ELECTRICITY BILL GENERATOR

PROJECT REPORT

Submitted by: Lakshya Bisht
SAP ID: 590027070
Course Name: Programming in C
Branch: B. Tech CSE

Under the Guidance of: Dr. Prashant Trivedi
Submission Date: 01/12/2025
School of Computer Science
University of Petroleum and Energy Studies

ABSTRACT

This project titled “Electricity Bill Generator” is developed using the C programming language. The main aim of this project is to automate the electricity bill generation process based on current tariff structure for residential consumers under **Uttarakhand Power Corporation Limited (UPCL).**

The program takes customer inputs such as ID, name, number of units consumed, connected load, and late payment status. Based on this information, it calculates energy charges, fixed charges, surcharge, electricity duty, and total bill amount, and generates a formatted bill receipt. The program also features searching of customer records by ID and name.

File handling is implemented so that customer data remains stored permanently even after the program is closed. The program also utilizes concepts of Loops, Conditional Statements, Structures, Functions and Pointers.

PROBLEM DEFINITION

In manual electricity billing systems, calculations are done by hand or using basic tools, which may lead to errors. Managing records manually, on paper, is time consuming, difficult to maintain and also leads to paper wastage.

Problems Identified:

- Billing calculation is slow and error-prone.
- Customer records may be lost, due to negligence.
- Searching records manually takes more time.
- No automation of slab-based bill calculation.

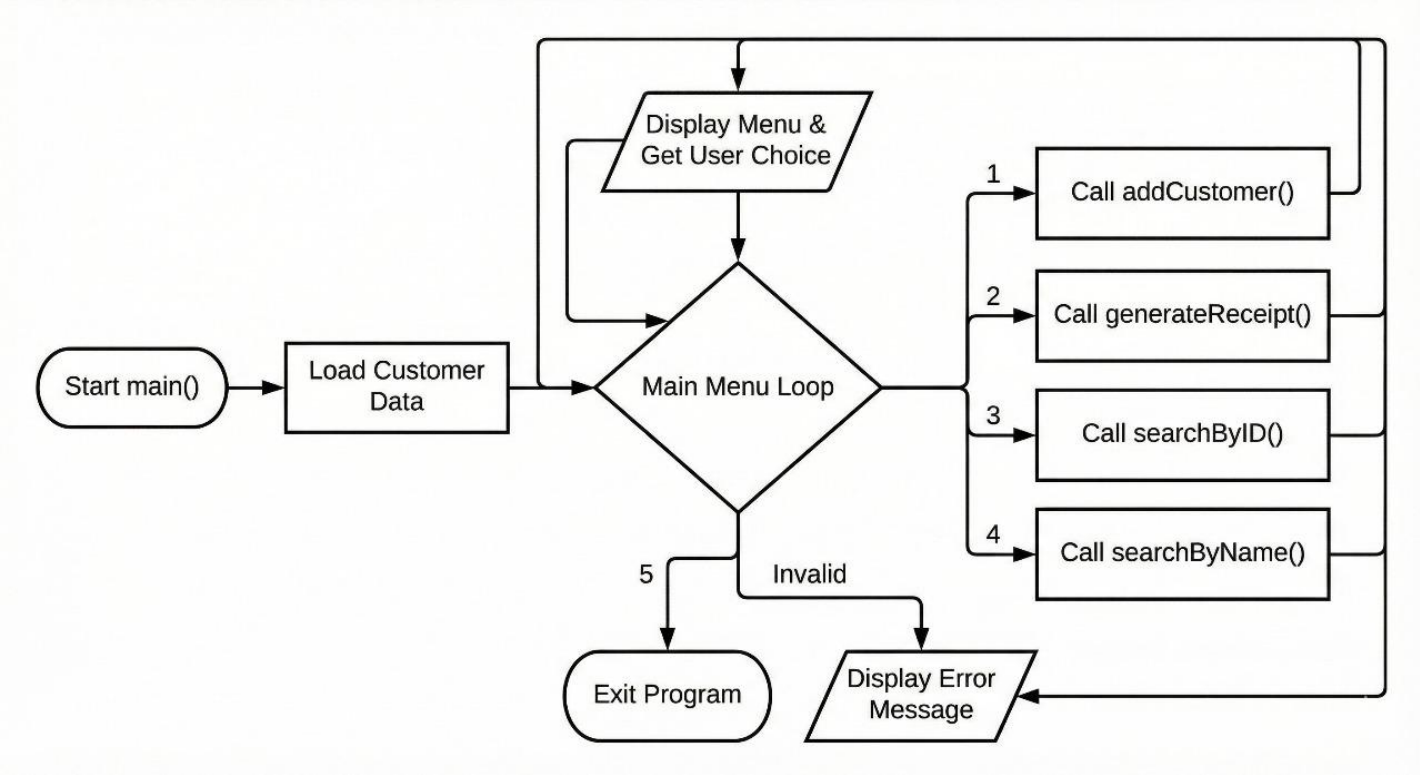
Objective of the Project:

