EXTENDED FORM OF CIBIL SCORE SYSTEM

RESEARCH :

In the earlier activity, we saw an improved version of CIBIL SCORE SYSTEM including a database for 5 customers and a program calculating CIBIL SCORE on its own .
While the earlier version of the “CIBIL Score Calculation System” focused on score computation and sorting, this extended research introduces a search-by-name functionality, enabling users to locate and display individual customer details instantly.

This mimics how real credit databases allow bank officers to access specific customer profiles by name or ID. Implementing this feature demonstrates how structured data storage and linear searching techniques can be used for effective data management in C programming.

OBJECTIVE :

- To integrate search functionality within the existing CIBIL score system.
- To enable fast retrieval of specific customer information using their name.
- To analyze the efficiency of linear search algorithms for small datasets.
- To simulate how banking systems access individual credit reports.
- To demonstrate structured programming techniques for data organization.

The system manages a list of customer records containing:

- Name
- Age

- Account Number
- Number of Loans
- Credit Utilisation
- Payment History
- Income
- Calculated CIBIL Score

The program performs three major operations:

1. Input & Score Calculation — Using conditional logic based on user-entered data.
2. Sorting by Score — To analyze and rank customers.
3. Search by Name — To find and display a particular customer's complete profile.

Sources :

- Written on my own
- Same as for activity 4
- https://www.researchgate.net/publication/356493603_Analysis_and_Prediction_of_CIBIL_Score_using_Machine_Learning
- https://www.scribd.com/document/508450015/CSS-Synopsis

ANALYSIS :

The program performs three major operations:

4. Input & Score Calculation — Using conditional logic based on user-entered data.

5. Sorting by Score — To analyze and rank customers.

6. Search by Name — To find and display a particular customer's complete profile

Interpretation of Results

- Customers with high payment history and low credit utilisation get higher scores, validating the scoring logic.

- Sorting allows clear visual comparison of creditworthiness among customers.

- Search-by-name adds a realistic, data-retrieval element, mimicking how banks or credit agencies locate specific clients in databases.

- The logical flow—from input to computation, sorting, and targeted search—represents the basic structure of an information management system.

ALGORITHM :

Step 1:

Start the program.

Step 2:

Declare the variables for Customer :

- Payment History
- Credit Utilisation

- Number of existing loans
- Monthly Income
- Name
- Age
- Account number

Step 3 :

Create a function for taking Input the following details from the user :

- Payment history (in %)
- Credit utilization (in %)
- Number of existing loans
- Monthly income
- Name
- Age
- Account Number

Step 4 :

Create a function for calculation Cibil score for a customer using following conditions :

1. Initialise score = 300

Step 5 :

Evaluate payment history :

- If payment history > 90, add 300 to score
- Else if payment history > 75, add 200 to score
- Else, add 100 to score

Step 6 :

Evaluate credit utilisation :

- If credit utilisation > 75, subtract 100 from score
- Else if credit utilisation > 50, subtract 50 from score

Step 7 :

Evaluate number of loans :

- If loans > 5, subtract 100 from score
- Else if loans >= 3, subtract 50 from score

Step 8 :

Evaluate income :

- If income > 50000, add 100 to score
- Else if income > 30000, add 50 to score

Step 9 :

Ensure the final score is within range

- If score > 900, set score = 900
- If score < 300, set score = 300

Step 10 :

Sort customers by ascending order of their CIBIL SCORE

Use a nested loop

Step 10 :

Display all the input taken from the user

Step 11 :

Display the calculated CIBIL SCORE for all the 10 customers

Step 12 :

Search for a customer by name.

Step 13 :

Make the CIBIL SCORE in the ascending order

Step 14 :

Stop

BUILD :

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

#define MAX_CUSTOMERS 5
#define MAX_NAME_LEN 50


struct Customer {
    char name[MAX_NAME_LEN];
    int age;
    long acc_no;
    int loans;
    int credit_utilisation;
```

```
    int payment_history;
    int income;
    int cibil_score;
};


int calculateCibil(struct Customer c) {
    int score = 300;



    if (c.payment_history > 90)
        score += 300;
    else if (c.payment_history > 75)
        score += 200;
    else
        score += 100;



    if (c.credit_utilisation > 75)
        score -= 100;
    else if (c.credit_utilisation > 50)
        score -= 50;
    else if (c.credit_utilisation > 25)
```

```
    score -= 25;
else
    score += 100;



if (c.loans > 5)
    score -= 100;
else if (c.loans >= 3)
    score -= 50;
else if (c.loans >= 1)
    score -= 25;
else
    score += 100;



if (c.income > 50000)
    score += 100;
else if (c.income > 30000)
    score += 50;
else
    score += 25;
```

