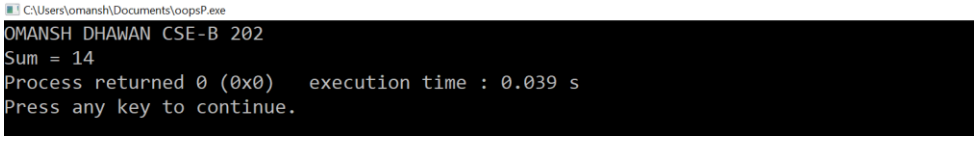
Experiment – 11

a) Write a program to implement Hybrid inheritance.

Source code :

```
#include <iostream>
using namespace std;
class A
{          public:
            int x;   };
class B : public A
{          public:
            B()  {  x = 10; }  };
class C
 {          public:
            int y;
            C() { y = 4; }  };
class D : public B, public C
{          public:
            void sum()
            { cout << "Sum = " << x + y;  }  };
int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
    D obj1;
            obj1.sum();
            return 0;   }
```

Output :

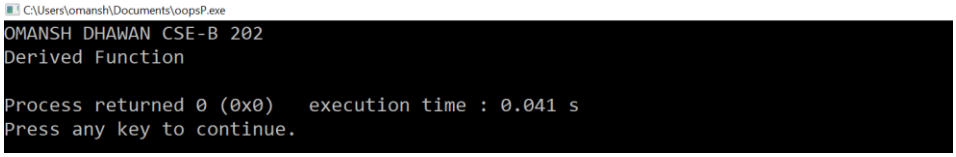


b) Write a program to implement Hybrid inheritance with virtual function.

Source code :

```
#include <iostream>
using namespace std;
class Base {
  public:
   virtual void print() {
      cout << "Base Function" << endl;
   }};
class Derived : public Base {
  public:
   void print() {
      cout << "Derived Function" << endl;
   }};
int main() {
   Derived derived1;
   Base* base1 = &derived1;
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   base1->print();

return 0;
}
```

Outout :



Experiment – 12

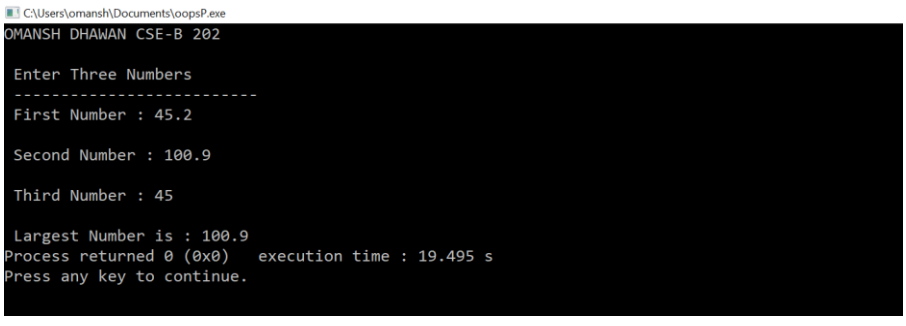
a) Write a program to implement Template function max 3 numbers.

Source code :

```
#include<iostream>

using namespace std;

template<class Test> Test FindLarge(Test num1,Test num2, Test num3)
{
	if(num1>=num2)
	{
		if(num1>=num3)
			return num1;
		else
			return num3;
	}
	else
	{
		if(num2 >= num3)
			return num2;
		else
			return num3;
	}
}
int main()
{
	float num1, num2, num3, large;

cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
```

```
    cout << "\n Enter Three Numbers";
    cout<<"\n -------------------------";

    cout<<"\n First Number : ";
    cin>>num1;

    cout<<"\n Second Number : ";
    cin>>num2;

    cout<<"\n Third Number : ";
    cin>>num3;

    large=FindLarge(num1,num2,num3);

    cout<<"\n Largest Number is : "<<large;

    return 0;
}
```

Output :

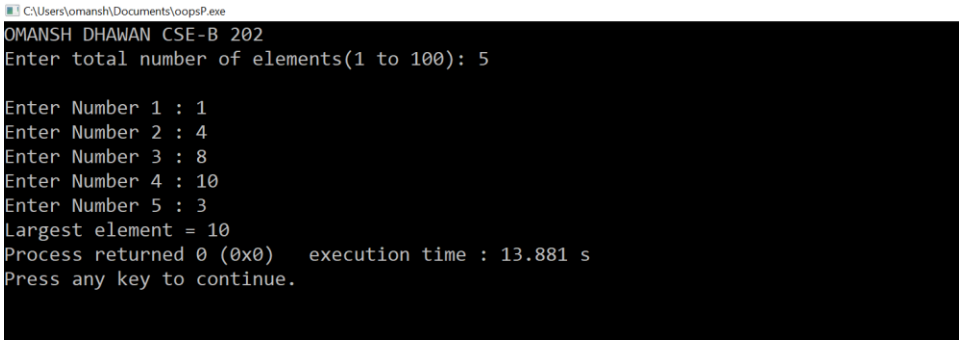


b) Write a program to find maximum element in an array.

