Travel Budget Management System

Research:

Sources-

Youtube, Google websites- geeksforgeeks.org, www.programiz.com, www.w3schools.com, www.tutorialspoint.com

Analysis:

1. **Main Function**:
    - **Variables**:
        - Strings for customer details (ID, name, phone number, type).
        - Integer for units_consumed.
        - Float for bill_amount.
    - **Input**: Collects customer details and units consumed.
    - **Bill Calculation**: Calls calculate_bill() and stores the result in bill_amount.
    - **Output**: Displays customer details, units consumed, and the final bill amount.
2. **Code Flow**:
    1. Gathers customer info and consumption data.
    2. Computes the bill using the calculate_bill() function.
    3. Outputs the bill summary.

Strengths:

- Modularity: Separated logic for easy maintenance.
- Clear logic: Simple if-else structure for tiered pricing.
- Scalability: Can easily add more customer types or tariffs.

Ideate:

Concept Applications:

Budget Management System- This program is a simple vacation budget planner that allows users to:

1. Set their destination

2. Set their total budget

3. Allocate funds to different categories (transportation, accommodation, food, activities, miscellaneous)

4. Display their budget breakdown

Build:

Algorithm (for base code):

1.Start

2. Input Data

- Prompt user for Customer ID.
- Prompt user for Customer Name.
- Prompt user for Phone Number.
- Prompt user for Customer Type (options: "domestic" or "commercial").
- Prompt user for Units Consumed.

3. Calculate Bill Amount

- Define a function calculate_bill(units_consumed, customer_type):
  - Initialize bill_amount to 0.
  - If customer_type is "domestic":
    - If units_consumed <= 100:
      - Set bill_amount = units_consumed * 0.50.
    - Else If units_consumed <= 200:
      - Set bill_amount = (100 * 0.50) + ((units_consumed - 100) * 0.75).
    - Else:
      - Set bill_amount = (100 * 0.50) + (100 * 0.75) + ((units_consumed - 200) * 1.00).
  - Else If customer_type is "commercial":
    - If units_consumed <= 100:
      - Set bill_amount = units_consumed * 1.00.
    - Else If units_consumed <= 200:
      - Set bill_amount = (100 * 1.00) + ((units_consumed - 100) * 1.25).
    - Else:
      - Set bill_amount = (100 * 1.00) + (100 * 1.25) + ((units_consumed - 200) * 1.50).
  - Return bill_amount.

4. Display Bill Information

- Print "Electricity Bill".

- Print Customer ID.
- Print Customer Name.
- Print Phone Number.
- Print Customer Type.
- Print Units Consumed.
- Print Bill Amount formatted to two decimal places.

5. End

Base code :

```
#include <stdio.h>
#include <string.h>
float calculate_bill(int units_consumed, char customer_type[])
{ float bill_amount = 0.0;
  if (strcmp(customer_type, "domestic") == 0)
    { if (units_consumed <= 100)
       { bill_amount = units_consumed * 0.50;
       }
       else if (units_consumed <= 200)
       {bill_amount = (100 * 0.50) + ((units_consumed - 100) * 0.75);
       }
       else
       {
          bill_amount = (100 * 0.50) + (100 * 0.75) + ((units_consumed - 200) * 1.00);
       } }
    else if (strcmp(customer_type, "commercial") == 0)
    {
       if (units_consumed <= 100)
       {
          bill_amount = units_consumed * 1.00;
```

```
        }
        else if (units_consumed <= 200)
        {     bill_amount = (100 * 1.00) + ((units_consumed - 100) * 1.25);
        }
        else
        {
           bill_amount = (100 * 1.00) + (100 * 1.25) + ((units_consumed - 200) * 1.50);
        }   }
return bill_amount;
}
int main()
{
    char customer_id[20], customer_name[50], phone_number[15], customer_type[20];
    int units_consumed;
    float bill_amount;
 printf("Enter Customer ID: ");
    scanf("%s", customer_id);
  printf("Enter Customer Name: ");
    scanf("%s", customer_name);
    printf("Enter Phone Number: ");
    scanf("%s", phone_number);
    printf("Enter Customer Type (domestic/commercial): ");
    scanf("%s", customer_type);
    printf("Enter number of units consumed: ");
    scanf("%d", &units_consumed);
    bill_amount = calculate_bill(units_consumed, customer_type);
    printf("\nElectricity Bill\n");
    printf("Customer ID     : %s\n", customer_id);
    printf("Customer Name   : %s\n", customer_name);
```

```
    printf("Phone Number    : %s\n", phone_number);
    printf("Customer Type   : %s\n", customer_type);
    printf("Units Consumed  : %d\n", units_consumed);
    printf("Bill Amount     : %.2f\n", bill_amount);
    return 0;
}
```

