# OBJECT ORIENTED PROGRAMMING

## LAB # 11

## ASSIGNMENT

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

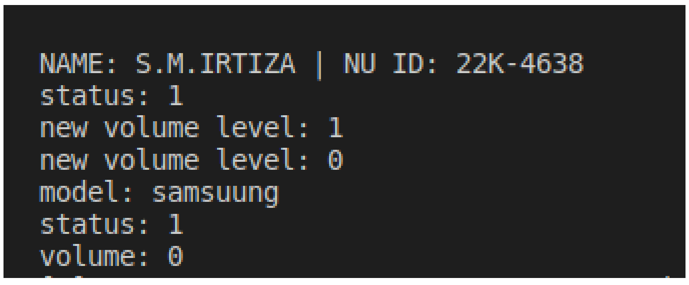
## QUESTION#01:

Write a C++ program that simulates a TV scenario using a friend class connecting multiple classes. The program has two classes, TV and Remote. The TV class has three private member variables: model, isOn, and volume. model stores the model name of the TV as a string, isOn is a boolean variable that indicates whether the TV is on or off, and volume stores the current volume level of the TV. The TV class has a constructor that takes a string argument for the model variable and initializes isOn to false and volume to 0. The class also has a display() function that prints out the current status of the TV, including its model name, whether it is on or off, and the current volume level. The Remote class has a TV object as a private member variable, which allows it to access the TV class's private member variables using the friend function mechanism. The Remote class has a constructor that takes a TV object as an argument and initializes the tv variable to that object. The class has three member functions, togglePower(), increaseVolume(), and decreaseVolume(), that correspond to the power button, volume up button, and volume down button on a remote control, respectively. The togglePower() function toggles the value of isOn, so the TV switches between on and off. It then prints a message indicating the current status of the TV. The increaseVolume() function increases the value of volume by one if the current volume is less than 10. It then prints a message indicating the new volume level. The decreaseVolume() function decreases the value of volume by one if the current volume is greater than 0. It then prints a message indicating the new volume level. The Remote class also has a display() function that calls the display() function of the TV object to print the current status of the TV. Finally, in the main() function, the program creates a TV object with the model name "Samsung" and a Remote object that takes the TV object as an argument. The program then calls various functions of the Remote object to simulate pressing buttons on a remote control, such as toggling the power on, increasing the volume, and decreasing the volume. The program then calls the display() function of the Remote object to print the current status of the TV
ngine of the sedan

**CODE:**

```
//NAME: S.M.IRTIZA | NU ID: 22K-4638
#include<iostream>
#include<string>

using namespace std;
class TV{
   string model;
   bool isOn;
   int volume;
   friend class Remote;

public:
TV(){}
TV(string a){
model=a;
isOn=false;
volume=0;
}

   string getModel()
   {
       return this->model;
   }
   void setModel(string model)
   {
       this->model = model;
   }
   bool getIsOn()
   {
       return this->isOn;
   }
   void setIsOn(bool isOn)
   {
       this->isOn = isOn;
   }
   int getVolume()
   {
       return this->volume;
   }
   void setVolume(int volume)
   {
       this->volume = volume;
   }

void display(){
cout<<"model: "<<getModel()<<endl;
cout<<"status: "<<getIsOn()<<endl;
cout<<"volume: "<<getVolume()<<endl;
}
};


class Remote{
   TV tv;
   public:
   Remote(){}
   Remote(TV t){
   tv.model=t.model;
   tv.isOn=t.isOn;
   tv.volume=t.volume;
   tv.setModel(t.getModel());
   }

   void togglePower(){
       if(tv.isOn==false){
           tv.isOn=true;
           cout<<"status: "<<tv.getIsOn()<<endl;
       }
       else{
           tv.isOn=false;
           cout<<"status: "<<tv.getIsOn()<<endl;
       }
   }
   void increaseVolume(){
    if(tv.volume<10){
        tv.volume+=1;
        cout<<"new volume level: "<<tv.getVolume()<<endl;
    }
    else
    cout<<"new volume level: "<<tv.getVolume()<<endl;
   }
   void decreaseVolume(){
    if(tv.volume>0){
        tv.volume-=1;
        cout<<"new volume level: "<<tv.getVolume()<<endl;
    }
    else
    cout<<"new volume level: "<<tv.getVolume()<<endl;
   }

   void display(){
    tv.display();
   }
};

int main(){
   cout<<"NAME: S.M.IRTIZA | NU ID: 22K-4638"<<endl;
   TV television("samsuung");
   Remote remote(television);
   remote.togglePower();
   remote.increaseVolume();
   remote.decreaseVolume();
   remote.display();

}
```

