**Batch: E2 Roll No.: 16010123325**

**Experiment / assignment / tutorial No.**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE : Implementing a billing application using OOP concepts using C++** |

**AIM:** Develop a C++ application that generates an Electricity Bill using a Consumer class.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO1:Apply the features of object oriented programming languages. (C++ and

Java)

CO2:Explore arrays, vectors, classes and objects in C++ and Java **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. E. Balagurusamy, “Programming with Java”, McGraw-Hill.
2. E. Balagurusamy, “Object Oriented Programming with C++”, McGraw-Hill.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Class Definition:

The Consumer class should encapsulate the following information:

* consumer\_no (integer): Unique identification number for the consumer.
* consumer\_name (string): Name of the consumer.
* previous\_reading (integer): Meter reading from the previous month.
* current\_reading (integer): Meter reading from the current month.
* connection\_type (string): Type of electricity connection (domestic or commercial).
* calculate\_bill (member function): This function should calculate the electricity bill amount based on the connection\_type and the number of units consumed (current reading - previous reading). The function should utilize a tiered pricing structure as specified below:

**Tiered Pricing:**

***Domestic Connection:***

First 100 units: Rs. 1 per unit

101-200 units: Rs. 2.50 per unit

201-500 units: Rs. 4 per unit

Above 501 units: Rs. 6 per unit

***Commercial Connection:***

First 100 units: Rs. 2 per unit

101-200 units: Rs. 4.50 per unit

201-500 units: Rs. 6 per unit

Above 501 units: Rs. 7 per unit

Additional Considerations:

* The application should prompt the user to enter the details for a consumer (consumer number, name, previous reading, current reading, and connection type).
* The calculate\_bill function should implement logic to determine the applicable unit charges based on the connection type and the number of units consumed within each tier.
* The application should display a clear breakdown of the bill, including the consumer details, number of units consumed, charge per unit for each tier, and the total bill amount.

**Algorithm:**

Input:

* Consumer number (cno)
* Consumer name (name)
* Previous reading (prev)
* Current reading (curr)
* Connection type (type)

Steps:

* Calculate units consumed: units = curr - prev
* Determine bill amount based on connection type and units consumed:

For domestic connections:

* If units <= 100, bill = 1.0 \* units
* If 100 < units <= 200, bill = 100 \* 1.0 + 2.5 \* (units-100)
* If 200 < units <= 500, bill = 100 \* 1.0 + 2.5 \* 100 + 4.0 \* (units-200)
* If units > 500, bill = 100 \* 1.0 + 2.5 \* 100 + 4.0 \* 300 + 6.0 \* (units-500)

For non-domestic connections:

* If units <= 100, bill = 2.0 \* units
* If 100 < units <= 200, bill = 100 \* 2.0 + 4.5 \* (units-100)
* If 200 < units <= 500, bill = 100 \* 2.0 + 200 \* 4.5 + 6.0 \* (units-200)
* If units > 500, bill = 100 \* 2.0 + 100 \* 4.5 + 300 \* 6.0 + 7.0 \* (units-500)
* Display bill details:
* Consumer number
* Consumer name
* Previous reading
* Current reading
* Connection type
* Units consumed
* Total bill amount

**Implementation details:**

#include <bits/stdc++.h>

using namespace std;

class Consumer {

    public:

        int consumer\_no;

        string consumer\_name;

        int previous\_reading;

        int current\_reading;

        string connection\_type;

    Consumer(int cno, string name, int prev, int curr, string type):consumer\_no(cno), consumer\_name(name), previous\_reading(prev), current\_reading(curr), connection\_type(type) {}

    double calculate\_bill() {

        int units = current\_reading - previous\_reading;

        double bill = 0.0;

        if(connection\_type=="domestic") {

            if(units <= 100) {

                bill = 1.0\* units;

            }

            else if(units > 100 && units <= 200) {

                bill = 100 \* 1.0 + 2.5 \* (units-100);

            }

            else if(units>200 && units<=500) {

                bill = 100 \* 1.0 + 2.5 \* 100 + 4.0 \* (units-200);

            } else {

                bill = 100 \* 1.0 + 2.5 \* 100 + 4.0 \* 300 + 6.0 \* (units-500);

            }

        }

        else {

            if(units <= 100) {

                bill = 2.0 \* units;

            } else if(units > 100 && units <= 200) {

                bill = 100 \* 2.0 + 4.5 \* (units-100);

            } else if(units > 200 && units <= 500) {

               bill = 100 \* 2.0 + 200 \* 4.5 + 6.0 \* (units-200);

            } else {

                bill = 100 \* 2.0 + 100 \* 4.5 + 300 \* 6.0 + 7.0 \* (units-500);

            }

        }

        return bill;

    }

    void display\_bill()

    {

        int units = current\_reading - previous\_reading;

        cout <<""<<endl;

        cout << "DETAILS:"<<endl;

        cout << "Consumer Number: " << consumer\_no << '\n';

        cout << "Consumer Name: " << consumer\_name << '\n';

        cout << "Previous Reading: " << previous\_reading << '\n';

        cout << "Current Reading: " << current\_reading << '\n';

        cout << "Connection Type: " << connection\_type << '\n';

        cout << "Units Consumed: " << units << '\n';

        cout << "Total Bill Amount: Rs. " << calculate\_bill() << '\n';

