Problem 1: Vehicle Fleet Management System

Requirements:

- Create a structure Vehicle with the following members:
    - char registrationNumber[15]
    - char model[30]
    - int yearOfManufacture
    - float mileage
    - float fuelEfficiency
- Implement functions to:
- Add a new vehicle to the fleet.
- Update the mileage and fuel efficiency for a vehicle.
- Display all vehicles manufactured after a certain year.
- Find the vehicle with the highest fuel efficiency.
- Use dynamic memory allocation to manage the fleet of vehicles.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Vehicle {
   char regNumber[15];
   char model[30];
   int year;
   float mileage;
   float fuelEfficiency;
};
int main() {
   int fleetSize = 5, count = 0;
   struct Vehicle *fleet = malloc(fleetSize * sizeof(struct Vehicle));
   void addVehicle() {
      printf("Enter Registration: ");
      scanf("%s", fleet[count].regNumber);
      printf("Enter Model: ");
```

```
    scanf("%s", fleet[count].model);
    printf("Enter Year: ");
    scanf("%d", &fleet[count].year);
    printf("Enter Mileage: ");
    scanf("%f", &fleet[count].mileage);
    printf("Enter Fuel Efficiency: ");
    scanf("%f", &fleet[count].fuelEfficiency);
    count++;
  }
  void displayVehiclesAfterYear(int year) {
    for (int i = 0; i < count; i++) {
      if (fleet[i].year > year) {
        printf("regNumbe:%s\n model:%s\n year:%d\n float mileage:%.2f\n float fuelEfficiency:
%.2f\n", fleet[i].regNumber, fleet[i].model, fleet[i].year, fleet[i].mileage, fleet[i].fuelEfficiency);
      }
    }
  }
  void findHighestFuelEfficiency() {
    if (count == 0)
    return;
    int maxIndex = 0;
    for (int i = 1; i < count; i++) {
      if (fleet[i].fuelEfficiency > fleet[maxIndex].fuelEfficiency) {
        maxIndex = i;
      }
    }
    printf("Highest Fuel Efficiency: %s %s %.2f\n", fleet[maxIndex].regNumber,
fleet[maxIndex].model, fleet[maxIndex].fuelEfficiency);
  }
  int choice;
  do {
```

```
        printf("\n1. Add Vehicle\n2. Display Vehicles After Year\n3. Find Highest Fuel Efficiency\n4. Exit\n");
        printf("Enter your choice:");
        scanf("%d", &choice);
        if (choice == 1) addVehicle();
        else if (choice == 2) {
            int year;
            printf("Enter year: ");
            scanf("%d", &year);
            displayVehiclesAfterYear(year);
        }
        else if (choice == 3)
        findHighestFuelEfficiency();
    } while (choice != 4);

    free(fleet);
    return 0;
}
```

o/p:

```
1. Add Vehicle
2. Display Vehicles After Year
3. Find Highest Fuel Efficiency
4. Exit
Enter your choice:1
Enter Registration: acvf
Enter Model: bmw
Enter Year: 2012
Enter Mileage: 2345.78
Enter Fuel Efficiency: 34.6

1. Add Vehicle
```

2. Display Vehicles After Year

3. Find Highest Fuel Efficiency

4. Exit

Enter your choice:1

Enter Registration: yhtt

Enter Model: aadi

Enter Year: 2015

Enter Mileage: 1234.6

Enter Fuel Efficiency: 45.6

1. Add Vehicle

2. Display Vehicles After Year

3. Find Highest Fuel Efficiency

4. Exit

Enter your choice:2

Enter year: 2012

regNumbe:yhtt

model:aadi

year:2015

float mileage:1234.60

float fuelEfficiency: 45.60

1. Add Vehicle

2. Display Vehicles After Year

3. Find Highest Fuel Efficiency

4. Exit

Enter your choice:3

Highest Fuel Efficiency: yhtt aadi 45.60

1. Add Vehicle

2. Display Vehicles After Year

3. Find Highest Fuel Efficiency

4. Exit

Enter your choice:4

Problem 2: Car Rental Reservation System

Requirements:

- Define a structure CarRental with members:
  - char carID[10]
  - char customerName[50]
  - char rentalDate[11] (format: YYYY-MM-DD)
  - char returnDate[11]
  - float rentalPricePerDay
- Write functions to:
- Book a car for a customer by inputting necessary details.
- Calculate the total rental price based on the number of rental days.
- Display all current rentals.
- Search for rentals by customer name.
- Implement error handling for invalid dates and calculate the number of rental days.

```
#include <stdio.h>
#include <string.h>
struct CarRental {
   char carID[10];
   char customerName[50];
   char rentalDate[11];
   char returnDate[11];
   float rentalPricePerDay;
   float totalPrice;
};
int calculateDays(char *rentalDate, char *returnDate) {
   int rentalYear, rentalMonth, rentalDay;
   int returnYear, returnMonth, returnDay;
```

```
    sscanf(rentalDate, "%d-%d-%d", &rentalYear, &rentalMonth, &rentalDay);
    sscanf(returnDate, "%d-%d-%d", &returnYear, &returnMonth, &returnDay);
    return (returnYear - rentalYear) * 365 + (returnMonth - rentalMonth) * 30 + (returnDay -
rentalDay);
}
int main() {
    struct CarRental rental;
    int choice;
    do {
        printf("\n1. Book Car\n2. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if (choice == 1) {
            printf("Enter Car ID: ");
            scanf("%s", rental.carID);
            printf("Enter Customer Name: ");
            scanf("%s",rental.customerName);
            printf("Enter Rental Date (YYYY-MM-DD): ");
            scanf("%s", rental.rentalDate);
            printf("Enter Return Date (YYYY-MM-DD): ");
            scanf("%s", rental.returnDate);
            printf("Enter Price per Day: ");
            scanf("%f", &rental.rentalPricePerDay);

            int days = calculateDays(rental.rentalDate, rental.returnDate);
            rental.totalPrice = days * rental.rentalPricePerDay;
            printf("Total Price: %.2f\n", rental.totalPrice);
        }
    } while (choice != 2);

    return 0;
```

}

o/p:

1. Book Car

2. Exit

Enter your choice: 1

Enter Car ID: ss23

Enter Customer Name: sofi

Enter Rental Date (YYYY-MM-DD): 2012-03-21

Enter Return Date (YYYY-MM-DD): 2012-05-12

Enter Price per Day: 200

Total Price: 10200.00

1. Book Car

2. Exit

Enter your choice: 2

Problem 3: Autonomous Vehicle Sensor Data Logger

Requirements:

- Create a structure SensorData with fields:
    - int sensorID
    - char timestamp[20] (format: YYYY-MM-DD HH:MM:SS)
    - float speed
    - float latitude
    - float longitude
- Functions to:
- Log new sensor data.
- Display sensor data for a specific time range.
- Find the maximum speed recorded.
- Calculate the average speed over a specific time period.
- Store sensor data in a dynamically allocated array and resize it as needed.

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

```
struct SensorData {
    int sensorID;
    char timestamp[20];
    float speed;
    float latitude;
    float longitude;
};

void logSensorData(struct SensorData *data, int *count) {
    printf("Enter Sensor ID: ");
    scanf("%d", &data[*count].sensorID);

    getchar();

    printf("Enter Timestamp (YYYY-MM-DD HH:MM:SS): ");
    scanf("%19[^\n]", data[*count].timestamp);  // Read until newline, avoid overflow

    printf("Enter Speed: ");
    scanf("%f", &data[*count].speed);
    printf("Enter Latitude: ");
    scanf("%f", &data[*count].latitude);
    printf("Enter Longitude: ");
    scanf("%f", &data[*count].longitude);

    (*count)++;
}

void displayData(struct SensorData *data, int count) {
    if (count == 0) {
        printf("No data available.\n");
        return;
```