**OUTPUT:**



```
NAME: S.M.IRTIZA | NU ID: 22K-4638
status: 1
new volume level: 1
new volume level: 0
model: samsuung
status: 1
volume: 0
```

## QUESTION # 02:

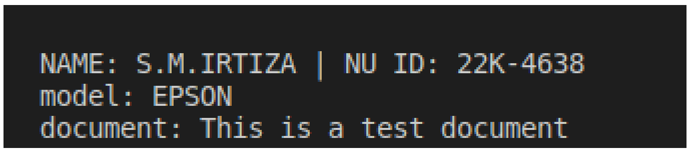
Write a c++ program in which two classes, Document and Printer. The Document class has a private member variable called content, which is a string that holds the text of the document. The class has a constructor that takes a string argument, which is used to initialize the content member. The Printer class has a private member variable called model, which is a string that holds the name of the printer model. The class has a constructor that takes a string argument, which is used to initialize the model member. The code then defines a friend function called printDocument, which takes a Document object and a Printer object as arguments. This function is declared as a friend of both the Document and Printer classes, which means it has access to the private members of both classes. Inside the printDocument function, the model of the printer and the content of the document are output to the console. In main, a Printer object is created with the name "Epson", and a Document object is created with the content "This is a test document." The printDocument function is then called with the Printer and Document objects as arguments, which prints the content of the document to the console using the specified printer.

**CODE:**

```
//NAME: S.M.IRTIZA | NU ID: 22K-4638
#include<iostream>
#include<string>
using namespace std;
class Printer;
class Document{
   string content;

   public:
   Document(){}
   Document(string a){
      content=a;
   }

   string getContent()
   {
      return this->content;
   }
   void setContent(string content)
   {
      this->content = content;
   }
    friend void printDocument(Document,Printer);

};
class Printer{
    string model;


public:
   Printer(){}
   Printer(string a){
      model=a;
   }
```

```
    friend void printDocument(Document,Printer);
         string getModel()
     {
        return this->model;
     }
    void setModel(string model)
     {
        this->model = model;
     }

};

void printDocument(Document d,Printer p){
    cout<<"model: "<<p.model<<endl;
    cout<<"document: "<<d.content<<endl;
}

int main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22K-4638"<<endl;
    Printer p("EPSON");
    Document d("This is a test document");
    printDocument(d,p);

    return 0;
}
```

**OUTPUT:**



```
NAME: S.M.IRTIZA | NU ID: 22K-4638
model: EPSON
document: This is a test document
```

## QUESTION # 03:

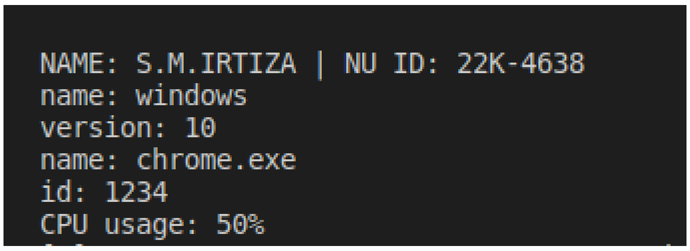
Write a c++ program in which there are two classes: OperatingSystem and Process. The OperatingSystem class represents an operating system and has a name and a version attribute. The Process class represents a running process and has a name, an id, and a cpu usage attribute. Declare two friend functions: display(OperatingSystem os) and display(Process p). These functions are declared as friends of both classes.The display(OperatingSystem os) function takes an OperatingSystem object as an argument and displays the name and version of the operating system.
The display(Process p) function takes a Process object as an argument and displays the name, id, and CPU usage of the process. In main(), we create an OperatingSystem object with the name "Windows" and the version number 10. We also create a Process object with the name "chrome.exe", the ID 1234, and a CPU usage of 50%. Call the display() function with both the OperatingSystem and Process objects as arguments. Because display() is a friend function of both classes, it has access to their private members and can display their attributes.