Real Life Applications:

Code:

```
#include <stdio.h>
#include <string.h>
#define MAX_CATEGORIES 5
typedef enum {
    TRANSPORTATION,
    ACCOMMODATION,
    FOOD,
    ACTIVITIES,
    MISCELLANEOUS
} Category;
void displayMenu() {
    printf("\nVacation Budget Planner\n");
    printf("1. Set Destination\n");
    printf("2. Set Budget\n");
    printf("3. Allocate Funds\n");
    printf("4. Display Budget\n");
    printf("5. Exit\n");
}
void setDestination(char* destination) {
    printf("Enter your vacation destination: ");
```

```
    getchar(); // to consume the newline character left by previous scanf
    fgets(destination, 100, stdin);
    destination[strcspn(destination, "\n")] = '\0'; // Remove trailing newline
}
void setBudget(float* totalBudget) {
    printf("Enter your total budget: $");
    while (scanf("%f", totalBudget) != 1) {
        printf("Invalid input. Please enter a valid number for budget: $");
        while (getchar() != '\n'); // clear invalid input from buffer
    }
}
void allocateFunds(float* categories, float totalBudget) {
    float percentage;
    for (int i = 0; i < MAX_CATEGORIES; i++) {
        printf("Enter percentage for ");
        switch (i) {
            case TRANSPORTATION:
                printf("Transportation: ");
                break;
            case ACCOMMODATION:
                printf("Accommodation: ");
                break;
            case FOOD:
                printf("Food: ");
                break;
            case ACTIVITIES:
                printf("Activities: ");
                break;
            case MISCELLANEOUS:
```

```
            printf("Miscellaneous: ");
            break;
        }
    while (scanf("%f", &percentage) != 1 || percentage < 0 || percentage > 100) {
            printf("Invalid input. Please enter a valid percentage (0-100): ");
            while (getchar() != '\n'); // clear invalid input from buffer
        }
       categories[i] = (percentage / 100) * totalBudget;
    }
}
void displayBudget(float* categories, float totalBudget, char* destination) {
    printf("\nBudget Breakdown for %s:\n", destination);
    printf("Transportation: $%.2f\n", categories[TRANSPORTATION]);
    printf("Accommodation: $%.2f\n", categories[ACCOMMODATION]);
    printf("Food: $%.2f\n", categories[FOOD]);
    printf("Activities: $%.2f\n", categories[ACTIVITIES]);
    printf("Miscellaneous: $%.2f\n", categories[MISCELLANEOUS]);
    printf("Total Budget: $%.2f\n", totalBudget);
}
int main() {
    float totalBudget = 0;
    float categories[MAX_CATEGORIES] = {0};
    char destination[100] = "Unknown";
    int choice;
 do {
        // Display current destination and budget before showing the menu
        printf("\nCurrent Destination: %s\n", destination);
        printf("Current Budget: $%.2f\n", totalBudget);
        displayMenu();
```

```
        printf("Enter your choice: ");
            if (scanf("%d", &choice) != 1) {
            printf("Invalid input. Please enter a valid number.\n");
            while (getchar() != '\n'); // clear the invalid input from buffer
            continue; // loop back to display the menu again
        }
            switch (choice) {
            case 1:
                setDestination(destination);
                break;
            case 2:
                setBudget(&totalBudget);
                break;
            case 3:
                allocateFunds(categories, totalBudget);
                break;
            case 4:
                displayBudget(categories, totalBudget, destination);
                break;
            case 5:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 5);
      return 0;
}
```

