Planet Catalogue System

1)Problem Statement

A group of researchers has hired our team to develop a software system designed to catalogue and manage information about planets within the universe. The system will allow both administrators and researchers to interact with a database of planets, with specific functionalities aimed at managing planet information, handling researcher accounts, and providing search and retrieval features. Additionally, the system will provide statistical reports related to the planets, including the number of catalogued planets, the number of discovered planets, and the number of planets observed. The software must also offer a user-friendly interface for managing researcher accounts, retrieving detailed planet data, and ensuring ease of search by various planet attributes.

The software system should meet the following capabilities:

1. The software should be able to manage various attributes of planets including the name of the planet, the classification of the planet, the size of the planet, the distance of the planet from its star. Additional relevant information may include the planet's discovery date, orbit, atmospheric composition, etc.

2. Researcher Account Management: every researcher should have a unique account number for identification. Administrators should be able to add, edit, or remove researcher accounts. Researchers should be able to register for new accounts, edit personal details, and track their contributions.

3.Researchers should be able to search for planets using attributes like name, type, or other characteristics and view detailed information about each planet after performing a search.

4. The system should be able to generate the total number of planets catalogued, the number of planets that have been discovered, the number of planets observed by researchers.

5. The system should support two types of users:

·Admin: An administrative user with privileges to manage planet information, researcher accounts, and view statistics.

·Researchers: Users who can search for planets, view detailed planet information, and manage their personal accounts and contributions.

2)Analysis

·On an Input:

The system expects users to input valid data for both planet attributes and researcher account information. For planet data, the input could include attributes like name, type, size, distance from the star, and any other relevant information. For researcher accounts, inputs such as name, contact details, and a unique researcher ID are expected. When a researcher searches for a planet, the system will expect search criteria like planet name, type, or size.

The input validation will ensure that only valid data is entered. If the input is invalid, the system should prompt the user to re-enter the data or terminate with an error message. Since the system uses text-based input, we can implement checks to validate whether the input matches the expected format.

·On Outputs:

After receiving valid input, the system will output the relevant planet data or statistics in a clear and structured format. For planet information, the system will display attributes such as the name, type, size, and distance from the star. For searches, it will display matching results and any additional planet details.

For administrative tasks, the system will output statistics such as the total number of catalogued planets, the number of planets discovered, and the number of planets observed. Additionally, account management outputs will include success or failure messages. Each output will be accompanied by a clear explanatory message to ensure that the user understands what is being presented.

·Data Structure:

The system requires the use of a relational database to store the planet information and researcher accounts.

Planet Data: A table for planets will store attributes like name, type, size, and distance from the star. This table may also include additional columns for other planet-specific attributes such as discovery date, atmosphere, and orbit information.

Researcher Accounts: A separate table will store researcher details, such as the researcher ID, name, contact information, and their contributions. Each researcher will have a unique account number, and this table will be linked to the planet data.

·Algorithm:

The core functionality of the system revolves around operations like adding, modifying, deleting, and retrieving records from the database, along with generating statistical reports. These tasks can be performed using basic CRUD (Create, Read, Update, Delete) operations on the database.

When a researcher searches for planets, the algorithm will execute a query on the planet table based on the search criteria. Efficient querying and indexing will ensure that the system can handle large data sets of planets. Statistics Calculations: The system will calculate the required statistics using simple aggregate queries. For account management tasks straightforward SQL “INSERT”, “UPDATE”, and “DELETE” statements will be used.

3) Design

A. Variable Declarations

A.1 Declare a Planet structure to represent the details of a planet:

name (string): Stores the planet's name.

type (string): Represents the planet's type, such as "terrestrial" or "gas giant."

size (float): Stores the size of the planet in Earth masses.

distance (float): Represents the distance of the planet from its star in light years.

discovered (integer): Flags whether the planet has been discovered (1 for yes, 0 for no).

observed (integer): Flags whether the planet has been observed (1 for yes, 0 for no).

A.2 Declare a Researcher structure for storing researcher account details:

accountNumber (integer): A unique identifier for the researcher.

name (string): Stores the researcher's name.

planetsCataloged (integer): Tracks how many planets the researcher has cataloged.

planetsDiscovered (integer): Tracks how many planets the researcher has discovered.