```
    if (score < 300) score = 300;
    if (score > 900) score = 900;

    return score;
}

void inputCustomers(struct Customer customers[], int n) {
    for (int i = 0; i < n; i++) {

printf("\n==========================================================
==================");
        printf("\nEnter details for Customer %d", i + 1);

printf("\n==========================================================
==================");

        printf("\nName: ");
        scanf(" %[^\n]", customers[i].name);
        printf("Age: ");
        scanf("%d", &customers[i].age);
        printf("Account Number: ");
        scanf("%ld", &customers[i].acc_no);
        printf("Number of Existing Loans: ");
```

```
        scanf("%d", &customers[i].loans);
        printf("Credit Utilisation (%%): ");
        scanf("%d", &customers[i].credit_utilisation);
        printf("Payment History (%% of on-time payments): ");
        scanf("%d", &customers[i].payment_history);
        printf("Monthly Income (in Rs): ");
        scanf("%d", &customers[i].income);



        customers[i].cibil_score = calculateCibil(customers[i]);
    }
}

void sortByCibil(struct Customer customers[], int n) {
    struct Customer temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (customers[i].cibil_score > customers[j].cibil_score) {
                temp = customers[i];
                customers[i] = customers[j];
                customers[j] = temp;
            }
        }
```

```
    }
}


void displayCustomers(struct Customer customers[], int n) {
    printf("\n\n============================== CUSTOMER REPORT ==============================\n");
    printf("%-20s %-5s %-12s %-10s %-20s %-20s %-10s %-12s\n",
           "Name", "Age", "Acc No", "Loans", "Credit Util(%)",
           "Payment Hist(%)", "Income", "CIBIL Score");
    printf("--------------------------------------------------------------------------------\n");

    for (int i = 0; i < n; i++) {
        printf("%-20s %-5d %-12ld %-10d %-20d %-20d %-10d %-12d\n",
               customers[i].name, customers[i].age, customers[i].acc_no,
               customers[i].loans, customers[i].credit_utilisation,
               customers[i].payment_history, customers[i].income,
               customers[i].cibil_score);
    }
}

void searchByName(struct Customer customers[], int n, char search_name[]) {
    for (int i = 0; i < n; i++) {
```

```
        if (strcmp(customers[i].name, search_name) == 0) {
            printf("\n\n================ CUSTOMER DETAILS FOUND
================\n");
            printf("Name: %s\n", customers[i].name);
            printf("Age: %d\n", customers[i].age);
            printf("Account Number: %ld\n", customers[i].acc_no);
            printf("Loans: %d\n", customers[i].loans);
            printf("Credit Utilisation: %d%%\n", customers[i].credit_utilisation);
            printf("Payment History: %d%%\n", customers[i].payment_history);
            printf("Income: Rs %d\n", customers[i].income);
            printf("CIBIL Score: %d\n", customers[i].cibil_score);
            return;
        }
    }


    printf("\nNo such customer found '%s'\n",search_name);
}




int main() {
```

```
    struct Customer customers[MAX_CUSTOMERS];
    char search_name[MAX_NAME_LEN];


printf("\n==================================================
==================");
    printf("\n            CIBIL SCORE CALCULATION SYSTEM (10
CUSTOMERS)");

printf("\n==================================================
==================\n");


    inputCustomers(customers, MAX_CUSTOMERS);
    sortByCibil(customers, MAX_CUSTOMERS);
    displayCustomers(customers, MAX_CUSTOMERS);


    printf("\n\nEnter the name of the customer to search: ");
    scanf(" %[^\n]", search_name);


    searchByName(customers, MAX_CUSTOMERS, search_name);



    return 0;
}
```

TESTING :

```
============================ CUSTOMER REPORT ============================
Name                Age   Acc No      Loans     Credit Util(%)      Payment Hist(%)     Income    CIBIL Score
-------------------------------------------------------------------------------
SARVESH MANDE       18    2           3         56                  67                  234768    400
MOKSHAD PATIL       19    5           2         76                  89                  2354879   475
JAYESH KULKARNI     18    3           3         34                  78                  342786    525
SHIVAM PATHADE      19    4           4         21                  56                  234798    550
VED MULEY           18    1           2         34                  89                  136591    550


Enter the name of the customer to search: MOKSHAD PATIL


=============== CUSTOMER DETAILS FOUND ===============
Name: MOKSHAD PATIL
Age: 19
Account Number: 5
Loans: 2
Credit Utilisation: 76%
Payment History: 89%
Income: Rs 2354879
CIBIL Score: 475


=== Code Execution Successful ===
```

As we can see, code has run and we searched for the 5th customer and the program has given us successful data for customer 5 also including the calculated CIBIL SCORE for customer 5 .

IMPLEMENTATION :

https://github.com/vedmuley536/Ved-Muley