Requirements

1. Define Data Types

Person Structure:

Create a structure Person to store details about an individual, including:

personID (integer): Unique identifier for the individual.

name (string): Full name of the individual.

age (integer): Age of the person.

weight (float): Weight in kilograms.

height (float): Height in meters.

bmi (float): Body Mass Index (calculated as BMI=weight / (height)^2

DietDetails Union:

Use a union DietDetails to store either:

calorieIntake (integer): Daily calorie intake in kilocalories.

waterIntake (float): Daily water intake in liters.

DietType Enumeration:

Use an enumeration DietType to classify diet focus:

WEIGHT_LOSS, WEIGHT_GAIN, or MAINTENANCE.

2. Features

Dynamic Memory Allocation:

Allocate memory dynamically for an array of Person structures based on the number of individuals.

Input and Output:

Input the details for each person, including their diet details and type.

Calculate and store the BMI for each person.

Display all individual details, categorized by diet type.

BMI Classification:

Based on BMI, classify individuals into categories:

Underweight: BMI < 18.5

Normal Weight: 18.5 ≤ BMI < 25

Overweight: 25 ≤ BMI < 30

Obese: BMI ≥ 30

Search:

Search for an individual by their personID or name and display their details.

Sorting:

Sort individuals by their BMI in ascending order.

Sort individuals by their age in descending order.

Health Metrics Analysis:

Calculate the average BMI of all individuals.

Calculate the total calorie and water intake of the group.

3. Typedef

Use typedef to define aliases for the Person and DietDetails structures and the DietType enumeration.

Program Requirements

1. Menu Options

Input Individual Details.

Display All Individuals Categorized by Diet Type.

Search for an Individual by ID or Name.

Sort Individuals by BMI or Age.

Display Health Metrics Analysis.

Exit.

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

typedef struct {
    int personID;
    char name[50];
    int age;
    float weight;
    float height;
    float bmi;
    union {
        int calorieIntake;
        float waterIntake;
    } dietDetails;
    enum {
        WEIGHT_LOSS, WEIGHT_GAIN, MAINTENANCE
    } dietType;
} Person;

void inputDetails(Person *people, int n);
void displayByDietType(Person *people, int n);
```

```
void searchByIDorName(Person *people, int n);
void sortByBMI(Person *people, int n);
void sortByAge(Person *people, int n);
void healthMetricsAnalysis(Person *people, int n);

char *getDietType(int dietType);
char *getBMIClassification(float bmi);

int main() {
  Person *people = NULL;
  int choice, n = 0;

  do {
    printf("\nMenu Options:\n");
    printf("1. Input Individual Details\n");
    printf("2. Display All Individuals Categorized by Diet Type\n");
    printf("3. Search for an Individual by ID or Name\n");
    printf("4. Sort Individuals by BMI\n");
    printf("5. Sort Individuals by Age\n");
    printf("6. Display Health Metrics Analysis\n");
    printf("7. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
      case 1:
        printf("Enter the number of individuals: ");
        scanf("%d", &n);
```

```
    people = (Person *)malloc(n * sizeof(Person));
    inputDetails(people, n);
    break;

case 2:
    if (people != NULL) {
        displayByDietType(people, n);
    } else {
        printf("No individuals to display. Please input details first.\n");
    }
    break;
case 3:
    if (people != NULL) {
        searchByIDorName(people, n);
    } else {
        printf("No individuals to search. Please input details first.\n");
    }
    break;
case 4:
    if (people != NULL) {
        sortByBMI(people, n);
    } else {
        printf("No individuals to sort. Please input details first.\n");
    }
    break;
case 5:
    if (people != NULL) {
        sortByAge(people, n);
```

```
            } else {
                printf("No individuals to sort. Please input details first.\n");
            }
            break;
        case 6:
            if (people != NULL) {
                healthMetricsAnalysis(people, n);
            } else {
                printf("No data available for analysis. Please input details first.\n");
            }
            break;
        case 7:
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 7);

  free(people);
  return 0;
}

void inputDetails(Person *people, int n) {
  for (int i = 0; i < n; i++) {
    printf("\nEnter details for individual %d:\n", i + 1);
    printf("Person ID: ");
    scanf("%d", &people[i].personID);
```