A.3 Declare global variables:

planetCatalog[MAX_PLANETS]: An array to store information for up to 100 planets.

researcherAccounts[MAX_RESEARCHERS]: An array to store up to 50 researcher accounts.

planetCount (integer): Tracks the current number of planets in the catalog.

researcherCount (integer): Tracks the current number of researchers in the system.

B. Functional Breakdown

B.1 Adding a Planet

1. Print a prompt for the user to enter planet details (name, type, size, distance, discovered, observed).
2. Validate user inputs for numeric values (size, distance, discovered, and observed).

3. Check if the catalog is full (planetCount >= MAX_PLANETS). If full, display an error message and exit the function.
4. Store the entered data in a Planet structure and add it to the planetCatalog array.
5. Increment the planetCount and display a success message.

B.2 Searching for a Planet

1. Prompt the user for the name of the planet they wish to search for.
2. Iterate through planetCatalog and compare the input name with each stored planet's name using strcmp.
3. If a match is found, display the planet's details (name, type, size, distance, discovery status, and observation status).
4. If no match is found, display a message stating that the planet is not in the catalog.

B.3 Displaying Planet Information

1. Check if there are any planets in the catalog (planetCount == 0). If not, display a message and exit the function.
2. Iterate through planetCatalog and print the details of each planet in a readable format with proper labels.

B.4 Viewing Statistics

1. Initialize counters for discovered and observed planets.
2. Iterate through planetCatalog:
    1. Increment discoveredCount for planets marked as discovered.

    2. Increment observedCount for planets marked as observed.

3. Print the total number of planets, discovered planets, and observed planets.

B.5 Adding a Researcher

1. Print a prompt to collect researcher details (name and unique account number).
2. Validate the account number and ensure it is unique by checking existing researcherAccounts.
3. Check if the researcher database is full (researcherCount >= MAX_RESEARCHERS). If full, display an error message and exit the function.
4. Store the entered data in a Researcher structure and add it to the researcherAccounts array.
5. Increment the researcherCount and display a success message.

B.6 Displaying Researcher Information

1. Check if there are any researcher accounts (researcherCount == 0). If not, display a message and exit the function.
2. Iterate through researcherAccounts and print the details of each researcher, including their name, account number, and counts of cataloged and discovered planets.

B.7 Exiting the program directly

4) Implementation

see the C code "sample.c" (at the end of this document) with comments.

5) Testing

Test 1

```
Enter your choice: 1

Enter planet name: Mars

Enter planet type: Terrestrial

Enter planet size (in Earth masses): 0.107

Enter distance from star (in light years): 225.0

Is the planet discovered? (1 for yes, 0 for no): 1

Has the planet been observed? (1 for yes, 0 for no): 1

Planet added successfully!
```

Test 2

```
Enter your choice: 1

Enter planet name: Venus

Enter planet type: Terrestrial

Enter planet size (in Earth masses): abc

Invalid size input.
```

Test 3

Test 3

```
Enter your choice: 1
Enter planet name: Jupiter
Enter planet type: Gas giant
Enter planet size (in Earth masses): 318.0
Enter distance from star (in light years): xyz
Invalid distance input.
```

Test 4

```
Enter your choice: 1
Enter planet name: Jupiter
Enter planet type: Gas giant
Enter planet size (in Earth masses): 95.0
Enter distance from star (in light years): 1.2
Is the planet discovered? (1 for yes, 0 for no): xyz
Invalid input for discovered status.
```

Test 5

```
Enter your choice: 1
Enter planet name: Neptune
Enter planet type: Ice giant
Enter planet size (in Earth masses): 17.2
Enter distance from star (in light years): 4.5
```

```
Is the planet discovered? (1 for yes, 0 for no): 1

Has the planet been observed? (1 for yes, 0 for no): abc

Invalid input for observed status.
```

Test 6

```
Enter your choice: 2

Enter planet name to search: Mars

Planet found: Mars, Type: Terrestrial, Size: 0.11, Distance: 225.00 light years
```

Test 7

```
Enter your choice: 2

Enter planet name to search: Pluto

Planet not found in the catalog.
```

Test 8 （If there are no planets in catalog)

```
Enter your choice: 3

No planets in the catalog yet.
```

Test 9 （There are planets in catalog)

```
Enter your choice: 3

Planet Name: Mars

Type: Terrestrial
```

Size: 0.11 Earth masses

Distance from Star: 225.00 light years

Discovered: Yes

Observed: Yes

Test 10 (If there are no planets in catalog)

Enter your choice: 4

Total planets cataloged: 0

Total planets discovered: 0

Total planets observed: 0

Test 11 (There are planets in catalog)

Enter your choice: 4

Total planets cataloged: 1

Total planets discovered: 1

Total planets observed: 1

Test 12

Enter your choice: 5

Enter researcher name: John

Enter unique researcher account number: 101

Researcher added successfully!