To design a program that:

- Calculates bills accurately based on slabs-wise tariff.
- Stores data permanently, using file handling.
- Searches customers easily, by ID or name.
- Displays a real-like formatted bill receipt.
- Reduces human error and human effort.

SYSTEM DESIGN

Flowchart:



Algorithms:

Algorithm for main() Function:

1. Start
2. Call loadFromFile() to retrieve previous data
3. Start WHILE (True) Loop.
4. Display the Main Menu (Options 1 to 5).
5. Prompt user for choice.
6. Validate Input**:** If non-number input, repeat the menu (Go to step 4).
7. Use switch statement based on choice:
    - Case 1: Call addCustomer().
    - Case 2: Call generateReceipt().
    - Case 3: Call searchByID().
    - Case 4: Call searchByName().
    - Case 5: Call exit(0).
    - Default: Print "Invalid choice."
8. End loop
9. End

Algorithm 1: SAVE TO FILE

1. Start
2. Open file in write binary mode
3. If file does not open, display error
4. Write count
5. Write customer records
6. Close file
7. End

Algorithm 2: LOAD FROM FILE

1. Start
2. Open file in read mode
3. If file not found, exit function
4. Read count
5. Read all records
6. Close file
7. End

Algorithm 3: CALCULATE BILL

1. Start
2. Accept units, load, late payment flag
3. Assign fixed charge based on load
4. Calculate energy charge using slabs
5. Add duty (15%)
6. Add surcharge (0.10 per unit)
7. If late payment, add 1.25%
8. Store in structure
9. Return bill
10. End

Algorithm 4: ADD CUSTOMER

1. Start
2. Check storage limit
3. Input ID
4. Check duplicate
5. Input name
6. Input units
7. Input load
8. Input late payment
9. Call bill function
10. Store record
11. Save to file
12. End

Algorithm 5: SEARCH BY ID

1. Input ID
2. Compare with every record
3. If match found – display
4. Else show not found
5. End

Algorithm 6: SEARCH BY NAME

1. Input name
2. Compare using string function
3. Display record if found
4. Else show error
5. End

Algorithm 7: GENERATE RECEIPT

1. Enter Customer ID
2. Display formatted bill
3. Show all charges
4. End

IMPLEMENTATION DETAILS

User-Defined Functions:

The project uses modular programming, separate functions for different tasks:

loadFromFile() and saveToFile(): Handle data storage.
calculateBill(): Contains all the financial logic.
addCustomer(): Manages user input and data saving.
searchByID(), searchByName(), generateReceipt(): Handle data retrieval and display.

Structure Used:

```
typedef struct
{
   int id;

   char name[50];
   float units;
   float load;
   float energyCharge, fixedCharge, duty, surcharge, totalBill;
} Customer;
```

File Handling Example:

```
// Saving data to a binary file
void saveToFile()
{
   FILE *fp=fopen("customers.dat","wb"); // 'wb' for Write Binary
   // ... error checking
   fwrite(&count, sizeof(int), 1, fp); // Save the total count first
   fwrite(customers, sizeof(Customer), count, fp); // Save the array data
   fclose(fp);
}
```

The use of fread and fwrite allows the entire array of structures to be stored and retrieved efficiently as raw data.