Source code :

```
#include <iostream>
using namespace std;
int main()
{  int i, n;
   float arr[100];
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   cout << "Enter total number of elements(1 to 100): ";
   cin >> n;
   cout << endl;
for(i = 0; i < n; ++i)
   { cout << "Enter Number " << i + 1 << " : ";
     cin >> arr[i];
   }
for(i = 1;i < n; ++i)
   { if(arr[0] < arr[i])
        arr[0] = arr[i];
   }
   cout << "Largest element = " << arr[0];
return 0; }
```

Output :



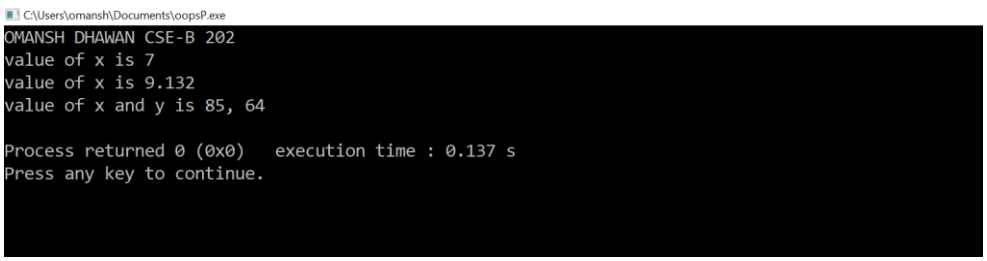
Experiment – 13

a) Write a program to implement compile time polymorphism.

Source code :

```
#include <bits/stdc++.h>
using namespace std;
class God
{
public:
        void func(int x)
                {  cout << "value of x is " << x << endl;  }
        void func(double x)
                { cout << "value of x is " << x << endl; }
        void func(int x, int y)
                { cout << "value of x and y is " << x << ", " << y << endl; } };
int main() {
        God obj1;
        cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
        obj1.func(7);
        obj1.func(9.132);
        obj1.func(85,64);
        return 0;
}
```

Output :



b) Write a program to implement runtime polymorphism.

Source code:

```
#include <iostream>
using namespace std;
class base
{    public:
        virtual void print ()
        { cout<< "print base class" <<endl; }
        void show ()
        { cout<< "show base class" <<endl; }  };
class derived:public base
{    public:
        void print ()
        { cout<< "print derived class" <<endl; }
        void show ()
        { cout<< "show derived class" <<endl; }   };
int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
        base *bptr;
        derived d;
        bptr = &d;
        bptr->print();
        bptr->show();
        return 0;   }
```

Output :

```
C:\Users\omansh\Documents\oopsP.exe
OMANSH DHAWAN CSE-B 202
print derived class
show base class

Process returned 0 (0x0)   execution time : 0.041 s
Press any key to continue.
```

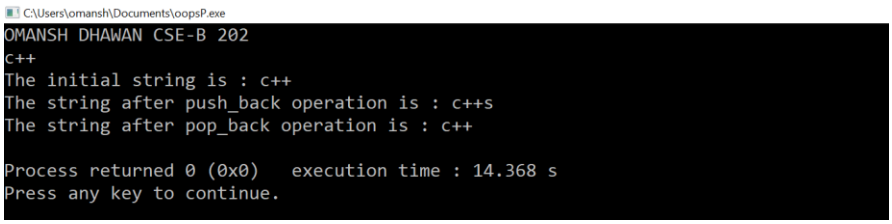
Experiment – 14

a) Write a program to implement class string .

