Assignment 10

Problem Statement

Write C++ Program with base class convert declares two variables, val1 and val2, which hold the initial and converted values, respectively. It also defines the functions getinit( ) and getconv( ), which return the initial value and the converted value. These elements of convert are fixed and applicable to all derived classes that will inherit convert. However, the function that will actually perform the conversion, compute( ), is a pure virtual function that must be defined by the classes derived from convert. The specific nature of compute ( ) will be determined by what type of conversion is taking place.

Objectives:

To learn the concept of pure virtual functions in c++ programming

Outcomes:

The performer will be able to design a program in c++ using polymorphism.

Software Requirements:

1. 64-bit Open source Linux or its derivative
2. Open Source C++ Programming tool like G++/GCC.

Theory:

A **pure virtual function** is a function that has ***the notation "= 0"*** in the declaration of that function.

class SomeClass {

```
public:
  virtual void pure_virtual() = 0;  // a pure virtual function
  // note that there is no function body
};
```

Pure virtual functions cannot have a definition inside the function declaration. Any class that **has at least one** pure virtual function is called an **abstract** class. An abstract class cannot have an instance of itself created. This also means that any class that derives from an abstract class must override the definition of the pure virtual function in the base class, and if it doesn't then the derived class becomes an abstract class as well.

Algorithm:

1. Design a base class convert
2. Create a parametrized constructor to add values to objects, convert(double i) { val1 = i; }
3. Create two functions getconv and getinit
4. Create a pure virtual function compute, virtual void compute() = 0;
5. Design to derived classes l_to_g & f_to_c , for converting liters to gallons and farenheit to celcius

6. Create definition for the pure virtual function in both the classes respectively,
7. Use a base class pointer in main(),convert *p;
8. use virtual function mechanism for conversion of units
9. Display the result

CODE

```
#include <iostream>
using namespace std;
 class convert {
 protected:
double val1; // initial value
double val2; // converted value
public:
convert(double i) { val1 = i; }
double getconv() { return val2; }
double getinit() { return val1; }
virtual void compute() = 0;
};
// Liters to gallons.
class l_to_g : public convert {
public:
l_to_g(double i) : convert(i) { }
void compute() { val2 = val1 / 3.7854;
}
};
// Fahrenheit to Celsius
class f_to_c : public
convert {
public:
f_to_c(double i) : convert(i) { }
 void compute() {
val2 = (val1-32) / 1.8;
```

```
}
};
int main() {
convert *p; // pointer to base class
 l_to_g lgob(4);
f_to_c fcob(70);
// use virtual function mechanism to convert
 p = &lgob;
cout<< p->getinit() << " liters is ";
p->compute();
cout<< p->getconv()<< " gallons\n"; // l_to_g
p = &fcob;
cout<< p->getinit() << " in Fahrenheit is ";
p->compute();
cout<< p->getconv()<< " Celsius\n"; // f_to_c
return 0; }
```

Sample Output

```
dypiemr-@dypiemr:~$ g++ virtual.cpp
dypiemr-@dypiemr:~$ ./a.out
4 liters is 1.05669 gallons
70 in Fahrenheit is 21.1111 Celsius
```