Test 13

```
Enter your choice: 5

Enter researcher name: John
Enter unique researcher account number: abc

Invalid account number.
```

Test 14　(No researchers)

```
Enter your choice: 6

No researcher accounts available.
```

Test 15　(Researchers Exist)

```
Enter your choice: 6

Researcher Name: John

Account Number: 101

Planets Cataloged: 0

Planets Discovered: 0
```

Test 16

```
Enter your choice: 7

Exiting the program.
```

C code

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

// Constants for maximum limits
#define MAX_PLANETS 100
#define MAX_RESEARCHERS 50
#define MAX_NAME_LEN 50
#define MAX_TYPE_LEN 20

// Structure to hold planet information
typedef struct {
    char name[MAX_NAME_LEN];       // Name of the planet
    char type[MAX_TYPE_LEN];       // Type of the planet (e.g., gas giant, terrestrial)
    float size;                    // Size of the planet (in Earth masses)
    float distance;                // Distance from its star (in light years)
    int discovered;                // 1 if discovered, 0 if not
    int observed;                  // 1 if observed, 0 if not
} Planet;

// Structure to hold researcher account information
typedef struct {
    int accountNumber;             // Unique account number for each researcher
    char name[MAX_NAME_LEN];       // Name of the researcher
    int planetsCataloged;          // Number of planets cataloged by the researcher
    int planetsDiscovered;         // Number of planets discovered by the researcher
} Researcher;

// Global variables
Planet planetCatalog[MAX_PLANETS];
Researcher researcherAccounts[MAX_RESEARCHERS];
int planetCount = 0;              // Tracks the number of planets in the catalog
int researcherCount = 0;          // Tracks the number of researchers

// Function declarations
void addPlanet();
void displayPlanetInfo();
void searchPlanet();
void viewStatistics();
int addResearcher();
void displayResearcherInfo();

// Main function - Entry point
int main() {
    int choice;

    // Main menu loop
    while (1) {
        printf("\n--- Planet Catalog System ---\n");
        printf("1. Add a Planet\n");
        printf("2. Search for a Planet\n");
        printf("3. Display Planet Information\n");
        printf("4. View Statistics\n");
        printf("5. Add Researcher Account\n");
        printf("6. Display Researcher Information\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");

        // Check if the input was successful
        if (scanf("%d", &choice) != 1) {
            // Handle invalid input
            printf("Invalid input. Please enter a valid option.\n");
            while (getchar() != '\n'); // Clear the input buffer
            continue;
        }

        switch (choice) {
        case 1:
            addPlanet();
            break;
        case 2:
            searchPlanet();
            break;
        case 3:
            displayPlanetInfo();
            break;
        case 4:
            viewStatistics();
            break;
        case 5:
            addResearcher();
            break;
        case 6:
            displayResearcherInfo();
            break;
        case 7:
            printf("Exiting the program.\n");
            return 0;
        default:
            printf("Invalid choice, please try again.\n");
        }
    }
}

// Function to add a planet
void addPlanet() {
    if (planetCount >= MAX_PLANETS) {
        printf("Planet catalog is full. Cannot add more planets.\n");
        return;
    }

    Planet newPlanet;

    printf("Enter planet name:");
    getchar();
    fgets(newPlanet.name, MAX_NAME_LEN, stdin);
    newPlanet.name[strcspn(newPlanet.name, "\n")] = 0;
    printf("Enter planet type:");
    fgets(newPlanet.type, MAX_TYPE_LEN, stdin);
    newPlanet.type[strcspn(newPlanet.type, "\n")] = 0;
    printf("Enter planet size (in Earth masses): ");
    if (scanf("%f", &newPlanet.size) != 1) {
        printf("Invalid size input.\n");
        while (getchar() != '\n');
        return;
    }
    printf("Enter distance from star (in light years): ");
    if (scanf("%f", &newPlanet.distance) != 1) {
        printf("Invalid distance input.\n");
        while (getchar() != '\n');
        return;
    }
    printf("Is the planet discovered? (1 for yes, 0 for no): ");
    if (scanf("%d", &newPlanet.discovered) != 1) {
        printf("Invalid input for discovered status.\n");
        while (getchar() != '\n');
        return;
    }
    printf("Has the planet been observed? (1 for yes, 0 for no): ");
    if (scanf("%d", &newPlanet.observed) != 1) {
        printf("Invalid input for observed status.\n");
        while (getchar() != '\n');
        return;
    }