```
    }
    for (int i = 0; i < count; i++) {
        printf("Sensor ID: %d, Timestamp: %s, Speed: %.2f, Latitude: %.2f, Longitude: %.2f\n",
               data[i].sensorID, data[i].timestamp, data[i].speed, data[i].latitude, data[i].longitude);
    }
}

void findMaxSpeed(struct SensorData *data, int count) {
    if (count == 0) {
        printf("No data available.\n");
        return;
    }

    float maxSpeed = data[0].speed;
    for (int i = 1; i < count; i++) {
        if (data[i].speed > maxSpeed) {
            maxSpeed = data[i].speed;
        }
    }
    printf("Maximum Speed: %.2f\n", maxSpeed);
}

void calculateAverageSpeed(struct SensorData *data, int count) {
    if (count == 0) {
        printf("No data available.\n");
        return;
    }
    float totalSpeed = 0;
    for (int i = 0; i < count; i++) {
        totalSpeed += data[i].speed;
    }
```

```
    printf("Average Speed: %.2f\n", totalSpeed / count);
}
int main() {
    struct SensorData data[100];
    int count = 0;
    int choice;
    do {
        printf("\nMenu:\n");
        printf("1. Log Sensor Data\n");
        printf("2. Display All Data\n");
        printf("3. Find Maximum Speed\n");
        printf("4. Calculate Average Speed\n");
        printf("5. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            logSensorData(data, &count);
        }
        else if (choice == 2) {
            displayData(data, count);
        }
        else if (choice == 3) {
            findMaxSpeed(data, count);
        }
        else if (choice == 4) {
            calculateAverageSpeed(data, count);
        }
        else if (choice == 5) {
            printf("Exiting\n");
```

```
        }
        else {
            printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 5);
    return 0;
}
```

o/p:

```
Menu:
1. Log Sensor Data
2. Display All Data
3. Find Maximum Speed
4. Calculate Average Speed
5. Exit
Enter your choice: 1
Enter Sensor ID: 21233
Enter Timestamp (YYYY-MM-DD HH:MM:SS): 2012-03-23 11:23:34
Enter Speed: 34.04
Enter Latitude: 123.00
Enter Longitude: 231.00
Menu:
1. Log Sensor Data
2. Display All Data
3. Find Maximum Speed
4. Calculate Average Speed
5. Exit
Enter your choice: 1
Enter Sensor ID: 8765
Enter Timestamp (YYYY-MM-DD HH:MM:SS):2014-05-23 12:45:21
Enter Speed: 45.0
Enter Latitude: 433.44
```

Enter Longitude: 341.00

Menu:

1. Log Sensor Data

2. Display All Data

3. Find Maximum Speed

4. Calculate Average Speed

5. Exit

Enter your choice: 3

Maximum Speed: 45.00

Menu:

1. Log Sensor Data

2. Display All Data

3. Find Maximum Speed

4. Calculate Average Speed

5. Exit

Enter your choice: 4

Average Speed: 39.52

Menu:

1. Log Sensor Data

2. Display All Data

3. Find Maximum Speed

4. Calculate Average Speed

5. Exit

Enter your choice: 5

Exiting

Problem 4: Engine Performance Monitoring System

Requirements:

- Define a structure EnginePerformance with members:
    - char engineID[10]
    - float temperature

- float rpm
- float fuelConsumptionRate
- float oilPressure

- Functions to:
- Add performance data for a specific engine.
- Display all performance data for a specific engine ID.
- Calculate the average temperature and RPM for a specific engine.
- Identify any engine with abnormal oil pressure (above or below specified thresholds).
- Use linked lists to store and manage performance data entries.

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

struct EnginePerformance {
    char engineID[10];
    float temperature;
    float rpm;
    float fuelConsumptionRate;
    float oilPressure;
};

void addPerformanceData(struct EnginePerformance *engines, int *count) {
    printf("Enter Engine ID, Temperature, RPM, Fuel Consumption Rate, Oil Pressure: ");
    scanf("%s %f %f %f %f", engines[*count].engineID, &engines[*count].temperature,
&engines[*count].rpm,
        &engines[*count].fuelConsumptionRate, &engines[*count].oilPressure);
    (*count)++;
}

void displayPerformanceData(struct EnginePerformance *engines, int count, char *engineID) {
    for (int i = 0; i < count; i++) {
```

```
        if (strcmp(engines[i].engineID, engineID) == 0) {
            printf("Engine %s: Temperature %.2f, RPM %.2f, Fuel Consumption %.2f, Oil Pressure %.2f\n",
                   engines[i].engineID, engines[i].temperature, engines[i].rpm,
engines[i].fuelConsumptionRate, engines[i].oilPressure);
        }
    }
}

void calculateAvgTemperatureRPM(struct EnginePerformance *engines, int count, char *engineID) {
    float totalTemp = 0, totalRPM = 0;
    int matchCount = 0;
    for (int i = 0; i < count; i++) {
        if (strcmp(engines[i].engineID, engineID) == 0) {
            totalTemp += engines[i].temperature;
            totalRPM += engines[i].rpm;
            matchCount++;
        }
    }
    if (matchCount > 0) {
        printf("Average Temperature: %.2f, Average RPM: %.2f\n", totalTemp / matchCount, totalRPM /
matchCount);
    } else {
        printf("No data found for engine %s.\n", engineID);
    }
}

void checkAbnormalOilPressure(struct EnginePerformance *engines, int count, float lowThreshold,
float highThreshold) {
    for (int i = 0; i < count; i++) {
        if (engines[i].oilPressure < lowThreshold || engines[i].oilPressure > highThreshold) {
            printf("Engine %s has abnormal oil pressure: %.2f\n", engines[i].engineID,
engines[i].oilPressure);
```

```
        }
    }
}

int main() {
    struct EnginePerformance engines[100];
    int count = 0, choice;
    char engineID[10];
    float lowThreshold = 10.0, highThreshold = 90.0; // Example thresholds

    while (1) {
        printf("\n1. Add Performance Data\n2. Display Performance Data\n3. Calculate Avg Temperature & RPM\n4. Check Abnormal Oil Pressure\n5. Exit\nChoice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            addPerformanceData(engines, &count);
        } else if (choice == 2) {
            printf("Enter Engine ID: ");
            scanf("%s", engineID);
            displayPerformanceData(engines, count, engineID);
        } else if (choice == 3) {
            printf("Enter Engine ID: ");
            scanf("%s", engineID);
            calculateAvgTemperatureRPM(engines, count, engineID);
        } else if (choice == 4) {
            checkAbnormalOilPressure(engines, count, lowThreshold, highThreshold);
        } else {
            break;
        }
    }
```

```
    return 0;
}
```

o/p:

```
1. Add Performance Data
2. Display Performance Data
3. Calculate Avg Temperature & RPM
4. Check Abnormal Oil Pressure
5. Exit
Choice: 1
Enter Engine ID, Temperature, RPM, Fuel Consumption Rate, Oil Pressure: E001 85.0 2200 12.5 75.0

1. Add Performance Data
2. Display Performance Data
3. Calculate Avg Temperature & RPM
4. Check Abnormal Oil Pressure
5. Exit
Choice: 2
Enter Engine ID: E001
Engine E001: Temperature 85.00, RPM 2200.00, Fuel Consumption 12.50, Oil Pressure 75.00

1. Add Performance Data
2. Display Performance Data
3. Calculate Avg Temperature & RPM
4. Check Abnormal Oil Pressure
5. Exit
Choice: 3
Enter Engine ID: E001
Average Temperature: 85.00, Average RPM: 2200.00

1. Add Performance Data
```

2. Display Performance Data

3. Calculate Avg Temperature & RPM

4. Check Abnormal Oil Pressure

5. Exit

Choice: 4

Engine E001 has abnormal oil pressure: 75.00

Problem 5: Vehicle Service History Tracker

Requirements:

- Create a structure ServiceRecord with the following:
    - char serviceID[10]
    - char vehicleID[15]
    - char serviceDate[11]
    - char description[100]
    - float serviceCost
- Functions to:
- Add a new service record for a vehicle.
- Display all service records for a given vehicle ID.
- Calculate the total cost of services for a vehicle.
- Sort and display service records by service date.