Slab Calculation Logic:

```
// Slab-wise energy charge calculation

if(units <= 100)
   energyCharge = units * 3.65;
else if(units <= 200)
   energyCharge = (100*3.65) + ((units-100)*5.25);
else if(units <= 400)
   energyCharge = (100*3.65) + (100*5.25) + ((units-200)*7.15);
else
   energyCharge = (100*3.65) + (100*5.25) + (200*7.15) + ((units-400)*7.80);
```

Pass-by-Reference:

Instead of returning a single value, the function uses a **pointer (Customer *c)** to update all calculated components directly inside the original Customer variable.

```
void calculateBill(float units, float load, int latePayment, Customer *c)
{
   //...calculations...

  c->energyCharge = energyCharge;
  c->fixedCharge = fixedCharge;
  c->duty = duty;
  c->surcharge = surcharges;
  c->totalBill = totalBill;
}
```

Menu System:

```
printf("1. Add Customer\n");
printf("2. Generate Bill\n");
printf("3. Search by ID\n");
printf("4. Search by Name\n");
printf("5. Exit\n");
```

TESTING & RESULTS

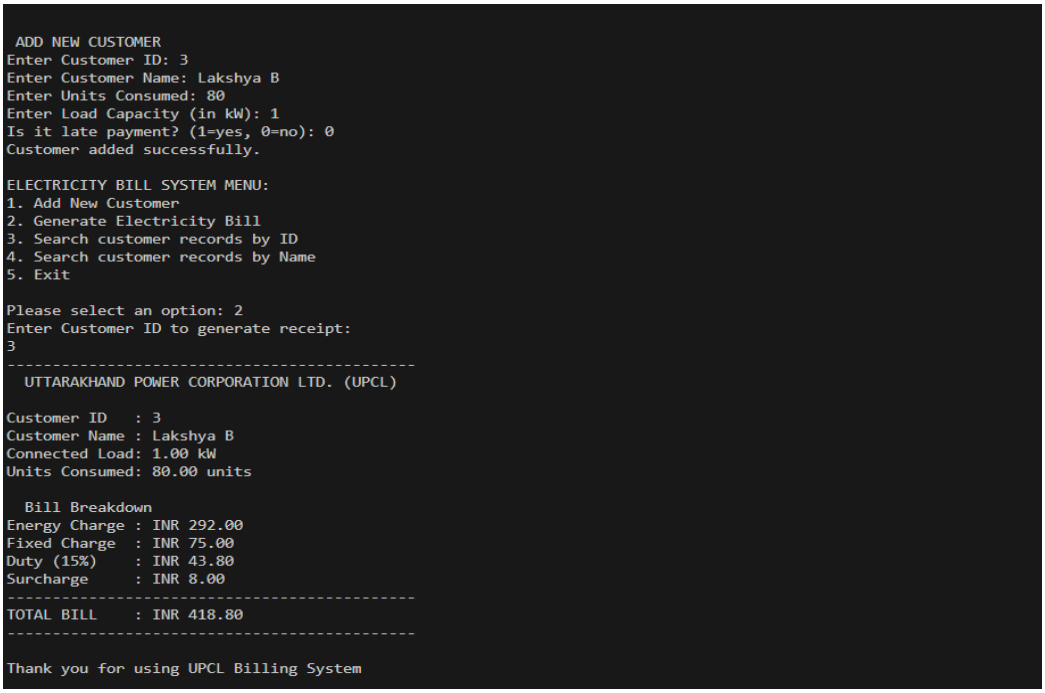
Performed 3 cases to verify the slab calculations, fixed charges, and late payment fee.

Test Case	Units Consumed	Connected Load (kW)	Late Payment	Expected Result (INR)	Actual Output (INR)	Verification
1. Low Slab	80	1.0	No (0)	(80*3.65 + 75.0) + (80*0.10) + Duty + Charges	418.80	**PASS**
2. Mid Slab	150	2.5	No (0)	(100*3.65 + 50*5.25) + (2.5*85.0) + Duty + Charges	949.13	**PASS**
3. High Slab + Fee	500	5.0	Yes (1)	3100 + (100 * 5.0) + Duty + Charges + 1.25%	4166.44	**PASS**

Result:

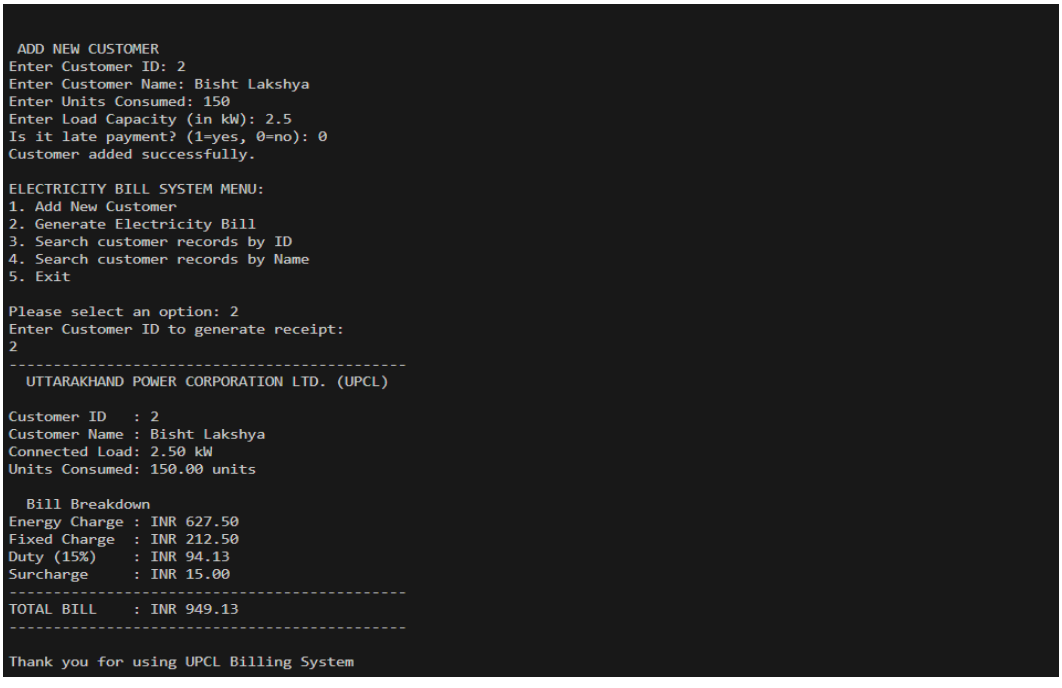
All functions are executed successfully without crash. Bill values change as per slabs and file saving works correctly.

Screenshot: Test case 1



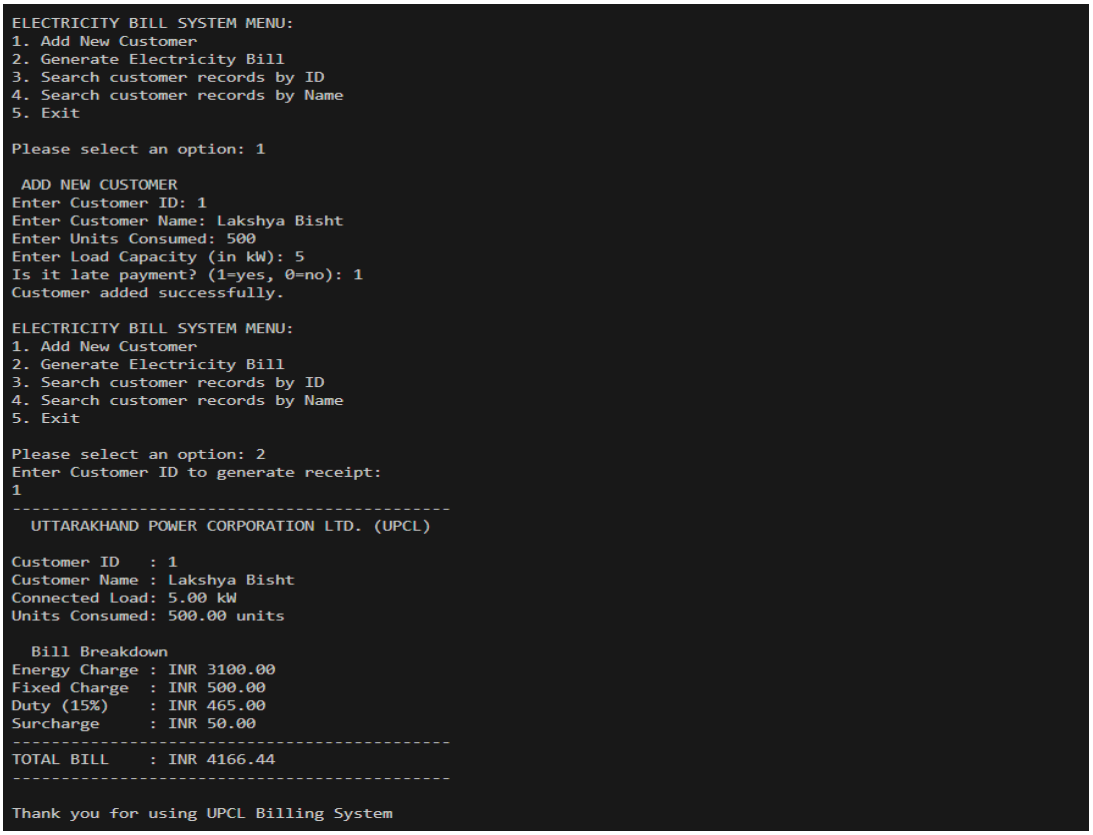
```
 ADD NEW CUSTOMER
Enter Customer ID: 3
Enter Customer Name: Lakshya B
Enter Units Consumed: 80
Enter Load Capacity (in kW): 1
Is it late payment? (1=yes, 0=no): 0
Customer added successfully.

ELECTRICITY BILL SYSTEM MENU:
1. Add New Customer
2. Generate Electricity Bill
3. Search customer records by ID
4. Search customer records by Name
5. Exit

Please select an option: 2
Enter Customer ID to generate receipt:
3
--------------------------------------------
  UTTARAKHAND POWER CORPORATION LTD. (UPCL)

Customer ID   : 3
Customer Name : Lakshya B
Connected Load: 1.00 kW
Units Consumed: 80.00 units

  Bill Breakdown
Energy Charge : INR 292.00
Fixed Charge  : INR 75.00
Duty (15%)    : INR 43.80
Surcharge     : INR 8.00
--------------------------------------------
TOTAL BILL    : INR 418.80
--------------------------------------------

Thank you for using UPCL Billing System
```