Source code :

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string str;
	cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
	getline(cin,str);
	cout << "The initial string is : ";
	cout << str << endl;
	str.push_back('s');
	cout << "The string after push_back operation is : ";
	cout << str << endl;
	str.pop_back();
	cout << "The string after pop_back operation is : ";
	cout << str << endl;
	return 0;
}
```

Output :

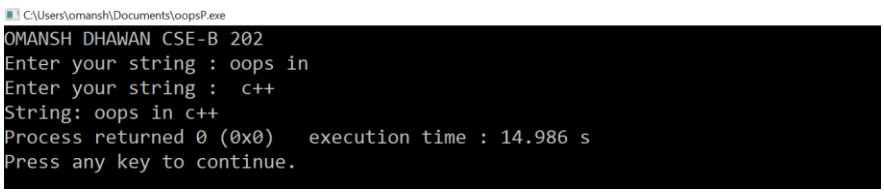


b) Write a program to overload (+) operator to concatenate string.

Source code :

```
#include<iostream>
#include<string.h>
using namespace std;
class String
{char str[20];
public:  void input()
   {   cout<<"Enter your string : ";
       cin.getline(str,20);}
   void display()
   {  cout<<"String: "<<str; }
   String operator+(String s)
   {  String obj;
       strcat(str,s.str);
       strcpy(obj.str,str);
       return obj; } };
int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   String str1,str2,str3;
   str1.input();
   str2.input();
   str3=str1+str2;
   str3.display();
   return 0; }
```

Output :



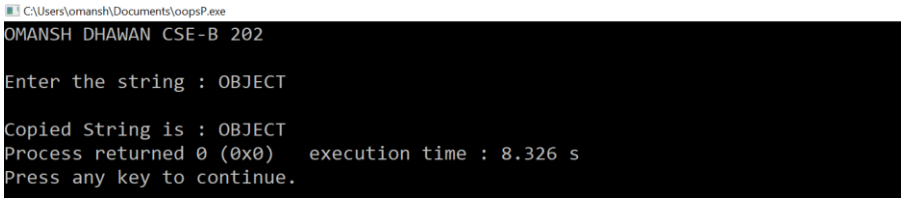
c) Write a program to overload (=) operator to copy string..

Source code :

```
#include<iostream>
#include<string.h>
using namespace std;
class my_string{
   private:

          char str[30];
   public:
         void getdata();
         void display();
         void operator= (my_string str1);

};



void my_string::getdata()
{
  cout<<"\nEnter the string : ";
  cin>>str;
}

void my_string::display()
{
   cout<<"\n"<<str;
}
```

```
void my_string::operator= (my_string str1)
{
  strcpy(str1.str,str);
  cout<<"\nCopied String is : "<<str1.str;
}


int main()
{
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   my_string a,b;
   a.getdata();
   a=b;

   return 0;
}
```

Output :



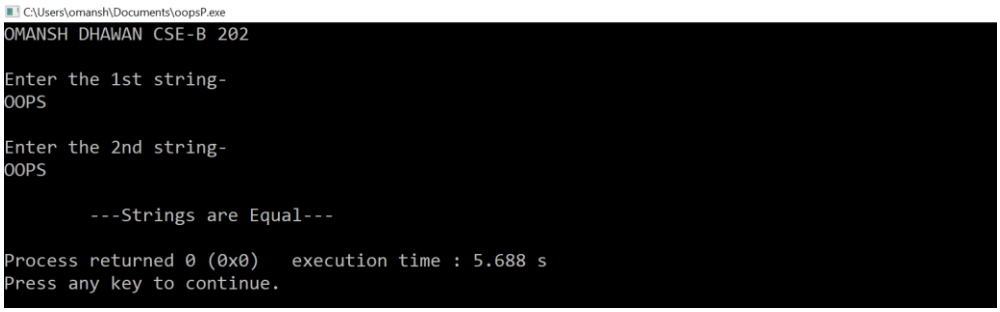
d) Write a program to overload (<=) operator for string comparison.

Source code :

```
#include<iostream>
#include<string.h>
using namespace std;
class my_string{
  private:

        char str[30];
  public:
       void getdata();
       void display();
       int operator<= (my_string str1);

};

void my_string::getdata()
{
 cin>>str;
}

void my_string::display()
{
  cout<<"\n"<<str;
}

int my_string::operator<= (my_string str1)
{
  if(strcmp(str,str1.str)==0)
    return 1;
```

```
    return 0;
}

int main()
{
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   my_string a,b;
   int c;
   cout<<"\nEnter the 1st string-\n";
     a.getdata();
     cout<<"\nEnter the 2nd string-\n";
     b.getdata();
      c=a<=b;
      if(c==1)
          cout<<"\n\t---Strings are Equal---\n";
      else
          cout<<"\n\t---Strings are not Equal---\n";
return 0;
}
```

Output :



Experiment – 15

a) Write a program to read and write to a file.

Source code :

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
        fstream ob;
        ob.open("test.txt", ios::out);
        ob << "hello world\n";
        ob << "this is my first file";
        ob.close();
        ob.open("test.txt", ios:: in);
       while (!ob.eof())
  {                 string str;
                   ob >> str;
                   cout << str << "\n";
   }
        ob.close();
        return 0;    }
```

Output :