```
        printf("Name: ");
        scanf(" %[^\n]", people[i].name);
        printf("Age: ");
        scanf("%d", &people[i].age);
        printf("Weight (kg): ");
        scanf("%f", &people[i].weight);
        printf("Height (m): ");
        scanf("%f", &people[i].height);
        printf("Diet Type (0: WEIGHT_LOSS, 1: WEIGHT_GAIN, 2: MAINTENANCE): ");
        scanf("%d", &people[i].dietType);
        if (people[i].dietType == WEIGHT_LOSS) {
            printf("Daily Calorie Intake (kcal): ");
            scanf("%d", &people[i].dietDetails.calorieIntake);
        } else {
            printf("Daily Water Intake (liters): ");
            scanf("%f", &people[i].dietDetails.waterIntake);
        }
        people[i].bmi = people[i].weight / (people[i].height * people[i].height);
    }
}

void displayByDietType(Person *people, int n) {
    for (int i = 0; i < 3; i++) {
        printf("\nIndividuals with diet type %s:\n", getDietType(i));
        for (int j = 0; j < n; j++) {
            if (people[j].dietType == i) {
                printf("ID: %d, Name: %s, Age: %d, BMI: %.2f, Classification: %s\n",
                    people[j].personID, people[j].name, people[j].age, people[j].bmi,
```

```
                getBMIClassification(people[j].bmi));
        }
      }
    }
}

char *getDietType(int dietType) {
    switch (dietType) {
        case 0: return "WEIGHT LOSS";
        case 1: return "WEIGHT GAIN";
        case 2: return "MAINTENANCE";
        default: return "UNKNOWN";
    }
}

char *getBMIClassification(float bmi) {
    if (bmi < 18.5) return "Underweight";
    if (bmi < 25.0) return "Normal Weight";
    if (bmi < 30.0) return "Overweight";
    return "Obese";
}

void searchByIDorName(Person *people, int n) {
    int choice;
    printf("\nSearch by:\n1. ID\n2. Name\nEnter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
```

```
        int searchID;
        printf("Enter Person ID: ");
        scanf("%d", &searchID);
        for (int i = 0; i < n; i++) {
            if (people[i].personID == searchID) {
                printf("Found: ID: %d, Name: %s, Age: %d, BMI: %.2f\n",
                    people[i].personID, people[i].name, people[i].age, people[i].bmi);
                return;
            }
        }
    } else if (choice == 2) {
        char searchName[50];
        printf("Enter Name: ");
        scanf(" %[^\n]", searchName);
        for (int i = 0; i < n; i++) {
            if (strcmp(people[i].name, searchName) == 0) {
                printf("Found: ID: %d, Name: %s, Age: %d, BMI: %.2f\n",
                    people[i].personID, people[i].name, people[i].age, people[i].bmi);
                return;
            }
        }
    } else {
        printf("Invalid choice.\n");
    }
    printf("Individual not found.\n");
}

void sortByBMI(Person *people, int n) {
```

```
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (people[j].bmi > people[j + 1].bmi) {
                Person temp = people[j];
                people[j] = people[j + 1];
                people[j + 1] = temp;
            }
        }
    }
    printf("\nSorted by BMI:\n");
    for (int i = 0; i < n; i++) {
        printf("ID: %d, Name: %s, BMI: %.2f\n", people[i].personID, people[i].name,
people[i].bmi);
    }
}

void sortByAge(Person *people, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (people[j].age < people[j + 1].age) {
                Person temp = people[j];
                people[j] = people[j + 1];
                people[j + 1] = temp;
            }
        }
    }
    printf("\nSorted by Age:\n");
    for (int i = 0; i < n; i++) {
```

```
        printf("ID: %d, Name: %s, Age: %d\n", people[i].personID, people[i].name,
people[i].age);
    }
}

void healthMetricsAnalysis(Person *people, int n) {
    float totalBMI = 0.0, totalWater = 0.0;
    int totalCalories = 0;

    for (int i = 0; i < n; i++) {
        totalBMI += people[i].bmi;
        if (people[i].dietType == WEIGHT_LOSS) {
            totalCalories += people[i].dietDetails.calorieIntake;
        } else {
            totalWater += people[i].dietDetails.waterIntake;
        }
    }

    printf("\nHealth Metrics Analysis:\n");
    printf("Average BMI: %.2f\n", totalBMI / n);
    printf("Total Calorie Intake: %d kcal\n", totalCalories);
    printf("Total Water Intake: %.2f liters\n", totalWater);
}
```