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

struct ServiceRecord {
   char serviceID[10];
   char vehicleID[15];
   char serviceDate[11];
   char description[100];
   float serviceCost;
};
```

```
void addServiceRecord(struct ServiceRecord records[], int *count) {
    printf("Enter service ID: ");
    scanf("%s", records[*count].serviceID);
    printf("Enter vehicle ID: ");
    scanf("%s", records[*count].vehicleID);
    printf("Enter service date (DD/MM/YYYY): ");
    scanf("%s", records[*count].serviceDate);
    printf("Enter service description: ");
    getchar();
    fgets(records[*count].description, 100, stdin);
    printf("Enter service cost: ");
    scanf("%f", &records[*count].serviceCost);
    (*count)++;
}

void displayServiceRecords(struct ServiceRecord records[], int count, char vehicleID[]) {
    printf("Service records for vehicle %s:\n", vehicleID);
    for (int i = 0; i < count; i++) {
        if (strcmp(records[i].vehicleID, vehicleID) == 0) {
            printf("Service ID: %s, Date: %s, Description: %s, Cost: %.2f\n",
                records[i].serviceID, records[i].serviceDate, records[i].description, records[i].serviceCost);
        }
    }
}

float totalServiceCost(struct ServiceRecord records[], int count, char vehicleID[]) {
    float total = 0;
    for (int i = 0; i < count; i++) {
        if (strcmp(records[i].vehicleID, vehicleID) == 0) {
            total += records[i].serviceCost;
        }
```

```
    }
    return total;
}

int compareDates(const void *a, const void *b) {
    return strcmp(((struct ServiceRecord*)a)->serviceDate, ((struct ServiceRecord*)b)->serviceDate);
}

void sortAndDisplayRecords(struct ServiceRecord records[], int count) {
    qsort(records, count, sizeof(struct ServiceRecord), compareDates);
    for (int i = 0; i < count; i++) {
        printf("Service ID: %s, Vehicle ID: %s, Date: %s, Cost: %.2f\n",
            records[i].serviceID, records[i].vehicleID, records[i].serviceDate, records[i].serviceCost);
    }
}

int main() {
    struct ServiceRecord records[100];
    int count = 0;
    int choice;
    char vehicleID[15];

    while (1) {
        printf("\n1. Add Service Record\n");
        printf("2. Display Service Records\n");
        printf("3. Total Service Cost\n");
        printf("4. Sort and Display Records\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        if (choice == 1) {
            addServiceRecord(records, &count);
        }
        else if (choice == 2) {
            printf("Enter vehicle ID to display: ");
            scanf("%s", vehicleID);
            displayServiceRecords(records, count, vehicleID);
        }
        else if (choice == 3) {
            printf("Enter vehicle ID to calculate total cost: ");
            scanf("%s", vehicleID);
            printf("Total service cost: %.2f\n", totalServiceCost(records, count, vehicleID));
        }
        else if (choice == 4) {
            sortAndDisplayRecords(records, count);
        }
        else if (choice == 5) {
            break;
        } else {
            printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

o/p:

1. Add Service Record
2. Display Service Records
3. Total Service Cost
4. Sort and Display Records
5. Exit

Enter your choice: 1

Enter service ID: 345

Enter vehicle ID: TC234

Enter service date (DD/MM/YYYY): 21/08/2023

Enter service description: Oil exchange

Enter service cost: 230

1. Add Service Record

2. Display Service Records

3. Total Service Cost

4. Sort and Display Records

5. Exit

Enter your choice: 2

Enter vehicle ID to display: TC234

Service records for vehicle TC234:

Service ID: 345, Date: 21/08/2023, Description: Oil exchange

, Cost: 230.00

1. Add Service Record

2. Display Service Records

3. Total Service Cost

4. Sort and Display Records

5. Exit

Enter your choice: 3

Enter vehicle ID to calculate total cost: TC234

Total service cost: 230.00

1. Add Service Record

2. Display Service Records

3. Total Service Cost

4. Sort and Display Records

5. Exit

Enter your choice: 4

Service ID: 345, Vehicle ID: TC234, Date: 21/08/2023, Cost: 230.00

1. Add Service Record

2. Display Service Records

3. Total Service Cost

4. Sort and Display Records

5. Exit

Enter your choice: 5

Problem 1: Player Statistics Management

Requirements:

- Define a structure Player with the following members:
    - char name[50]
    - int age
    - char team[30]
    - int matchesPlayed
    - int totalRuns
    - int totalWickets
- Functions to:
- Add a new player to the system.
- Update a player's statistics after a match.
- Display the details of players from a specific team.
- Find the player with the highest runs and the player with the most wickets.
- Use dynamic memory allocation to store player data in an array and expand it as needed.

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

```
struct Player {
    char name[50];
    int age;
    char team[30];
    int matchesPlayed;
    int totalRuns;
    int totalWickets;
};

void addPlayer(struct Player **players, int *count) {
    *players = realloc(*players, (*count + 1) * sizeof(struct Player));

    printf("Enter player's name: ");
    scanf("%s",&(*players)[*count].name);
    printf("Enter player's age: ");
    scanf("%d", &(*players)[*count].age);

    printf("Enter player's team: ");
    scanf("%s",&(*players)[*count].team);

    printf("Enter number of matches played: ");
    scanf("%d", &(*players)[*count].matchesPlayed);

    printf("Enter total runs: ");
    scanf("%d", &(*players)[*count].totalRuns);

    printf("Enter total wickets: ");
    scanf("%d", &(*players)[*count].totalWickets);

    (*count)++;
```

```
}

void displayPlayers(struct Player *players, int count) {
    for (int i = 0; i < count; i++) {
        printf("Name: %s\n", players[i].name);
        printf("Age: %d\n", players[i].age);
        printf("Team: %s\n", players[i].team);
        printf("Matches Played: %d\n", players[i].matchesPlayed);
        printf("Runs: %d\n", players[i].totalRuns);
        printf("Wickets: %d\n\n", players[i].totalWickets);
    }
}

void displayTopPlayer(struct Player *players, int count) {
    int highestRunsIndex = 0;
    int highestWicketsIndex = 0;

    for (int i = 1; i < count; i++) {
        if (players[i].totalRuns > players[highestRunsIndex].totalRuns) {
            highestRunsIndex = i;
        }
        if (players[i].totalWickets > players[highestWicketsIndex].totalWickets) {
            highestWicketsIndex = i;
        }
    }

    printf("Top Batsman (Most Runs): %s with %d runs\n", players[highestRunsIndex].name,
players[highestRunsIndex].totalRuns);
    printf("Top Bowler (Most Wickets): %s with %d wickets\n", players[highestWicketsIndex].name,
players[highestWicketsIndex].totalWickets);
}
```

```
int main() {
    struct Player *players = NULL;
    int count = 0;
    int choice;

    while (1) {
        printf("\n1. Add Player\n2. Display Players\n3. Display Top Players\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            addPlayer(&players, &count);
        } else if (choice == 2) {
            displayPlayers(players, count);
        } else if (choice == 3) {
            displayTopPlayer(players, count);
        } else if (choice == 4) {
            free(players);
            break;
        } else {
            printf("Invalid choice! Try again.\n");
        }
    }

    return 0;
}
```

o/p:

```
Enter your choice: 1
Enter player's name: christo
Enter player's age: 21
Enter player's team: a
```

Enter number of matches played: 234

Enter total runs: 1234

Enter total wickets: 34

1. Add Player

2. Display Players

3. Display Top Players

4. Exit

Enter your choice: 1

Enter player's name: jeron

Enter player's age: 23

Enter player's team: b

Enter number of matches played: 125

Enter total runs: 1123

Enter total wickets: 23

1. Add Player

2. Display Players

3. Display Top Players

4. Exit

Enter your choice: 2

Name: christo

Age: 21

Team: a

Matches Played: 234

Runs: 1234

Wickets: 34

Name: jeron

Age: 23

Team: b

Matches Played: 125

Runs: 1123

Wickets: 23

1. Add Player

2. Display Players

3. Display Top Players

4. Exit

Enter your choice: 3

Top Batsman (Most Runs): christo with 1234 runs

Top Bowler (Most Wickets): christo with 34 wickets

1. Add Player

2. Display Players

3. Display Top Players

4. Exit

Enter your choice: 4

Problem 2: Tournament Fixture Scheduler

Requirements:

- Create a structure Match with members:
    - char team1[30]
    - char team2[30]
    - char date[11] (format: YYYY-MM-DD)
    - char venue[50]
- Functions to:
- Schedule a new match between two teams.
- Display all scheduled matches.
- Search for matches scheduled on a specific date.
- Cancel a match by specifying both team names and the date.

- Ensure that the match schedule is stored in an array, with the ability to dynamically adjust its size.

