PROGRAMMING FUNDAMENTALS
LAB 11 ASSIGNMENT

MASHHOOD RIAZ

24K – 0530

BSCS 1D

Tasks are as follows:
Take the following structures and create functions that perform CRUD (create, read, update delete) operations in a file or multiple files depending on the need based on your understanding. Add a comment that exlpains your choice.

```
struct player{
char name[20];
char team[20];
};

struct bowl{
char type[10]; // seemer, pacer, spinner// N/A
char arm[5]; //left or right
struct player ply;
};

struct bat{
char type[10]; // top order, middle order, lower order
char handed[8]; //lefty or righty
struct bowl ply2;
};

// suppose that you have to store data for 5 players with at least 3 bowlers. The rest will be N/A.
```

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Define the structures

struct player {

   char name[20];

   char team[20];

};


struct bowl {

   char type[10];  // Seamer, Pacer, Spinner, or N/A

   char arm[5];    // Left or Right

   struct player ply;  // Associated player (bowler)

};


struct bat {

   char type[10];  // Top order, Middle order, Lower order

   char handed[8]; // Lefty or Righty

   struct bowl ply2; // Associated bowler

};


// Function to create and write data to a file (Create operation)

void createPlayerData(const char *filename) {

   FILE *file = fopen(filename, "wb"); // Open file in binary write mode

   if (file == NULL) {

      printf("Error opening file for writing.\n");

      return;

   }
```

```
    struct bat players[5] = {
        {"Top Order", "Righty", {"Seamer", "Right", {"Mash", "Team A"}}},
        {"Middle Order", "Lefty", {"Pacer", "Left", {"Rafay", "Team B"}}},
        {"Lower Order", "Righty", {"Spinner", "Right", {"Abdullah", "Team A"}}},
        {"Top Order", "Lefty", {"N/A", "N/A", {"Akif", "Team C"}}},
        {"Middle Order", "Righty", {"N/A", "N/A", {"Sarim", "Team D"}}}
    };

    // Write player data to file
    fwrite(players, sizeof(struct bat), 5, file);

    fclose(file);
    printf("Player data created and saved to file.\n");
}

// Function to read and display player data from the file (Read operation)
void readPlayerData(const char *filename) {
    FILE *file = fopen(filename, "rb"); // Open file in binary read mode
    if (file == NULL) {
        printf("Error opening file for reading.\n");
        return;
    }

    struct bat players[5];

    // Read player data from the file
    fread(players, sizeof(struct bat), 5, file);
    // Display the read data
    for (int i = 0; i < 5; i++) {
        printf("Player %d: Name: %s, Team: %s, Batting: %s, Handed: %s, Bowling: %s, Arm: %s\n",
               i + 1, players[i].ply2.ply.name, players[i].ply2.ply.team,
               players[i].type, players[i].handed, players[i].ply2.type, players[i].ply2.arm);
    }

    fclose(file);
}

// Function to update a player's data (Update operation)
void updatePlayerData(const char *filename, int index, struct bat updatedPlayer) {
    FILE *file = fopen(filename, "rb+"); // Open file in binary read-write mode
    if (file == NULL) {
        printf("Error opening file for updating.\n");
        return;
    }

    struct bat players[5];

    // Read the current data
    fread(players, sizeof(struct bat), 5, file);

    // Update the specific player's data
    if (index >= 0 && index < 5) {
        players[index] = updatedPlayer;
        fseek(file, index * sizeof(struct bat), SEEK_SET); // Move the file pointer to the correct position
        fwrite(&updatedPlayer, sizeof(struct bat), 1, file); // Write updated data
        printf("Player %d updated.\n", index + 1);
    }

    fclose(file);
}

// Function to delete a player's data (Delete operation)
void deletePlayerData(const char *filename, int index) {
    FILE *file = fopen(filename, "rb+"); // Open file in binary read-write mode
    if (file == NULL) {
        printf("Error opening file for deletion.\n");
        return;
    }

    struct bat players[5];

    // Read the current data
    fread(players, sizeof(struct bat), 5, file);

    // "Delete" the player's data by shifting others and clearing the last one
    if (index >= 0 && index < 5) {
        for (int i = index; i < 4; i++) {
            players[i] = players[i + 1]; // Shift players to the left
        }
        memset(&players[4], 0, sizeof(struct bat)); // Clear the last player

        fseek(file, 0, SEEK_SET); // Reset file pointer to beginning
        fwrite(players, sizeof(struct bat), 5, file); // Rewrite the data to the file
        printf("Player %d deleted.\n", index + 1);
    }
    fclose(file);
}

// Main function to execute the operations
int main() {
    const char *filename = "players_data.dat";

    // Create player data (initial data)
    createPlayerData(filename);

    // Read and display the data
    printf("\nInitial Player Data:\n");
    readPlayerData(filename);

    // Update Player 2's batting order
    struct bat updatedPlayer = {"Lower Order", "Righty", {"Pacer", "Left", {"Rafay", "Team B"}}};
    updatePlayerData(filename, 1, updatedPlayer); // Update Player 2

    // Read and display the updated data
    printf("\nUpdated Player Data:\n");
    readPlayerData(filename);

    // Delete Player 4's data
    deletePlayerData(filename, 3); // Delete Player 4

    // Read and display the final data after deletion
    printf("\nFinal Player Data after Deletion:\n");
    readPlayerData(filename);
    return 0;
}
```