Second Assessment

//Problem 14: Fitness Tracker

Specifications:

Variables: Activity type, duration, calories burned.

Static & Const: Static variable for total activities; const for maximum types.

Switch Case: Menu for adding, viewing, and analyzing activities.

Looping Statements: Loop through activity records.

Pointers: Pointer for activity data manipulation.

Functions: Separate functions for each fitness operation.

Arrays: Store activity details.

Structures: Structure for activity information.

Nested Structures: Nested structures for activity and health details.

Unions: Union for storing different activity metrics.

Nested Unions: Nested union for various health parameters.

Output Expectations: Display activities and health analysis.

Menu Example:

1. Add Activity

2. View Activities

3. Analyze Activities

4. Exit

```c
#include <stdio.h>

#include <string.h>

#define max_activities 100


struct Activity {

   char type[20];

   int duration;

   float caloriesBurned;

   int steps;

};


struct Activity activities[max_activities];

int totalActivities = 0;


void addActivity() {

   if (totalActivities >= max_activities) {

      printf("Activity limit reached.\n");

      return;

   }


   printf("Enter activity type: ");

   scanf("%s", activities[totalActivities].type);

   printf("Enter duration (in minutes): ");

   scanf("%d", &activities[totalActivities].duration);

   printf("Enter calories burned: ");

   scanf("%f", &activities[totalActivities].caloriesBurned);

   printf("Enter steps: ");

   scanf("%d", &activities[totalActivities].steps);


   totalActivities++;

}
```

```
void viewActivities() {
    if (totalActivities == 0) {
        printf("No activities to display.\n");
        return;
    }

    for (int i = 0; i < totalActivities; i++) {
        printf("Activity %d: Type: %s, Duration: %d mins, Calories: %.2f, Steps: %d\n",
                i + 1, activities[i].type, activities[i].duration,
                activities[i].caloriesBurned, activities[i].steps);
    }
}

void analyzeActivities() {
    if (totalActivities == 0) {
        printf("No activities to analyze.\n");
        return;
    }

    float totalCalories = 0;
    int totalDuration = 0;

    for (int i = 0; i <totalActivities; i++) {
        totalCalories +=activities[i].caloriesBurned;
        totalDuration +=activities[i].duration;
    }

    printf("Total activities: %d, Total duration: %d mins, Total calories: %.2f\n",
            totalActivities, totalDuration, totalCalories);
}
```

```
int main() {
    int choice;

    while (1) {
        printf("\n1. Add activity\n2. View activities\n3. Analyze activities\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addActivity();
                break;
            case 2:
                viewActivities();
                break;
            case 3:
                analyzeActivities();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```