Screenshot: Test case 2



```
 ADD NEW CUSTOMER
Enter Customer ID: 2
Enter Customer Name: Bisht Lakshya
Enter Units Consumed: 150
Enter Load Capacity (in kW): 2.5
Is it late payment? (1=yes, 0=no): 0
Customer added successfully.

ELECTRICITY BILL SYSTEM MENU:
1. Add New Customer
2. Generate Electricity Bill
3. Search customer records by ID
4. Search customer records by Name
5. Exit

Please select an option: 2
Enter Customer ID to generate receipt:
2
--------------------------------------------
  UTTARAKHAND POWER CORPORATION LTD. (UPCL)

Customer ID   : 2
Customer Name : Bisht Lakshya
Connected Load: 2.50 kW
Units Consumed: 150.00 units

  Bill Breakdown
Energy Charge : INR 627.50
Fixed Charge  : INR 212.50
Duty (15%)    : INR 94.13
Surcharge     : INR 15.00
--------------------------------------------
TOTAL BILL    : INR 949.13
--------------------------------------------

Thank you for using UPCL Billing System
```

Screenshot: Test case 3



```
ELECTRICITY BILL SYSTEM MENU:
1. Add New Customer
2. Generate Electricity Bill
3. Search customer records by ID
4. Search customer records by Name
5. Exit

Please select an option: 1

 ADD NEW CUSTOMER
Enter Customer ID: 1
Enter Customer Name: Lakshya Bisht
Enter Units Consumed: 500
Enter Load Capacity (in kW): 5
Is it late payment? (1=yes, 0=no): 1
Customer added successfully.

ELECTRICITY BILL SYSTEM MENU:
1. Add New Customer
2. Generate Electricity Bill
3. Search customer records by ID
4. Search customer records by Name
5. Exit

Please select an option: 2
Enter Customer ID to generate receipt:
1
--------------------------------------------
  UTTARAKHAND POWER CORPORATION LTD. (UPCL)

Customer ID   : 1
Customer Name : Lakshya Bisht
Connected Load: 5.00 kW
Units Consumed: 500.00 units

  Bill Breakdown
Energy Charge : INR 3100.00
Fixed Charge  : INR 500.00
Duty (15%)    : INR 465.00
Surcharge     : INR 50.00
--------------------------------------------
TOTAL BILL    : INR 4166.44
--------------------------------------------

Thank you for using UPCL Billing System
```

Detailed Calculation for Test Case 3 (500 Units, 5.0 kW, Late)

Energy Charge:

Slab 1 (0-100): 100 * 3.65 = 365.00
Slab 2 (101-200): 100 * 5.25 = 525.00
Slab 3 (201-400): 200 * 7.15 = 1430.00
Slab 4 (401+): (500 – 100) * 7.80 = 780.00
Total Energy Charge: INR 3100.00

1. **Fixed Charge:** 5.0 * 100.00 = INR 500.00
2. **Duty (15%):** 3100.00 * 0.15 = INR 465.00
3. **Surcharge (Rs. 0.10/unit):** 500 * 0.10 = 50.00
4. **Subtotal:** 3100.00 + 500.00 + 465.00 + 50.00 = INR 4115.00
5. **Late Fee (1.25%):** 4115.00 * 0.0125 = INR 51.44
6. **Total Bill:** 4115.00 + 51.44 = 4166.44

CONCLUSION & FUTURE WORK

Conclusion

This project successfully demonstrates how core C programming concepts such as structures, file handling, functions and conditional statements can be used to develop a real-world project. The electricity bill generator is simple to use, works efficiently and generates accurate results. The system also ensures data safety and ease of search.

Future Works

The project can be extended in future by adding the following features:

1. Date and Time display on generated bills.
To display of date and time on generated bill for better record keeping.

2. Password Protected Access
A login system can be added allowing only authorized users to modify records.

3. Graphical User Interface (GUI)
A graphical interface can be developed to make the system more user-friendly.

4. Online Payment Feature
An option for online bill payment can make the system more practical.

5. Monthly Billing History
Future versions may store bill details month-wise so that old bill records can be viewed and analyzed easily.

REFERENCES

1. **Course Material:** Notes and concepts covered in the classroom lectures.
2. **Let Us C** by Yashwant Kanetkar: To study concepts in-depth.
3. **Tariff Source:** Official UPCL Residential Tariff Order for 2025, Government of Uttarakhand.

   Revised Rate Schedules in conformity of tariff order dated 11-04-2025. | Uttarakhand Power Corporation Limited | India
4. **Youtube:** C programming lectures to understand and develop concepts.