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

struct Match {
    char team1[30];
    char team2[30];
    char date[11];  // Format: YYYY-MM-DD
    char venue[50];
};

void scheduleMatch(struct Match **matches, int *matchCount) {
    *matches = realloc(*matches, (*matchCount + 1) * sizeof(struct Match));

    printf("Enter Team 1: ");
    scanf("%s", (*matches)[*matchCount].team1);
    printf("Enter Team 2: ");
    scanf("%s", (*matches)[*matchCount].team2);
    printf("Enter Match Date (YYYY-MM-DD): ");
    scanf("%s", (*matches)[*matchCount].date);
    printf("Enter Venue: ");
    getchar();  // To consume the newline character left by previous input
    fgets((*matches)[*matchCount].venue, sizeof((*matches)[*matchCount].venue), stdin);
    (*matches)[*matchCount].venue[strcspn((*matches)[*matchCount].venue, "\n")] = 0;  // Remove newline

    (*matchCount)++;
}

void displayMatches(struct Match *matches, int matchCount) {
```

```
    if (matchCount == 0) {
        printf("No matches scheduled.\n");
        return;
    }

    for (int i = 0; i < matchCount; i++) {
        printf("Match %d:\n", i + 1);
        printf("Team 1: %s\n", matches[i].team1);
        printf("Team 2: %s\n", matches[i].team2);
        printf("Date: %s\n", matches[i].date);
        printf("Venue: %s\n", matches[i].venue);
        printf("\n");
    }
}

void searchMatchByDate(struct Match *matches, int matchCount, const char *date) {
    int found = 0;
    for (int i = 0; i < matchCount; i++) {
        if (strcmp(matches[i].date, date) == 0) {
            printf("Match between %s and %s on %s at %s\n", matches[i].team1, matches[i].team2,
matches[i].date, matches[i].venue);
            found = 1;
        }
    }
    if (!found) {
        printf("No matches found on this date.\n");
    }
}

void cancelMatch(struct Match *matches, int *matchCount, const char *team1, const char *team2,
const char *date) {
    int found = 0;
```

```
    for (int i = 0; i < *matchCount; i++) {
        if (strcmp(matches[i].team1, team1) == 0 && strcmp(matches[i].team2, team2) == 0 &&
strcmp(matches[i].date, date) == 0) {
            for (int j = i; j < *matchCount - 1; j++) {
                matches[j] = matches[j + 1]; // Shift elements left
            }
            (*matchCount)--; // Decrease match count
            printf("Match between %s and %s on %s has been canceled.\n", team1, team2, date);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("No such match found to cancel.\n");
    }
}

int main() {
    struct Match *matches = NULL;
    int matchCount = 0;
    int choice;
    char date[11], team1[30], team2[30];

    while (1) {
        printf("\nTournament Fixture Scheduler\n");
        printf("1. Schedule a Match\n");
        printf("2. Display All Matches\n");
        printf("3. Search for Matches by Date\n");
        printf("4. Cancel a Match\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
```

```
        scanf("%d", &choice);

        if (choice == 1) {
            scheduleMatch(&matches, &matchCount);
        } else if (choice == 2) {
            displayMatches(matches, matchCount);
        } else if (choice == 3) {
            printf("Enter date (YYYY-MM-DD) to search: ");
            scanf("%s", date);
            searchMatchByDate(matches, matchCount, date);
        } else if (choice == 4) {
            printf("Enter Team 1: ");
            scanf("%s", team1);
            printf("Enter Team 2: ");
            scanf("%s", team2);
            printf("Enter Match Date (YYYY-MM-DD): ");
            scanf("%s", date);
            cancelMatch(matches, &matchCount, team1, team2, date);
        } else if (choice == 5) {
            free(matches);  // Free allocated memory before exit
            printf("Exiting...\n");
            return 0;
        } else {
            printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

o/p:

```
Tournament Fixture Scheduler
1. Schedule a Match
```

2. Display All Matches

3. Search for Matches by Date

4. Cancel a Match

5. Exit

Enter your choice: 1

Enter Team 1: TeamA

Enter Team 2: TeamB

Enter Match Date (YYYY-MM-DD): 2025-02-20

Enter Venue: Stadium A

Tournament Fixture Scheduler

1. Schedule a Match

2. Display All Matches

3. Search for Matches by Date

4. Cancel a Match

5. Exit

Enter your choice: 2

Match 1:

Team 1: TeamA

Team 2: TeamB

Date: 2025-02-20

Venue: Stadium A

Tournament Fixture Scheduler

1. Schedule a Match

2. Display All Matches

3. Search for Matches by Date

4. Cancel a Match

5. Exit

Enter your choice: 3

Enter date (YYYY-MM-DD) to search: 2025-02-20

Match between TeamA and TeamB on 2025-02-20 at Stadium A

Tournament Fixture Scheduler

1. Schedule a Match

2. Display All Matches

3. Search for Matches by Date

4. Cancel a Match

5. Exit

Enter your choice: 4

Enter Team 1: TeamA

Enter Team 2: TeamB

Enter Match Date (YYYY-MM-DD): 2025-02-20

Match between TeamA and TeamB on 2025-02-20 has been canceled.

Tournament Fixture Scheduler

1. Schedule a Match

2. Display All Matches

3. Search for Matches by Date

4. Cancel a Match

5. Exit

Enter your choice: 5

Exiting

Problem 3: Sports Event Medal Tally

Requirements:

- Define a structure CountryMedalTally with members:
    - char country[30]
    - int gold
    - int silver
    - int bronze
- Functions to:
- Add a new country's medal tally.

- Update the medal count for a country.
- Display the medal tally for all countries.
- Find and display the country with the highest number of gold medals.
- Use an array to store the medal tally, and resize the array dynamically as new countries are added.

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

struct CountryMedalTally {
    char country[30];
    int gold;
    int silver;
    int bronze;
};

void addCountryMedalTally(struct CountryMedalTally **medals, int *count) {
    *medals = realloc(*medals, (*count + 1) * sizeof(struct CountryMedalTally));

    printf("Enter country name: ");
    scanf("%s", (*medals)[*count].country);

    printf("Enter number of Gold medals: ");
    scanf("%d", &(*medals)[*count].gold);

    printf("Enter number of Silver medals: ");
    scanf("%d", &(*medals)[*count].silver);

    printf("Enter number of Bronze medals: ");
    scanf("%d", &(*medals)[*count].bronze);
```

```
    (*count)++;
}

void updateMedalTally(struct CountryMedalTally *medals, int count, const char *country) {
    for (int i = 0; i < count; i++) {
        if (strcmp(medals[i].country, country) == 0) {
            printf("Enter new number of Gold medals: ");
            scanf("%d", &medals[i].gold);

            printf("Enter new number of Silver medals: ");
            scanf("%d", &medals[i].silver);

            printf("Enter new number of Bronze medals: ");
            scanf("%d", &medals[i].bronze);
            return;
        }
    }
    printf("Country not found!\n");
}

void displayMedalTally(struct CountryMedalTally *medals, int count) {
    printf("\nMedal Tally:\n");
    for (int i = 0; i < count; i++) {
        printf("Country: %s\n", medals[i].country);
        printf("Gold: %d\n", medals[i].gold);
        printf("Silver: %d\n", medals[i].silver);
        printf("Bronze: %d\n", medals[i].bronze);
        printf("\n");
    }
}
```

```
void findHighestGold(struct CountryMedalTally *medals, int count) {
    if (count == 0) {
        printf("No countries available.\n");
        return;
    }

    int maxGold = medals[0].gold;
    int maxGoldIndex = 0;

    for (int i = 1; i < count; i++) {
        if (medals[i].gold > maxGold) {
            maxGold = medals[i].gold;
            maxGoldIndex = i;
        }
    }

    printf("Country with highest Gold medals: %s\n", medals[maxGoldIndex].country);
    printf("Gold: %d\n", medals[maxGoldIndex].gold);
    printf("Silver: %d\n", medals[maxGoldIndex].silver);
    printf("Bronze: %d\n", medals[maxGoldIndex].bronze);
}

int main() {
    struct CountryMedalTally *medals = NULL;
    int count = 0;
    int choice;
    char country[30];

    while (1) {
        printf("\n1. Add Country Medal Tally\n");
        printf("2. Update Medal Tally for a Country\n");
```

