# OBJECT ORIENTED PROGRAMMING

## LAB # 5

## ASSIGNMENT

**NAME:** S.M.IRTIZA
**ROLL NO. :** 22K-4638
**CLASS:** 2-F

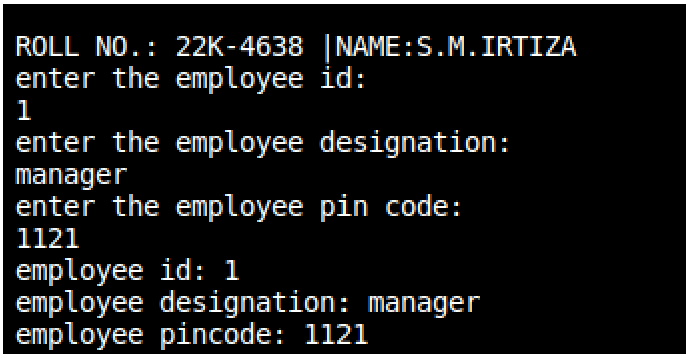
# QUESTION # 01:

Write a program in which a class named EMPLOYEE has private member variables named EMP_ID,EMP_DESIGNATION , EMP_PINCODE. Use a public function to initialize the variables and print all data.

**CODE:**

```
//ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
#include<iostream>
#include<string>
using namespace std;
class EMPLOYEE {
int EMP_ID;
string EMP_DESIGNATION;
string EMP_PINCODE;

public:
EMPLOYEE(){}
void inputData(){
cout<<"enter the employee id: "<<endl;
cin>>EMP_ID;
cout<<"enter the employee designation: "<<endl;
cin>>EMP_DESIGNATION;
cout<<"enter the employee pin code: "<<endl;
cin>>EMP_PINCODE;
}
void display(){
cout<<"employee id: "<<EMP_ID<<endl;
cout<<"employee designation: "<<EMP_DESIGNATION<<endl;
cout<<"employee pincode: "<<EMP_PINCODE<<endl;
}
};
int main(){
cout<<"ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA"<<endl;
EMPLOYEE E;
E.inputData();
E.display();
}
```

**OUTPUT:**



```
ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
enter the employee id:
1
enter the employee designation:
manager
enter the employee pin code:
1121
employee id: 1
employee designation: manager
employee pincode: 1121
```

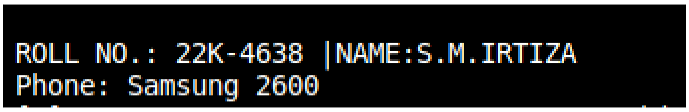
# QUESTION # 02:

Find out and specify where and why the static keyword should be used, and rectify if the program has any errors.

**CODE:**

```
//ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
#include <iostream>
#include<string> //this header file is missing
using namespace std;
class Samsung{
private:
static string ph_name; //here static keyword is used
public:
static void name(){ // here static keyword is used
cout << "Phone: "<< ph_name<<endl;

}

static void set_name(string name){ //here static keyword is used
ph_name = name;
}
};

//Initializing private static member
string Samsung::ph_name = ""; //the variable name is not same, so correction done

int main()
{
cout<<"ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA"<<endl;
//no object has been created

//accessing static function directly with class name
Samsung::set_name("Samsung 2600"); //class name is Samsung not the nokia, so correction done

Samsung::name(); //class name is Samsung not the nokia, so correction done
return 0;
}
```

**OUTPUT:**



```
ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
Phone: Samsung 2600
```

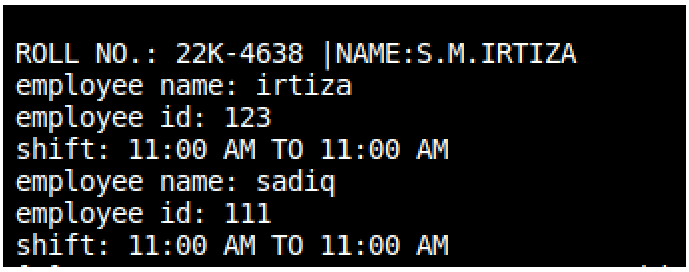
## QUESTION # 03:

Write a program of your own in which you demonstrate the concept of constant keyword.