APPENDIX

Appendix A

Complete Source Code

```
/*
PROJECT TOPIC:
Electricity Bill Generator(with Different slabs)

PROJECT OVERVIEW:
In this project, based on C programming Language, I am working on generating a realistic
Electicity Monthly Bill, as per the Current Tariff
and all Other Charges applicable for Residential connection under UPCL in Dehradun,
Uttarakhand. Apart from calculating slab-wise energy charges,
the program also stores customer details, allows searching, and displays a formatted bill. File
handling is used so that customer records
remain saved even after the program is closed and opened again. Basic search options (by ID
and by name) are also provided for ease of usage.
Concepts Used: Structures, File Handling, Dynamic Memory allocation, Functions, Pointers,
Searching.

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//Structure to store customer information
typedef struct
{
   int id;
   char name[50];
   float units;
   float load; //Connected load in kW
   float energyCharge;
   float fixedCharge;
   float duty; //Electricity duty
   float surcharge;
   float totalBill;
} Customer;
```

```
Customer *customers = NULL;
int count=0;

//Function to save data to file
void saveToFile()
{
   if(customers == NULL)
   return;

   FILE *fp=fopen("customers.dat","wb");
   if(fp == NULL)
   {
      printf("Error: Cannot save file.\n");
      return;
   }
   fwrite(&count, sizeof(int), 1, fp);
   fwrite(customers, sizeof(Customer), count, fp);

   fclose(fp);
}

// Load data from file when program starts
void loadFromFile()
{
   FILE *fp = fopen("customers.dat","rb");
   if(fp == NULL)
   return;

   fread(&count, sizeof(int), 1, fp);

   customers = (Customer *)malloc(count * sizeof(Customer));
   if (customers == NULL)
   {
      printf("Memory allocation failed\n");
      fclose(fp);
      exit(1);
   }

   fread(customers, sizeof(Customer), count, fp);

   fclose(fp);
}

//Calculates bill breakdown and total bill
void calculateBill(float units, float load, int latePayment, Customer *c)
{
   float energyCharge=0, totalBill=0, surcharges=0, fixedCharge=0, duty=0;
```

```
  //Fixed charge based on load capacity
  if(load <= 1)
  fixedCharge = 75.0 * load; //75/kW for upto 1kW
  else if(load <= 4)
  fixedCharge = 85.0 * load; //85/kW for upto 4kW
  else
  fixedCharge = 100 * load; //100/kW for greater than 4kW

  //Slab-wise energy charge calculation
  if(units <= 100) //First 100 units @3.65/unit
  {
    energyCharge = units * 3.65;
  }
  else if(units <= 200) //Next 100 units @5.25/unit, including fixed charge for first 100 units
  {
    energyCharge = (100 * 3.65) + ((units - 100) * 5.25);
  }
  else if(units <= 400) //Next 200 units @7.15/unit, including fixed charges for first 200 units
  {
    energyCharge = (100 * 3.65) + (100 * 5.25) + ((units - 200) * 7.15);
  }
  else //After 400 units @7.80/unit, including fixed charges for first 400 units
  {
    energyCharge = (100 * 3.65) + (100 * 5.25) + (200 * 7.15) + ((units - 400) * 7.80);
  }

  //Duty charges 15% of energy charges
  duty = energyCharge * 0.15;

  //Surcharges  Rs. 0.10 per unit
  surcharges = units * 0.10;

  //Sum of all components = Total bill
  totalBill = energyCharge + fixedCharge + duty + surcharges ;

  //Late payment fee 1.25% (if applicable)
  if(latePayment == 1)
  totalBill += totalBill * 0.0125;

  //Store all calculated components back in structure
  c->energyCharge = energyCharge;
  c->fixedCharge = fixedCharge;
  c->duty = duty;
  c->surcharge = surcharges;
  c->totalBill = totalBill;
}
```

```
//Function to add a new customer
void addCustomer()
{
   Customer c;
   int latePayment;
   int tempChar;

   customers = (Customer *)realloc(customers, (count + 1) * sizeof(Customer));
   if (customers == NULL)
   {
      printf("Memory error\n");
      exit(1);
   }

   printf("\n ADD NEW CUSTOMER \n");

   printf("Enter Customer ID: ");
   scanf("%d", &c.id);
   for(int i=0; i<count; i++)
   {
      //Duplicate ID check
      if(customers[i].id == c.id)
      {
         printf("Error: Customer ID already exists.\n");
         return;
      }
   }

   //Clear the "Enter" key left by scanf so fgets works
   while ((tempChar=getchar()) != '\n' && tempChar != EOF);

   printf("Enter Customer Name: ");
   fgets(c.name, sizeof(c.name), stdin);
   //Removes new line at the end of name
   c.name[strcspn(c.name,"\n")]='\0';

   printf("Enter Units Consumed: ");
   scanf("%f", &c.units);
   if(c.units < 0)
   {
      printf("Invalid input. Units cannot be less than zero.\n");
      return;
   }

   printf("Enter Load Capacity (in kW): ");
   scanf("%f", &c.load);
```

```
    if(c.load <= 0)
    {
        printf("Invalid input. Load capacity must be greater than zero.\n");
        return;
    }

    printf("Is it late payment? (1=yes, 0=no): ");
    scanf("%d", &latePayment);
    if(latePayment != 0 && latePayment != 1)
    {
        printf("Invalid input. Please enter 1(late payment) or not late(0).\n");
        return;
    }

    //Calculate bill and save it to structure
    calculateBill(c.units, c.load, latePayment, &c);

    customers[count]=c;
    count++;

    saveToFile(); //Save input data to file

    printf("Customer added successfully.\n");
}

//Function to search customer records by ID
void searchByID()
{
    int id, found=0;
    printf("Enter Customer ID to search: ");
    scanf("%d", &id);

    for(int i=0; i<count; i++)
    {
        if(customers[i].id == id)
        {
            printf("\nCustomer Found:\n");
            printf("ID: %d\n", customers[i].id);
            printf("Name: %s\n", customers[i].name);
            printf("Units: %.2f\n", customers[i].units);
            printf("Load: %.2f kW\n", customers[i].load);
            printf("Total Bill: INR %.2f\n", customers[i].totalBill);
            found=1;
            break;
        }
    }
```

```
    if(found == 0)
    printf("Customer not found.\n");
}

//Function to search customer records by name
void searchByName()
{
    char searchName[50];
    int found=0;
    int tempChar;

    while ((tempChar=getchar()) != '\n' && tempChar != EOF);

    printf("Enter Customer Name to search: ");
    fgets(searchName, sizeof(searchName), stdin);
    searchName[strcspn(searchName, "\n")]='\0';

    for(int i=0; i<count; i++)
    {
        if(strcmp(customers[i].name, searchName) == 0)
        {
            printf("\nCustomer Found:\n");
            printf("ID: %d\n", customers[i].id);
            printf("Name: %s\n", customers[i].name);
            printf("Units: %.2f\n", customers[i].units);
            printf("Load: %.2f kW\n", customers[i].load);
            printf("Total Bill: INR %.2f\n", customers[i].totalBill);

            found=1;
        }
    }

    if(found == 0)
    printf("No customer found with the entered name.\n");
}

//Display formatted electricity bill
void generateReceipt()
{
    int id, found=0;
    printf("Enter Customer ID to generate receipt: ");
    scanf("%d", &id);

    for (int i=0; i<count; i++)
    {
        if (customers[i].id == id)
        {
```

```
            printf("-------------------------------------------\n");
            printf("  UTTARAKHAND POWER CORPORATION LTD. (UPCL) \n\n");
            printf("Customer ID   : %d\n", customers[i].id);
            printf("Customer Name : %s\n", customers[i].name);
            printf("Connected Load: %.2f kW\n", customers[i].load);
            printf("Units Consumed: %.2f units\n", customers[i].units);
            printf("\n     Bill Breakdown\n");
            printf("Energy Charge : INR %.2f\n", customers[i].energyCharge);
            printf("Fixed Charge  : INR %.2f\n", customers[i].fixedCharge);
            printf("Duty (15%%)    : INR %.2f\n", customers[i].duty);
            printf("Surcharge     : INR %.2f\n", customers[i].surcharge);
            printf("-------------------------------------------\n");
            printf("TOTAL BILL    : INR %.2f\n", customers[i].totalBill);
            printf("-------------------------------------------\n");
            printf("\nThank you for using UPCL Billing System\n");

            found=1;
            break;
        }
    }

    if(found == 0)
    printf("Customer data not found\n");
}

//Main menu for users to interact with available options
int main()
{
    loadFromFile(); //Load customer data from file

    int choice;
    while(1)
    {
        printf("\n\nELECTRICITY BILL SYSTEM MENU:\n");
        printf("1. Add New Customer\n");
        printf("2. Generate Electricity Bill\n");
        printf("3. Search customer records by ID\n");
        printf("4. Search customer records by Name\n");
        printf("5. Exit\n");

        printf("\nPlease select an option: ");
        if(scanf("%d", &choice) != 1) //If customer did not entered a number
        {
            printf("Invalid input. Please enter number (1-5)\n");
            while (getchar() != '\n');
            continue; //Go back to the start of the while loop
        }
```

```
    //switch case to call functions based on user's choice
    switch(choice)
    {
      case 1:
        addCustomer();
        break;

      case 2:
        generateReceipt();
        break;

      case 3:
        searchByID();
        break;

      case 4:
        searchByName();
        break;

      case 5:
        saveToFile();
        free(customers);
        printf("Exiting program...\n");
        exit(0);

      default:
        printf("Invalid choice...Try again\n");
    }
  }

  return 0;
}
```