```
        printf("3. Display Medal Tally for All Countries\n");
        printf("4. Find Country with Highest Gold Medals\n");
        printf("5. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            addCountryMedalTally(&medals, &count);
        } else if (choice == 2) {
            printf("Enter country to update: ");
            scanf("%s", country);
            updateMedalTally(medals, count, country);
        } else if (choice == 3) {
            displayMedalTally(medals, count);
        } else if (choice == 4) {
            findHighestGold(medals, count);
        } else if (choice == 5) {
            free(medals);
            printf("Exiting...\n");
            return 0;
        } else {
            printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```

o/p:

1. Add Country Medal Tally
2. Update Medal Tally for a Country

3. Display Medal Tally for All Countries

4. Find Country with Highest Gold Medals

5. Exit

Enter your choice: 1

Enter country name: USA

Enter number of Gold medals: 25

Enter number of Silver medals: 10

Enter number of Bronze medals: 15

1. Add Country Medal Tally

2. Update Medal Tally for a Country

3. Display Medal Tally for All Countries

4. Find Country with Highest Gold Medals

5. Exit

Enter your choice: 1

Enter country name: India

Enter number of Gold medals: 15

Enter number of Silver medals: 10

Enter number of Bronze medals: 5

1. Add Country Medal Tally

2. Update Medal Tally for a Country

3. Display Medal Tally for All Countries

4. Find Country with Highest Gold Medals

5. Exit

Enter your choice: 3

Medal Tally:

Country: USA

Gold: 25

Silver: 10

Bronze: 15

Country: India

Gold: 15

Silver: 10

Bronze: 5

1. Add Country Medal Tally

2. Update Medal Tally for a Country

3. Display Medal Tally for All Countries

4. Find Country with Highest Gold Medals

5. Exit

Enter your choice: 4

Country with highest Gold medals: USA

Gold: 25

Silver: 10

Bronze: 15

Problem 4: Athlete Performance Tracker

Requirements:

- Create a structure Athlete with fields:
    - char athleteID[10]
    - char name[50]
    - char sport[30]
    - float personalBest
    - float lastPerformance
- Functions to:
- Add a new athlete to the system.
- Update an athlete's last performance.
- Display all athletes in a specific sport.
- Identify and display athletes who have set a new personal best in their last performance.
- Utilize dynamic memory allocation to manage athlete data in an expandable array.

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

struct Athlete {
    char athleteID[10];
    char name[50];
    char sport[30];
    float personalBest;
    float lastPerformance;
};

void addAthlete(struct Athlete **athletes, int *count) {
    *athletes = realloc(*athletes, (*count + 1) * sizeof(struct Athlete));

    printf("Enter athlete ID: ");
    scanf("%s", (*athletes)[*count].athleteID);

    printf("Enter athlete name: ");
    scanf(" %[^\n]", (*athletes)[*count].name);

    printf("Enter athlete sport: ");
    scanf("%s", (*athletes)[*count].sport);

    printf("Enter athlete personal best: ");
    scanf("%f", &(*athletes)[*count].personalBest);

    printf("Enter athlete last performance: ");
    scanf("%f", &(*athletes)[*count].lastPerformance);
```

```
    (*count)++;
}

void updatePerformance(struct Athlete *athletes, int count, const char *athleteID) {
    for (int i = 0; i < count; i++) {
        if (strcmp(athletes[i].athleteID, athleteID) == 0) {
            printf("Enter new last performance for athlete %s: ", athleteID);
            scanf("%f", &athletes[i].lastPerformance);
            return;
        }
    }
    printf("Athlete with ID %s not found.\n", athleteID);
}

void displayAthletesInSport(struct Athlete *athletes, int count, const char *sport) {
    printf("\nAthletes in sport %s:\n", sport);
    for (int i = 0; i < count; i++) {
        if (strcmp(athletes[i].sport, sport) == 0) {
            printf("ID: %s, Name: %s, Personal Best: %.2f, Last Performance: %.2f\n",
                athletes[i].athleteID, athletes[i].name, athletes[i].personalBest, athletes[i].lastPerformance);
        }
    }
}

void displayNewPersonalBest(struct Athlete *athletes, int count) {
    printf("\nAthletes with new personal best in last performance:\n");
    for (int i = 0; i < count; i++) {
        if (athletes[i].lastPerformance > athletes[i].personalBest) {
            printf("ID: %s, Name: %s, Sport: %s, Personal Best: %.2f, Last Performance: %.2f\n",
```

```
            athletes[i].athleteID, athletes[i].name, athletes[i].sport, athletes[i].personalBest,
athletes[i].lastPerformance);
        }
    }
}

int main() {
    struct Athlete *athletes = NULL;
    int count = 0;
    int choice;
    char sport[30];
    char athleteID[10];

    while (1) {
        printf("\n1. Add New Athlete\n");
        printf("2. Update Athlete Performance\n");
        printf("3. Display Athletes in a Sport\n");
        printf("4. Display Athletes with New Personal Best\n");
        printf("5. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            addAthlete(&athletes, &count);
        } else if (choice == 2) {
            printf("Enter athlete ID to update performance: ");
            scanf("%s", athleteID);
            updatePerformance(athletes, count, athleteID);
        } else if (choice == 3) {
            printf("Enter sport to display athletes: ");
```

```
            scanf("%s", sport);
            displayAthletesInSport(athletes, count, sport);
        } else if (choice == 4) {
            displayNewPersonalBest(athletes, count);
        } else if (choice == 5) {
            free(athletes);
            printf("Exiting...\n");
            return 0;
        } else {
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

o/p:

```
1. Add New Athlete
2. Update Athlete Performance
3. Display Athletes in a Sport
4. Display Athletes with New Personal Best
5. Exit
Enter your choice: 1
Enter athlete ID: A001
Enter athlete name: John Doe
Enter athlete sport: Swimming
Enter athlete personal best: 50.25
Enter athlete last performance: 51.30

1. Add New Athlete
2. Update Athlete Performance
3. Display Athletes in a Sport
```

4. Display Athletes with New Personal Best

5. Exit

Enter your choice: 1

Enter athlete ID: A002

Enter athlete name: Sarah Smith

Enter athlete sport: Running

Enter athlete personal best: 12.50

Enter athlete last performance: 12.40

1. Add New Athlete

2. Update Athlete Performance

3. Display Athletes in a Sport

4. Display Athletes with New Personal Best

5. Exit

Enter your choice: 4

Athletes with new personal best in last performance:

ID: A001, Name: John Doe, Sport: Swimming, Personal Best: 50.25, Last Performance: 51.30

1. Add New Athlete

2. Update Athlete Performance

3. Display Athletes in a Sport

4. Display Athletes with New Personal Best

5. Exit

Enter your choice: 5

Exiting...

Problem 5: Sports Equipment Inventory System

Requirements:

- Define a structure Equipment with members:
    - char equipmentID[10]
    - char name[30]

    - char category[20] (e.g., balls, rackets)
    - int quantity
    - float pricePerUnit
- Functions to:
- Add new equipment to the inventory.
- Update the quantity of existing equipment.
- Display all equipment in a specific category.
- Calculate the total value of equipment in the inventory.
- Store the inventory data in a dynamically allocated array and ensure proper resizing when needed.