**CODE:**

```
//ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
#include<iostream>
#include<string>
using namespace std;

class Employee{
    const string shiftTime= "11:00 AM TO 11:00 AM";
    int empID;
    string name;
    public:

    Employee (){}
    Employee(string a, int b){
        name=a;
        empID=b;
    }

    void display(){
        cout<<"employee name: "<<name<<endl;
        cout<<"employee id: "<<empID<<endl;
        cout<<"shift: "<<shiftTime<<endl;
    }
    };
int main(){
    cout<<"ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA"<<endl;
    Employee E("irtiza", 123),E1("sadiq", 111);
    E.display();
    E1.display();
    return 0;
}
```

**OUTPUT:**



```
ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
employee name: irtiza
employee id: 123
shift: 11:00 AM TO 11:00 AM
employee name: sadiq
employee id: 111
shift: 11:00 AM TO 11:00 AM
```

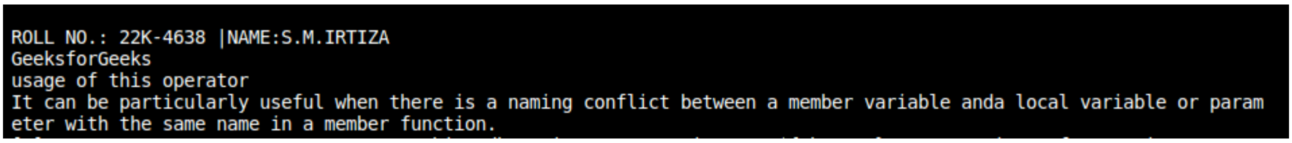
# QUESTION # 04

Where this-> operator must be used in the following program and why?

## CODE:

```
//ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
#include <iostream>
#include<string>
using namespace std;
class Abc
{
string name;
public:
Abc(string name)
{
this->name = name; //here i used the this pointer
}
void display()
{

cout << name << endl;

}
};
// Driver code
int main()
{

cout<<"ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA"<<endl;
Abc gfg("GeeksforGeeks");
gfg.display();

cout<<"usage of this operator"<<endl;
cout<<"It can be particularly useful when there is a naming conflict between a member variable and";
cout<<"a local variable or parameter with the same name in a member function."<<endl;
return 0;
}
```

## OUTPUT:



```
ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
GeeksforGeeks
usage of this operator
It can be particularly useful when there is a naming conflict between a member variable anda local variable or param
eter with the same name in a member function.
```

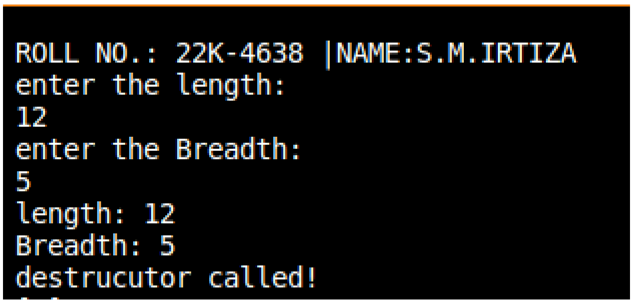
# QUESTION #05

Make a Rectangle class, calculate its Length and Breadth , create a constructor and destructor for the same class.

**CODE:**

```
//ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
#include<iostream>
#include<string>
using namespace std;

class Rectangle {
float Length;
float Breadth;

public:

Rectangle(){
cout<<"enter the length: "<<endl;
cin>>Length;
cout<<"enter the Breadth: "<<endl;
cin>>Breadth;
}
~Rectangle(){
cout<<"destrucutor called!"<<endl;
}

void display(){
cout<<"length: "<<Length<<endl;
cout<<"Breadth: "<<Breadth<<endl;
}
};

int main(){
cout<<"ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA"<<endl;
Rectangle R;
R.display();
return 0;
}
```

**OUTPUT:**



```
ROLL NO.: 22K-4638 |NAME:S.M.IRTIZA
enter the length:
12
enter the Breadth:
5
length: 12
Breadth: 5
destrucutor called!
```