## CODE:

```
//NAME: S.M.IRTIZA | NU ID: 22K-4638
#include<iostream>
#include<string>
using namespace std;
class Process;
class OperatingSystem{
   string name;
   int version;

   public:
   OperatingSystem(){}
   OperatingSystem(string a,int b){
      name=a;
      version=b;
   }
   string getName() {
                return this->name;
        }

         void setName(string name) {
                this->name = name;
        }

         int getVersion() {
                return this->version;
        }

         void setVersion(int version) {
                this->version = version;
        }
   friend void display(Process p);
   friend void display(OperatingSystem os);
};
class Process{
   string name;
   int id;
   int cpuUsage;
public:
   Process(){}
   Process(string a,int b,int c){
      name=a;
      id=b;
```

```
        cpuUsage=c;
    }
   string getName()
    {
        return this->name;
    }


    void setName(string name)
    {
        this->name = name;
    }


    int getId()
    {
        return this->id;
    }


    void setId(int id)
    {
        this->id = id;
    }


    int getCpuUsage()
    {
        return this->cpuUsage;
    }


    void setCpuUsage(int cpuUsage)
    {
        this->cpuUsage = cpuUsage;
    }
    friend void display(Process p);
    friend void display(OperatingSystem os);
};
void display(OperatingSystem os){
cout<<"name: "<<os.name<<endl;
cout<<"version: "<<os.version<<endl;
}
void display(Process p){
    cout<<"name: "<<p.name<<endl;
    cout<<"id: "<<p.id<<endl;
    cout<<"CPU usage: "<<p.cpuUsage<<"%"<<endl;
}

int main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22K-4638"<<endl;
    OperatingSystem O("windows", 10);
    Process P("chrome.exe", 1234, 50);
    display(O);
    display(P);
    return 0;
}
```

**OUTPUT:**



```
NAME: S.M.IRTIZA | NU ID: 22K-4638
name: windows
version: 10
name: chrome.exe
id: 1234
CPU usage: 50%
```

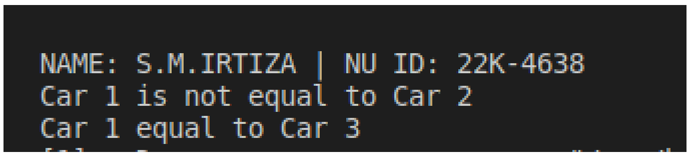
## QUESTION # 04

Car class has three private member variables - make, model, and year. The equality operator == is overloaded as a friend function of the Car class. The overloaded operator== function takes two const Car references as arguments and returns a boolean value indicating whether the two Car objects have the same make, model, and year values. In the main function, we create three Car objects and compare them using the overloaded == operator. The output of the program is:
Car 1 is not equal to Car 2.
Car 1 is equal to Car 3.