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

struct Equipment {
    char equipmentID[10];
    char name[30];
    char category[20];
    int quantity;
    float pricePerUnit;
};

void addEquipment(struct Equipment **inventory, int *count) {
    *inventory = realloc(*inventory, (*count + 1) * sizeof(struct Equipment));

    printf("Enter equipment ID: ");
    scanf("%s", (*inventory)[*count].equipmentID);

    printf("Enter equipment name: ");
    scanf(" %[^\n]", (*inventory)[*count].name);
```

```
    printf("Enter equipment category: ");
    scanf("%s", (*inventory)[*count].category);

    printf("Enter equipment quantity: ");
    scanf("%d", &(*inventory)[*count].quantity);

    printf("Enter price per unit: ");
    scanf("%f", &(*inventory)[*count].pricePerUnit);

    (*count)++;
}

void updateQuantity(struct Equipment *inventory, int count, const char *equipmentID, int
newQuantity) {
    for (int i = 0; i < count; i++) {
        if (strcmp(inventory[i].equipmentID, equipmentID) == 0) {
            inventory[i].quantity = newQuantity;
            printf("Quantity updated successfully.\n");
            return;
        }
    }
    printf("Equipment with ID %s not found.\n", equipmentID);
}

void displayEquipmentInCategory(struct Equipment *inventory, int count, const char *category) {
    printf("\nEquipment in category %s:\n", category);
    for (int i = 0; i < count; i++) {
        if (strcmp(inventory[i].category, category) == 0) {
            printf("ID: %s, Name: %s, Quantity: %d, Price per unit: %.2f\n",
                inventory[i].equipmentID, inventory[i].name, inventory[i].quantity,
inventory[i].pricePerUnit);
        }
```

```
    }
}

float calculateTotalValue(struct Equipment *inventory, int count) {
    float totalValue = 0;
    for (int i = 0; i < count; i++) {
        totalValue += inventory[i].quantity * inventory[i].pricePerUnit;
    }
    return totalValue;
}

int main() {
    struct Equipment *inventory = NULL;
    int count = 0;
    int choice;
    char category[20];
    char equipmentID[10];
    int newQuantity;

    while (1) {
        printf("\n1. Add New Equipment\n");
        printf("2. Update Equipment Quantity\n");
        printf("3. Display Equipment in a Category\n");
        printf("4. Calculate Total Value of Inventory\n");
        printf("5. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            addEquipment(&inventory, &count);
```

```
        } else if (choice == 2) {
            printf("Enter equipment ID to update quantity: ");
            scanf("%s", equipmentID);
            printf("Enter new quantity: ");
            scanf("%d", &newQuantity);
            updateQuantity(inventory, count, equipmentID, newQuantity);
        } else if (choice == 3) {
            printf("Enter category to display equipment: ");
            scanf("%s", category);
            displayEquipmentInCategory(inventory, count, category);
        } else if (choice == 4) {
            float totalValue = calculateTotalValue(inventory, count);
            printf("Total value of equipment in inventory: %.2f\n", totalValue);
        } else if (choice == 5) {
            free(inventory);
            printf("Exiting...\n");
            return 0;
        } else {
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

o/p:

```
1. Add New Equipment
2. Update Equipment Quantity
3. Display Equipment in a Category
4. Calculate Total Value of Inventory
5. Exit
Enter your choice: 1
```

Enter equipment ID: E001

Enter equipment name: Football

Enter equipment category: Balls

Enter equipment quantity: 10

Enter price per unit: 15.5

1. Add New Equipment

2. Update Equipment Quantity

3. Display Equipment in a Category

4. Calculate Total Value of Inventory

5. Exit

Enter your choice: 1

Enter equipment ID: E002

Enter equipment name: Tennis Racket

Enter equipment category: Rackets

Enter equipment quantity: 5

Enter price per unit: 45.75

1. Add New Equipment

2. Update Equipment Quantity

3. Display Equipment in a Category

4. Calculate Total Value of Inventory

5. Exit

Enter your choice: 3

Enter category to display equipment: Balls

Equipment in category Balls:

ID: E001, Name: Football, Quantity: 10, Price per unit: 15.50

1. Add New Equipment

2. Update Equipment Quantity

3. Display Equipment in a Category

4. Calculate Total Value of Inventory

5. Exit

Enter your choice: 4

Total value of equipment in inventory: 283.75

1. Add New Equipment

2. Update Equipment Quantity

3. Display Equipment in a Category

4. Calculate Total Value of Inventory

5. Exit

Enter your choice: 5

Exiting...

Problem 1: Research Paper Database Management

Requirements:

Define a structure ResearchPaper with the following members:

char title[100]

char author[50]

char journal[50]

int year

char DOI[30]

Functions to:

Add a new research paper to the database.

Update the details of an existing paper using its DOI.

Display all papers published in a specific journal.

Find and display the most recent papers published by a specific author.

Use dynamic memory allocation to store and manage the research papers in an array, resizing it as needed.

```
#include <stdio.h>

#include <stdlib.h>
```

```
#include <string.h>

struct ResearchPaper {
    char title[100];
    char author[50];
    char journal[50];
    int year;
    char DOI[30];
};

void addResearchPaper(struct ResearchPaper **papers, int *count) {
    *papers = realloc(*papers, (*count + 1) * sizeof(struct ResearchPaper));
    printf("Enter title, author, journal, year, DOI: ");
    scanf("%s %s %s %d %s", (*papers)[*count].title, (*papers)[*count].author,
(*papers)[*count].journal, &(*papers)[*count].year, (*papers)[*count].DOI);
    (*count)++;
}

void updateResearchPaper(struct ResearchPaper *papers, int count, char *DOI) {
    for (int i = 0; i < count; i++) {
        if (strcmp(papers[i].DOI, DOI) == 0) {
            printf("Enter new title, author, journal, year, DOI: ");
            scanf("%s %s %s %d %s", papers[i].title, papers[i].author, papers[i].journal, &papers[i].year,
papers[i].DOI);
            printf("Paper details updated.\n");
            return;
        }
    }
    printf("Paper with DOI %s not found.\n", DOI);
}

void displayPapersByJournal(struct ResearchPaper *papers, int count, char *journal) {
```

```
    for (int i = 0; i < count; i++) {
        if (strcmp(papers[i].journal, journal) == 0) {
            printf("%s by %s (%d) DOI: %s\n", papers[i].title, papers[i].author, papers[i].year,
papers[i].DOI);
        }
    }
}

void displayRecentPapersByAuthor(struct ResearchPaper *papers, int count, char *author) {
    int maxYear = -1;
    for (int i = 0; i < count; i++) {
        if (strcmp(papers[i].author, author) == 0 && papers[i].year > maxYear) {
            maxYear = papers[i].year;
        }
    }
    for (int i = 0; i < count; i++) {
        if (strcmp(papers[i].author, author) == 0 && papers[i].year == maxYear) {
            printf("%s by %s (%d) DOI: %s\n", papers[i].title, papers[i].author, papers[i].year,
papers[i].DOI);
        }
    }
}

int main() {
    struct ResearchPaper *papers = NULL;
    int count = 0;
    int choice;
    char DOI[30], journal[50], author[50];

    while (1) {
        printf("\n1. Add Paper\n2. Update Paper\n3. Display by Journal\n4. Display Recent by
Author\n5. Exit\nChoice: ");
```

```
        scanf("%d", &choice);

        if (choice == 1) {
            addResearchPaper(&papers, &count);
        } else if (choice == 2) {
            printf("Enter DOI of paper to update: ");
            scanf("%s", DOI);
            updateResearchPaper(papers, count, DOI);
        } else if (choice == 3) {
            printf("Enter journal name: ");
            scanf("%s", journal);
            displayPapersByJournal(papers, count, journal);
        } else if (choice == 4) {
            printf("Enter author name: ");
            scanf("%s", author);
            displayRecentPapersByAuthor(papers, count, author);
        } else if (choice == 5) {
            free(papers);
            break;
        }
    }

    return 0;
}
```

o/p:

```
1. Add Paper
2. Update Paper
3. Display by Journal
4. Display Recent by Author
5. Exit
Choice: 1
```

Enter title, author, journal, year, DOI: AI John AI_Journal 2021 10.1234/ai2021

1. Add Paper

2. Update Paper

3. Display by Journal

4. Display Recent by Author

5. Exit

Choice: 2

Enter DOI of paper to update: 10.1234/ai2021

Enter new title, author, journal, year, DOI: AI_Research John AI_Journal 2022 10.5678/ai2022

Paper details updated.

1. Add Paper

2. Update Paper

3. Display by Journal

4. Display Recent by Author

5. Exit

Choice: 3

Enter journal name: AI_Journal

AI_Research by John (2022) DOI: 10.5678/ai2022

1. Add Paper

2. Update Paper

3. Display by Journal

4. Display Recent by Author

5. Exit

Choice: 4

Enter author name: John

AI_Research by John (2022) DOI: 10.5678/ai2022

Problem 2: Experimental Data Logger

Requirements:

- Create a structure Experiment with members:
    - char experimentID[10]
    - char researcher[50]
    - char startDate[11] (format: YYYY-MM-DD)
    - char endDate[11]
    - float results[10] (store up to 10 result readings)
- Functions to:
- Log a new experiment.
- Update the result readings of an experiment.
- Display all experiments conducted by a specific researcher.
- Calculate and display the average result for a specific experiment.
- Use a dynamically allocated array for storing experiments and manage resizing as more data is logged.