Appendix B
Sample Input & Output
Input:

```
ELECTRICITY BILL SYSTEM MENU:
1. Add New Customer
2. Generate Electricity Bill
3. Search customer records by ID
4. Search customer records by Name
5. Exit

Please select an option: 1

 ADD NEW CUSTOMER
Enter Customer ID: 1
Enter Customer Name: Lakshya Bisht
Enter Units Consumed: 500
Enter Load Capacity (in kW): 5
Is it late payment? (1=yes, 0=no): 1
Customer added successfully.
```

Output:

```
---------------------------------------------------------------
  UTTARAKHAND POWER CORPORATION LTD. (UPCL)

Customer ID   : 1
Customer Name : Lakshya Bisht
Connected Load: 5.00 kW
Units Consumed: 500.00 units

  Bill Breakdown
Energy Charge : INR 3100.00
Fixed Charge  : INR 500.00
Duty (15%)    : INR 465.00
Surcharge     : INR 50.00
---------------------------------------------------------------
TOTAL BILL    : INR 4166.44
---------------------------------------------------------------

Thank you for using UPCL Billing System
```

Appendix C
Data File Description

The customer records are stored in a file named as:
customers.dat

This file stores:
1. Total number of customers
2. All customer records using fwrite() and fread()
3. Data remains saved even after closing the program