**CODE:**

```
//NAME: S.M.IRTIZA | NU ID: 22K-4638
#include<iostream>
#include<string>
using namespace std;
class Car{
    string make;
    string model;
    int year;
    public:
    Car(){}
    Car(string a,string b,int c){
        make=a;
        model=b;
        year=c;
    }

    friend bool operator==(const Car c1,const Car c2);
};
bool operator==(const Car c1,const Car c2)
{
        if(c1.make==c2.make&&c1.model==c2.model&&c1.year==c2.year)
        return true;
        else
        return false;
}
int  main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22K-4638"<<endl;
    Car c1("Toyota", "B-10",2012),c2("Suzuki", "S-11", 2011),c3("Toyota", "B-10",2012);
    if((c1==c2)==false){
        cout<<"Car 1 is not equal to Car 2"<<endl;
    }
    else{
        cout<<"Car 1 equal to Car 2"<<endl;
    }

    if((c1==c3)==false){
        cout<<"Car 1 is not equal to Car 3"<<endl;
    }
    else{
        cout<<"Car 1 equal to Car 3"<<endl;
    }

}
```

**OUTPUT:**



```
NAME: S.M.IRTIZA | NU ID: 22K-4638
Car 1 is not equal to Car 2
Car 1 equal to Car 3
```

## QUESTION# 05:

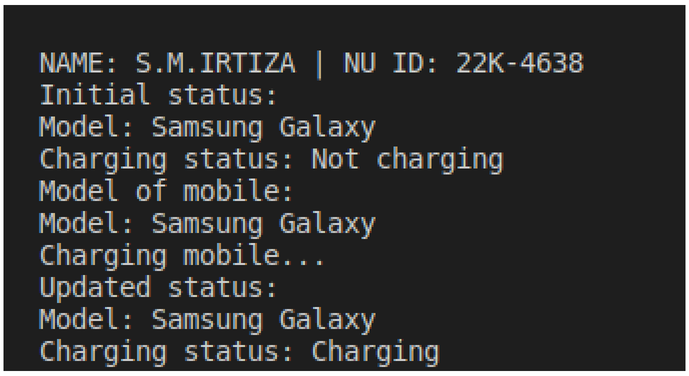
Write a c++ Program in which there are three classes. The Mobile class has a model attribute and an isCharging attribute, and it also has a friend class Display and a friend function Charger::charge(). The Charger class has a charge() method that takes a Mobile object by reference and sets its isCharging attribute to true. This method is declared as a friend function of the Mobile class. The Mobile class also has a display() method that displays the model and charging status of the mobile, and a friend class Display that can access the model attribute of the Mobile class. In the main() function, create a Mobile object m, a Charger object c, and a Display object d, call m.display() to display the initial status of the mobile, call d.showModel(m) to display the model of the mobile using the friend class Display, and call c.charge(m) to charge the mobile using the friend function Charger::charge(). Finally, call m.display() again to display the updated charging status of the mobile.

## CODE:

```
//NAME: S.M.IRTIZA | NU ID: 22K-4638
#include <iostream>
#include <string>
using namespace std;
class Display;
class Mobile;
class Charger {
public:
   void charge(Mobile& m);
};
class Mobile {
private:
   string model;
   bool isCharging;
public:
   Mobile(std::string model) : model(model), isCharging(false) {}
   friend class Display;
   friend void Charger::charge(Mobile& m);
   void display() const {
      cout << "Model: " << model << "\n";
      cout << "Charging status: " << (isCharging ? "Charging" : "Not charging") << "\n";
   }
};

void Charger::charge(Mobile& m){
      m.isCharging = true;
}
class Display {
public:
   void showModel(const Mobile& m) const {
      std::cout << "Model: " << m.model << "\n";
   }
};

int main() {
   cout<<"NAME: S.M.IRTIZA | NU ID: 22K-4638"<<endl;
   Mobile m("Samsung Galaxy");
   Charger c;
   Display d;

   std::cout << "Initial status:\n";
```

```
    m.display();

    std::cout << "Model of mobile:\n";
    d.showModel(m);

    std::cout << "Charging mobile...\n";
    c.charge(m);

    std::cout << "Updated status:\n";
    m.display();

    return 0;
}
```

**OUTPUT:**



```
NAME: S.M.IRTIZA | NU ID: 22K-4638
Initial status:
Model: Samsung Galaxy
Charging status: Not charging
Model of mobile:
Model: Samsung Galaxy
Charging mobile...
Updated status:
Model: Samsung Galaxy
Charging status: Charging
```