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

struct Experiment {
    char experimentID[10];
    char researcher[50];
    char startDate[11];
    char endDate[11];
    float results[10];
};

void logExperiment(struct Experiment **experiments, int *count) {
    *experiments = realloc(*experiments, (*count + 1) * sizeof(struct Experiment));
    printf("Enter experiment ID, researcher name, start date (YYYY-MM-DD), end date (YYYY-MM-DD):
");
    scanf("%s %s %s %s", (*experiments)[*count].experimentID, (*experiments)[*count].researcher,
                (*experiments)[*count].startDate, (*experiments)[*count].endDate);
    printf("Enter up to 10 results: ");
```

```
    for (int i = 0; i < 10; i++) {
        scanf("%f", &(*experiments)[*count].results[i]);
        if ((*experiments)[*count].results[i] == -1) break;
    }
    (*count)++;
}

void updateResults(struct Experiment *experiments, int count, char *experimentID) {
    for (int i = 0; i < count; i++) {
        if (strcmp(experiments[i].experimentID, experimentID) == 0) {
            printf("Enter new results for experiment ID %s:\n", experimentID);
            for (int j = 0; j < 10; j++) {
                scanf("%f", &experiments[i].results[j]);
                if (experiments[i].results[j] == -1) break;
            }
            printf("Results updated for experiment ID %s.\n", experimentID);
            return;
        }
    }
    printf("Experiment ID not found.\n");
}

void displayByResearcher(struct Experiment *experiments, int count, char *researcher) {
    for (int i = 0; i < count; i++) {
        if (strcmp(experiments[i].researcher, researcher) == 0) {
            printf("Experiment ID: %s, Start Date: %s, End Date: %s\n", experiments[i].experimentID,
experiments[i].startDate, experiments[i].endDate);
        }
    }
}
```

```
void calculateAverage(struct Experiment *experiments, int count, char *experimentID) {
    for (int i = 0; i < count; i++) {
        if (strcmp(experiments[i].experimentID, experimentID) == 0) {
            float sum = 0;
            int validResults = 0;
            for (int j = 0; j < 10; j++) {
                if (experiments[i].results[j] == -1) break;
                sum += experiments[i].results[j];
                validResults++;
            }
            printf("Average result: %.2f\n", validResults > 0 ? sum / validResults : 0);
            return;
        }
    }
    printf("Experiment not found.\n");
}

int main() {
    struct Experiment *experiments = NULL;
    int count = 0, choice;
    char researcher[50], experimentID[10];

    while (1) {
        printf("\n1. Log Experiment\n2. Update Experiment Results\n3. Display by Researcher\n4.
Calculate Average\n5. Exit\nChoice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            logExperiment(&experiments, &count);
        } else if (choice == 2) {
            printf("Enter experiment ID to update results: ");
```

```
            scanf("%s", experimentID);
            updateResults(experiments, count, experimentID);
        } else if (choice == 3) {
            printf("Enter researcher name: ");
            scanf("%s", researcher);
            displayByResearcher(experiments, count, researcher);
        } else if (choice == 4) {
            printf("Enter experiment ID to calculate average: ");
            scanf("%s", experimentID);
            calculateAverage(experiments, count, experimentID);
        } else if (choice == 5) {
            free(experiments);
            break;
        }
    }

    return 0;
}
```

o/p:

1. Log Experiment

2. Update Experiment Results

3. Display by Researcher

4. Calculate Average

5. Exit

Choice: 1

Enter experiment ID, researcher name, start date (YYYY-MM-DD), end date (YYYY-MM-DD): EXP001 Alice 2023-01-01 2023-01-10

Enter up to 10 results: 10 20 30 -1

1. Log Experiment

2. Update Experiment Results

3. Display by Researcher

4. Calculate Average

5. Exit

Choice: 2

Enter experiment ID to update results: EXP001

Enter new results for experiment ID EXP00: 15 25 35 -1

Results updated for experiment ID EXP001.

1. Log Experiment

2. Update Experiment Results

3. Display by Researcher

4. Calculate Average

5. Exit

Choice: 4

Enter experiment ID to calculate average: EXP001

Average result: 25.00

Problem 3: Grant Application Tracker

Requirements:

- Define a structure GrantApplication with the following members:
    - char applicationID[10]
    - char applicantName[50]
    - char projectTitle[100]
    - float requestedAmount
    - char status[20] (e.g., Submitted, Approved, Rejected)
- Functions to:
- Add a new grant application.
- Update the status of an application.
- Display all applications requesting an amount greater than a specified value.
- Find and display applications that are currently "Approved."
- Store the grant applications in a dynamically allocated array, resizing it as necessary.

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

```
#include <stdlib.h>
#include <string.h>
struct GrantApplication {
    char applicationID[10], applicantName[50], projectTitle[100], status[20];
    float requestedAmount;
};

void addApplication(struct GrantApplication **apps, int *count) {
    *apps = realloc(*apps, (*count + 1) * sizeof(struct GrantApplication));
    printf("Enter ID, Name, Project, Amount, Status: ");
    scanf("%s %s %[^\n]s %f %s", (*apps)[*count].applicationID, (*apps)[*count].applicantName,
        (*apps)[*count].projectTitle, &(*apps)[*count].requestedAmount, (*apps)[*count].status);
    (*count)++;
}

void updateStatus(struct GrantApplication *apps, int count, char *id, char *status) {
    for (int i = 0; i < count; i++) {
        if (strcmp(apps[i].applicationID, id) == 0) {
            strcpy(apps[i].status, status);
            printf("Updated application %s to status %s.\n", id, status);
            return;
        }
    }
    printf("Application %s not found.\n", id);
}

void displayByAmount(struct GrantApplication *apps, int count, float amount) {
    for (int i = 0; i < count; i++) {
        if (apps[i].requestedAmount > amount) {
            printf("ID: %s, Name: %s, Project: %s, Amount: %.2f, Status: %s\n",
                apps[i].applicationID, apps[i].applicantName, apps[i].projectTitle,
```

```
            apps[i].requestedAmount, apps[i].status);
        }
    }
}

void displayApproved(struct GrantApplication *apps, int count) {
    for (int i = 0; i < count; i++) {
        if (strcmp(apps[i].status, "Approved") == 0) {
            printf("ID: %s, Name: %s, Project: %s, Amount: %.2f, Status: %s\n",
                apps[i].applicationID, apps[i].applicantName, apps[i].projectTitle,
                apps[i].requestedAmount, apps[i].status);
        }
    }
}

int main() {
    struct GrantApplication *applications = NULL;
    int count = 0, choice;
    char id[10], status[20];
    float amount;

    while (1) {
        printf("\n1. Add Application\n2. Update Status\n3. Display by Amount\n4. Display Approved\n5. Exit\nChoice: ");
        scanf("%d", &choice);
        if (choice == 1) addApplication(&applications, &count);
        else if (choice == 2) {
            printf("Enter ID and new status: ");
            scanf("%s %s", id, status);
            updateStatus(applications, count, id, status);
        }
```

```
        else if (choice == 3) {
            printf("Enter minimum requested amount: ");
            scanf("%f", &amount);
            displayByAmount(applications, count, amount);
        }
        else if (choice == 4) displayApproved(applications, count);
        else break;
    }

    free(applications);
    return 0;
}
```

o/p:

```
1. Add Application
2. Update Status
3. Display by Amount
4. Display Approved
5. Exit
Choice: 1
Enter ID, Name, Project, Amount, Status: APP001 John SolarEnergy 5000 Submitted

1. Add Application
2. Update Status
3. Display by Amount
4. Display Approved
5. Exit
Choice: 2
Enter ID and new status: APP001 Approved
Updated application APP001 to status Approved.