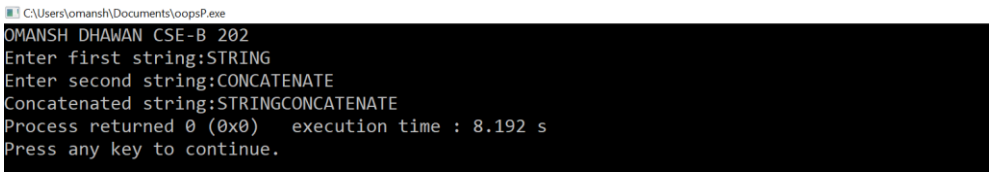
```
C:\Users\omansh\Documents\oopsP.exe
OMANSH DHAWAN CSE-B 202
hello
world
this
is
my
first
file

Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

b) Write a program to concatenate two strings without library functions.

Source code :

```
#include<iostream>
using namespace std;
int main() {
   char str1[100], str2[100];
   int i = 0, j = 0;
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   cout << "Enter first string:";
   cin.getline(str1, 100);
   cout << "Enter second string:";
   cin.getline(str2, 100);
   while (str1[i] != '\0') {
      i++;
   }
   while (str2[j] != '\0') {
      str1[i] = str2[j];
      j++;
      i++;
   }
   str1[i] = '\0';
   cout << "Concatenated string:" << str1;
   return 0;
}
```

Output :

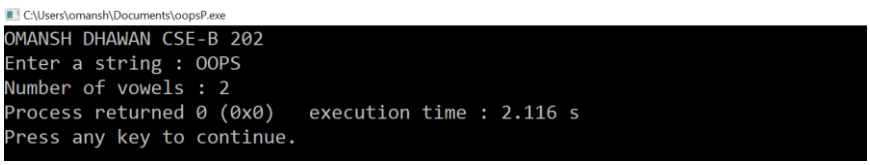


c) Write a program to find number of vowels in a string.

Source code :

```
#include<iostream>
#include<string.h>
using namespace std;
int main ()
{
   char str[50];
   int v = 0, c = 0, n = 0, s = 0;

   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   cout << "Enter a string : ";
   gets(str);
   for (int i = 0; str[i]!='\0'; ++i)
   {
      if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' || str[i] == 'o' || str[i] == 'u' || str[i] == 'A' || str[i] == 'E' || str[i] == 'I' || str[i] == 'O' || str[i] == 'U')
         ++v;
   }

   cout << "Number of vowels : " << v;
   return 0;
}
```

Output :



Experiment – 16

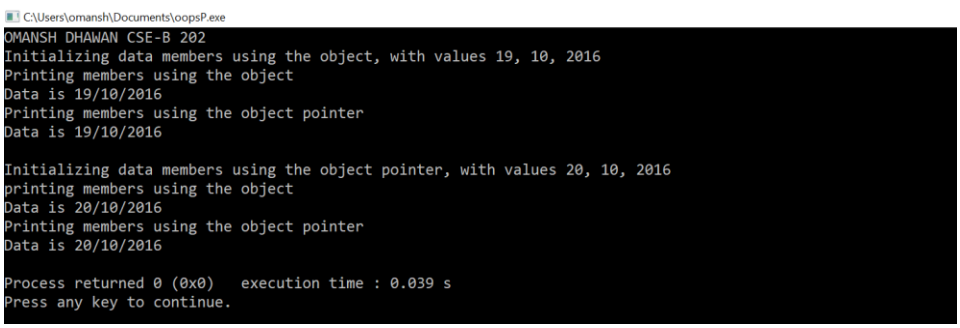
a) Write a c++ program to explain concept of pointer to object.

Source code :

```
#include<iostream>
using namespace std;
class Date
{
   private:
      short int dd, mm, yy;
   public:
      Date() //constrctor:
         {
            dd = mm = yy = 0;
         }
      void getdata(int i, int j, int k)
         {
            dd = i;
            mm = j;
            yy = k;
         }
      void prndata(void)
         {
            cout<<"\nData is "<<dd<<"/"<<mm<<"/"<<yy<<"\n";
         }
};
int main()
{
   Date D1;
```