    }

};

int main()

{

    int cno, prev, curr;

    string name, type;

    cout << "Enter Consumer Number: ";

    cin >> cno;

    cout << "Enter Consumer Name: ";

    cin.ignore();

    getline(cin, name);

    cout << "Enter previous reading: ";

    cin >> prev;

    cout << "Enter Current Reading: ";

    cin >> curr;

    cout << "Enter connection type: ";

    cin.ignore();

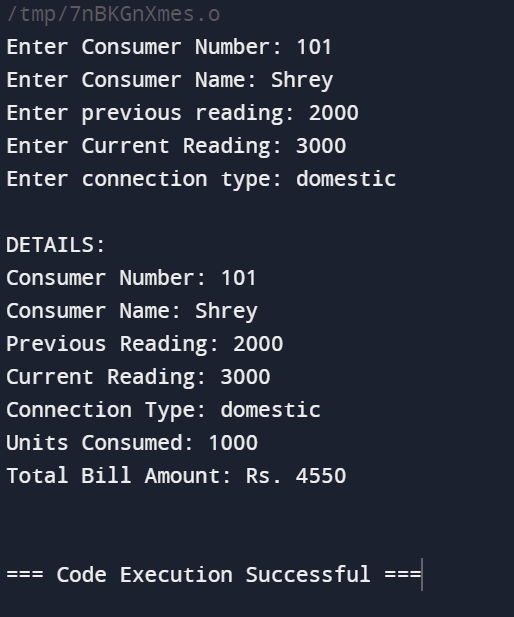
    getline(cin,type);

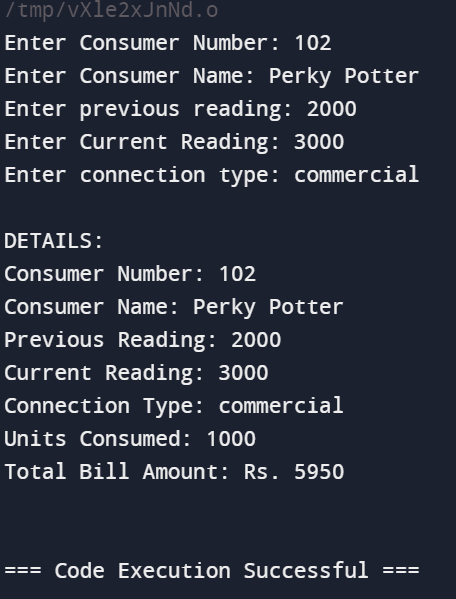
    Consumer obj(cno, name, prev, curr, type);

    obj.display\_bill();

}

**Output:**





**Conclusion:**

The Electricity Bill Calculator program successfully calculates and displays the total bill amount for a consumer based on their connection type and units consumed, providing a simple and efficient way to manage and calculate electricity bills.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions:**

Q.1 Explain the concept of constructors and destructors in C++.

Ans:

**Constructors:**

* Constructors are special member functions of a class that are executed automatically whenever an object of that class is created.
* They are used to initialize the data members of the class and set them to their default or specified values.
* Constructors have the same name as the class and do not have a return type, not even void.
* There are different types of constructors in C++, such as default constructor, parameterized constructor, copy constructor, etc.

**Destructors:**

* Destructors are special member functions of a class that are executed automatically whenever an object of that class goes out of scope or is explicitly destroyed using the delete operator.
* They are used to deallocate any memory or resources that were allocated by the constructor or during the lifetime of the object.
* Destructors have the same name as the class, preceded by a tilde (~) symbol, and do not have any parameters or a return type.
* Destructors are important for ensuring that objects are cleaned up properly and that resources are not leaked.

Q.2 Write the output of following program with suitable explanation

#include<iostream>

**using** **namespace** std;

**class** Test

{

**static** **int** i;

**int** j;

};

**int** Test::i;

**int** main()

{

cout << **sizeof**(Test);

**return** 0;

}

**Output:**

**4**

The output of the program is 4. This is because sizeof(Test) measures the size of an object of class Test. The class contains a non-static member int j, which typically takes up 4 bytes. Static members are not included in the object's size calculation.

Q.3 Explain all the applications of the scope resolution operator in C++.

Ans:

* **Accessing Global Variables**: To access global variables when a local variable has the same name.
* **Accessing Static Members**: To access static members (variables or functions) of a class.
* **Defining Functions Outside the Class**: To define functions outside a class when they are declared inside the class.
* **Accessing Namespace Members**: To access members (variables, functions, or classes) of a namespace.
* **Overloading Operators**: To overload operators for user-defined classes.
* **Specifying Namespace**: To specify a namespace when using a class or function from that namespace.
* **Defining Nested Classes**: To define nested classes outside the outer class.
* **Accessing Enum Members**: To access members of an enumeration.