1. Add Application
```

2. Update Status

3. Display by Amount

4. Display Approved

5. Exit

Choice: 4

ID: APP001, Name: John, Project: SolarEnergy, Amount: 5000.00, Status: Approved

Problem 4: Research Collaborator Management

Requirements:

- Create a structure Collaborator with members:
    - char collaboratorID[10]
    - char name[50]
    - char institution[50]
    - char expertiseArea[30]
    - int numberOfProjects
- Functions to:
- Add a new collaborator to the database.
- Update the number of projects a collaborator is involved in.
- Display all collaborators from a specific institution.
- Find collaborators with expertise in a given area.
- Use dynamic memory allocation to manage the list of collaborators, allowing for expansion as more are added.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct Collaborator {

   char collaboratorID[10], name[50], institution[50], expertiseArea[30];

   int numberOfProjects;

};
```

```
void addCollaborator(struct Collaborator **collaborators, int *count) {
    *collaborators = realloc(*collaborators, (*count + 1) * sizeof(struct Collaborator));
    printf("Enter ID, Name, Institution, Expertise, Projects: ");
    scanf("%s %s %s %s %d", (*collaborators)[*count].collaboratorID, (*collaborators)[*count].name,
        (*collaborators)[*count].institution, (*collaborators)[*count].expertiseArea,
&(*collaborators)[*count].numberOfProjects);
    (*count)++;
}

void updateProjects(struct Collaborator *collaborators, int count, char *id, int newProjects) {
    for (int i = 0; i < count; i++) {
        if (strcmp(collaborators[i].collaboratorID, id) == 0) {
            collaborators[i].numberOfProjects = newProjects;
            printf("Updated %s with %d projects.\n", id, newProjects);
            return;
        }
    }
    printf("Collaborator %s not found.\n", id);
}

void displayByInstitution(struct Collaborator *collaborators, int count, char *institution) {
    for (int i = 0; i < count; i++) {
        if (strcmp(collaborators[i].institution, institution) == 0) {
            printf("%s %s %s %s %d\n", collaborators[i].collaboratorID, collaborators[i].name,
collaborators[i].institution,
                collaborators[i].expertiseArea, collaborators[i].numberOfProjects);
        }
    }
}

void findByExpertise(struct Collaborator *collaborators, int count, char *expertise) {
    for (int i = 0; i < count; i++) {
```

```
        if (strcmp(collaborators[i].expertiseArea, expertise) == 0) {
            printf("%s %s %s %d\n", collaborators[i].collaboratorID, collaborators[i].name,
collaborators[i].institution,
                collaborators[i].numberOfProjects);
        }
    }
}

int main() {
    struct Collaborator *collaborators = NULL;
    int count = 0, choice;
    char id[10], institution[50], expertise[30];
    int newProjects;

    while (1) {
        printf("\n1. Add Collaborator\n2. Update Projects\n3. Display by Institution\n4. Find by
Expertise\n5. Exit\nChoice: ");
        scanf("%d", &choice);
        if (choice == 1) addCollaborator(&collaborators, &count);
        else if (choice == 2) {
            printf("Enter ID and new number of projects: ");
            scanf("%s %d", id, &newProjects);
            updateProjects(collaborators, count, id, newProjects);
        }
        else if (choice == 3) {
            printf("Enter institution: ");
            scanf("%s", institution);
            displayByInstitution(collaborators, count, institution);
        }
        else if (choice == 4) {
            printf("Enter expertise: ");
            scanf("%s", expertise);
```

```
        findByExpertise(collaborators, count, expertise);
      }
      else break;
    }

    free(collaborators);
    return 0;
}
```

o/p:

```
1. Add Collaborator
2. Update Projects
3. Display by Institution
4. Find by Expertise
5. Exit
Choice: 1
Enter ID, Name, Institution, Expertise, Projects: C001 John UniversityX AI 5

1. Add Collaborator
2. Update Projects
3. Display by Institution
4. Find by Expertise
5. Exit
Choice: 2
Enter ID and new number of projects: C001 7
Updated C001 with 7 projects.

1. Add Collaborator
2. Update Projects
3. Display by Institution
4. Find by Expertise
5. Exit
```

Choice: 3

Enter institution: UniversityX

C001 John UniversityX AI 7

1. Add Collaborator

2. Update Projects

3. Display by Institution

4. Find by Expertise

5. Exit

Choice: 4

Enter expertise: AI

C001 John UniversityX AI 7

Problem 5: Scientific Conference Submission Tracker

Requirements:

- Define a structure ConferenceSubmission with the following:
    - char submissionID[10]
    - char authorName[50]
    - char paperTitle[100]
    - char conferenceName[50]
    - char submissionDate[11]
    - char status[20] (e.g., Pending, Accepted, Rejected)
- Functions to:
- Add a new conference submission.
- Update the status of a submission.
- Display all submissions to a specific conference.
- Find and display submissions by a specific author.
- Store the conference submissions in a dynamically allocated array, resizing the array as needed when more submissions are added.

```
#include <stdio.h>

#include <stdlib.h>
```

```
#include <string.h>

struct ConferenceSubmission {
    char submissionID[10], authorName[50], paperTitle[100], conferenceName[50],
submissionDate[11], status[20];
};

void addSubmission(struct ConferenceSubmission **submissions, int *count) {
    *submissions = realloc(*submissions, (*count + 1) * sizeof(struct ConferenceSubmission));
    printf("Enter ID, Author, Paper Title, Conference, Date, Status: ");
    scanf("%s %s %s %s %s %s", (*submissions)[*count].submissionID,
(*submissions)[*count].authorName,
        (*submissions)[*count].paperTitle, (*submissions)[*count].conferenceName,
(*submissions)[*count].submissionDate,
        (*submissions)[*count].status);
    (*count)++;
}

void updateStatus(struct ConferenceSubmission *submissions, int count, char *id, char *newStatus) {
    for (int i = 0; i < count; i++) {
        if (strcmp(submissions[i].submissionID, id) == 0) {
            strcpy(submissions[i].status, newStatus);
            printf("Updated submission %s with status %s.\n", id, newStatus);
            return;
        }
    }
    printf("Submission %s not found.\n", id);
}

void displayByConference(struct ConferenceSubmission *submissions, int count, char *conference) {
    for (int i = 0; i < count; i++) {
        if (strcmp(submissions[i].conferenceName, conference) == 0) {
```

```
        printf("%s %s %s %s %s\n", submissions[i].submissionID, submissions[i].authorName,
submissions[i].paperTitle,
            submissions[i].submissionDate, submissions[i].status);
    }
  }
}

void displayByAuthor(struct ConferenceSubmission *submissions, int count, char *author) {
  for (int i = 0; i < count; i++) {
    if (strcmp(submissions[i].authorName, author) == 0) {
        printf("%s %s %s %s %s\n", submissions[i].submissionID, submissions[i].authorName,
submissions[i].paperTitle,
            submissions[i].conferenceName, submissions[i].status);
    }
  }
}

int main() {
  struct ConferenceSubmission *submissions = NULL;
  int count = 0, choice;
  char id[10], conference[50], author[50], newStatus[20];

  while (1) {
    printf("\n1. Add Submission\n2. Update Status\n3. Display by Conference\n4. Display by
Author\n5. Exit\nChoice: ");
    scanf("%d", &choice);
    if (choice == 1) addSubmission(&submissions, &count);
    else if (choice == 2) {
      printf("Enter ID and new status: ");
      scanf("%s %s", id, newStatus);
      updateStatus(submissions, count, id, newStatus);
    }
```

```
        else if (choice == 3) {
            printf("Enter conference name: ");
            scanf("%s", conference);
            displayByConference(submissions, count, conference);
        }
        else if (choice == 4) {
            printf("Enter author name: ");
            scanf("%s", author);
            displayByAuthor(submissions, count, author);
        }
        else break;
    }

    free(submissions);
    return 0;
}
```

o/p:

```
1. Add Submission
2. Update Status
3. Display by Conference
4. Display by Author
5. Exit
Choice: 1
Enter ID, Author, Paper Title, Conference, Date, Status: S001 John AIConference "AI in Healthcare" 2023-05-01 Pending

1. Add Submission
2. Update Status
3. Display by Conference
4. Display by Author
5. Exit
```

Choice: 2

Enter ID and new status: S001 Accepted

Updated submission S001 with status Accepted.

1. Add Submission

2. Update Status

3. Display by Conference

4. Display by Author

5. Exit

Choice: 3

Enter conference name: AIConference

S001 John AI in Healthcare 2023-05-01 Accepted

1. Add Submission

2. Update Status

3. Display by Conference

4. Display by Author

5. Exit

Choice: 4

Enter author name: John

S001 John AI in Healthcare AIConference 2023-05-01 Accepted