```
    Date *dptr;
    cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
    cout<<"Initializing data members using the object, with values 19, 10, 2016"<<endl;
    D1.getdata(19,10,2016);

    cout<<"Printing members using the object ";
    D1.prndata();

    dptr = &D1;
    cout<<"Printing members using the object pointer ";
    dptr->prndata();

    cout<<"\nInitializing data members using the object pointer, with values 20, 10, 2016"<<endl;
    dptr->getdata(20, 10, 2016);
    cout<<"printing members using the object ";
    D1.prndata();

    cout<<"Printing members using the object pointer ";
    dptr->prndata();

    return 0;
}
```

Output :

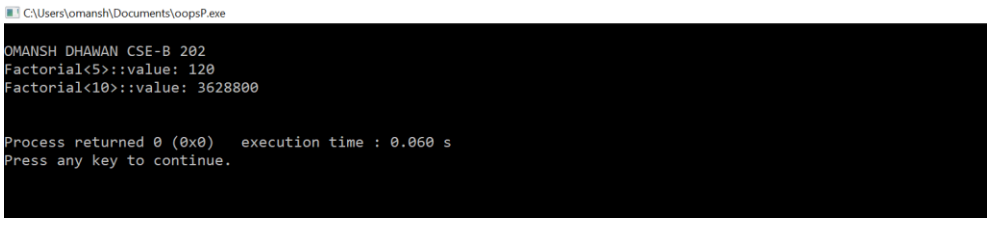


b) Study the concept of components of template meta programming.

Source code :

```
#include <iostream>
using namespace std;
template <int N>
struct Factorial{
   static int const value = N * Factorial<N-1>::value;
};

template <>
struct Factorial<1>{
   static int const value = 1;
};

int main(){
   cout <<endl;
   cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
   cout << "Factorial<5>::value: " << Factorial<5>::value <<endl;
   cout << "Factorial<10>::value: " << Factorial<10>::value << endl;
   cout <<endl;

}
```

Output :



c) Compile time code optimization.

Types of Code Optimization –The optimization process can be broadly classified into two types :

1. Machine Independent Optimization – This code optimization phase attempts to improve the intermediate code to get a better target code as the output. The part of the intermediate code which is transformed here does not involve any CPU registers or absolute memory locations.

2. Machine Dependent Optimization – Machine-dependent optimization is done after the target code has been generated and when the code is transformed according to the target machine architecture. It involves CPU registers and may have absolute memory references rather than relative references. Machine-dependent optimizers put efforts to take maximum advantage of the memory hierarchy.

Code Optimization is done in the following different ways :

1. **Compile Time Evaluation :**

(i) A = 2*(22.0/7.0)*r
Perform 2*(22.0/7.0)*r at compile time.
(ii) x = 12.4
y = x/2.3
Evaluate x/2.3 as 12.4/2.3 at compile time.

2. **Variable Propagation :**

```
//Before Optimization
c = a * b
x = a
till
d = x * b + 4
//After Optimization
c = a * b
x = a
till
d = a * b + 4
```

3. **Dead code elimination :**

```
c = a * b

x = a

till

d = a * b + 4

//After elimination :
c = a * b
till
d = a * b + 4
```

4. Code Motion :

```
a = 200;
while(a>0)
{
        b = x + y;
        if (a % b == 0}
        printf("%d", a);
}


//This code can be further optimized as
a = 200;
b = x + y;
while(a>0)
{
        if (a % b == 0}
        printf("%d", a);
}
```

5. Induction Variable and Strength Reduction :

```
i = 1;

while (i<10)

{

        y = i * 4;
}


//After Reduction
i = 1
t = 4
{
while( t<40)
y = t;
t = t + 4;
}
```

d) Implement the rule of Big 5 in a program.

Source code :

```
#include <cstring>
#include <iostream>
#include <string>
using namespace std;
class CPPR5Class
{
public:
   CPPR5Class() :
      data (new char[14])
   {    strcpy(data, "Hello, World!\nWelcome to Easy Engineering");    }
   CPPR5Class (const CPPR5Class& original) :
      data (new char[strlen (original.data) + 1])
   {    strcpy(data, original.data);   }
   CPPR5Class (CPPR5Class&& original) noexcept :
      data(original.data)
   {   original.data = nullptr;   }
   ~CPPR5Class() noexcept
   {   delete[] data;  }
   CPPR5Class& operator= (const CPPR5Class& original)
   {
      CPPR5Class tmp(original);
      *this = move(tmp);
      return *this;
   }
   CPPR5Class& operator= (CPPR5Class&& original) noexcept
   {   if (this == &original)
      {   return *this;  }
      delete[] data;
      data = original.data;
```

```
        original.data = nullptr;
        return *this;
    }
private:
    friend ostream& operator<< (ostream& os, const CPPR5Class& cPPR5Class)
    {   os << cPPR5Class.data;
        return os;
    }
    char* data;
};
int main()
{
    const CPPR5Class cPPR5Class;
    cout<<"OMANSH DHAWAN CSE-B 202"<<endl;
    cout << cPPR5Class <<endl;
    return 0;
}
```

Output :