Test:

This code was tested on various compilers like VS Studio, and online compilers like Programiz.com, w3schools.com. The following output was attained for different input cases.
OUTPUT-

Current Destination: Unknown

Current Budget: $0.00

Vacation Budget Planner

1. Set Destination

2. Set Budget

3. Allocate Funds

4. Display Budget

5. Exit

Enter your choice:

Enter your choice: 1

Enter your vacation destination: Hawaii

Current Destination: Hawaii

Current Budget: $0.00

Vacation Budget Planner

1. Set Destination

2. Set Budget

3. Allocate Funds

4. Display Budget

5. Exit

Enter your choice:

Enter your choice: 2

Enter your total budget: $1000

Current Destination: Hawaii

Current Budget: $1000.00

Vacation Budget Planner

1. Set Destination

2. Set Budget

3. Allocate Funds

4. Display Budget

5. Exit

Enter your choice:

Enter your choice: 3

Enter percentage for Transportation: 20

Enter percentage for Accommodation: 40

Enter percentage for Food: 20

Enter percentage for Activities: 10

Enter percentage for Miscellaneous: 10

Current Destination: Hawaii

Current Budget: $1000.00

Vacation Budget Planner

1. Set Destination

2. Set Budget

3. Allocate Funds

4. Display Budget

5. Exit

Enter your choice:

Enter your choice: 4

Budget Breakdown for Hawaii:

Transportation: $200.00

Accommodation: $400.00

Food: $200.00

Activities: $100.00

Miscellaneous: $100.00

Total Budget: $1000.00

Enter your choice: 5

Exiting program...

Implement:

Validation Function:

validateFloatInput(float* value):

Reads input as a string and attempts to convert it to a float. It checks if the conversion was successful by ensuring the entire string was parsed.

validatePercentageInput(float percentage)

Ensures the percentage is between 0 and 100.

validateBudgetInput(float budget):

Checks that the budget is a positive number.

Input Handling:

Using fgets: For reading strings like the destination, fgets is safe and handles spaces properly.

Custom Validation Functions: Each type of input is validated using specific functions. This modular approach makes the main logic clearer and easier to maintain.

Error Feedback Loop: For both budget and percentage inputs, the program repeatedly prompts the user until valid input is provied, ensuring data integrity.

Buffer Clearing: After each read, the input buffer is cleared to remove any unwanted characters that might interfere with subsequent inputs.

User Interaction:

Clear Prompts: The program uses straightforward language in prompts, such as "Please enter your total vacation budget in dollars: $".

Consistent Terminology: Consistent use of terms like "budget" and "categories" helps avoid confusion.

Guidance and Examples: Input prompts include examples (e.g., "Enter percentage for Transportation (e.g., 25 for 25%):") to clarify expectations.

Helpful Error Messages: When input is invalid, the program provides specific feedback, e.g., "Invalid input. Please enter a valid percentage (0-100):".

Logical Flow: The program is organized logically, guiding users through setting their destination, budget, allocating funds, and displaying results.

Confirmation Messages: After setting the budget or allocating funds, users receive confirmation of their actions, reinforcing understanding.

Summary Output:

Set a Destination: Input their vacation destination.

Set a Total Budget: Enter their total budget for the trip.

Allocate Funds: Distribute the budget across five categories: Transportation, Accommodation, Food, Activities, and Miscellaneous, using percentages.

Display Budget Breakdown: View a summary of the budget allocation for each category along with the total budget.

Group Members:-

Sanskruti H. Gudhe :B24CE1043
Shivtej C. Wagare :B24CE1041
Sakshi A. Badakh :B24CE1045
Shubham S. Dorake :B24CE1042

Soham D. Badarkhe :B24CE1040