    // Add the new planet to the catalog
    planetCatalog[planetCount++] = newPlanet;
    printf("Planet added successfully!\n");
}

// Function to search for a planet by name
void searchPlanet() {
    char searchName[MAX_NAME_LEN];
    printf("Enter planet name to search: ");
    getchar();
    fgets(searchName, MAX_NAME_LEN, stdin);
    searchName[strcspn(searchName, "\n")] = '\0';

    int found = 0;
    for (int i = 0; i < planetCount; i++) {
        if (strcmp(planetCatalog[i].name, searchName) == 0) {
            printf("Planet found: %s, Type: %s, Size: %.2f, Distance: %.2f light
years\n",
                planetCatalog[i].name, planetCatalog[i].type,
                planetCatalog[i].size, planetCatalog[i].distance);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Planet not found in the catalog.\n");
    }
}

// Function to display all planet information
void displayPlanetInfo() {
    if (planetCount == 0) {
        printf("No planets in the catalog yet.\n");
        return;
    }
```

```c
    for (int i = 0; i < planetCount; i++) {
        printf("\nPlanet Name: %s\n", planetCatalog[i].name);
        printf("Type: %s\n", planetCatalog[i].type);
        printf("Size: %.2f Earth masses\n", planetCatalog[i].size);
        printf("Distance from Star: %.2f light years\n", planetCatalog[i].distance);
        printf("Discovered: %s\n", planetCatalog[i].discovered ? "Yes" : "No");
        printf("Observed: %s\n", planetCatalog[i].observed ? "Yes" : "No");
    }
}

// Function to view catalog statistics
void viewStatistics() {
    int discoveredCount = 0;
    int observedCount = 0;

    for (int i = 0; i < planetCount; i++) {
        if (planetCatalog[i].discovered) discoveredCount++;
        if (planetCatalog[i].observed) observedCount++;
    }

    printf("Total planets cataloged: %d\n", planetCount);
    printf("Total planets discovered: %d\n", discoveredCount);
    printf("Total planets observed: %d\n", observedCount);
}

// Function to add a researcher
int addResearcher() {
    if (researcherCount >= MAX_RESEARCHERS) {
        printf("Researcher accounts are full.\n");
        return -1;
    }

    Researcher newResearcher;
    printf("Enter researcher name: ");
    getchar();
    fgets(newResearcher.name, MAX_NAME_LEN, stdin);
    newResearcher.name[strcspn(newResearcher.name, "\n")] = 0;
    printf("Enter unique researcher account number: ");
    if (scanf("%d", &newResearcher.accountNumber) != 1) {
        printf("Invalid account number.\n");
        while (getchar() != '\n');
        return -1;
    }

    newResearcher.planetsCataloged = 0;
    newResearcher.planetsDiscovered = 0;

    researcherAccounts[researcherCount++] = newResearcher;
    printf("Researcher added successfully!\n");
    return 0;
}

// Function to display researcher information
void displayResearcherInfo() {
    if (researcherCount == 0) {
        printf("No researcher accounts available.\n");
        return;
    }

    for (int i = 0; i < researcherCount; i++) {
        printf("\nResearcher Name: %s\n", researcherAccounts[i].name);
        printf("Account Number: %d\n", researcherAccounts[i].accountNumber);
        printf("Planets Cataloged: %d\n", researcherAccounts[i].planetsCataloged);
        printf("Planets Discovered: %d\n", researcherAccounts[i].planetsDiscovered);
    }
}
```

Name	ID Number	Contribution(%)	Signature
1.侯佳汭	1931378	30	侯佳汭
2.马熙翔	2255229	20	马熙翔
3.冒埼宇	2361492	10	冒埼宇
4.宋天顺	2255795	10	宋天顺
5.王先正	2360086	30	王先正
Total		100	