

# Assignment Instructions

## PART 1

To accomplish this task, each team will be stored as a structure with the following attributes:

- Team code (eg. 0, 1, 2, 3, 4, etc.)
  - Each team code must be unique. Only integers 0-31 are acceptable
- Team name (eg. "Australia", "Cameroon", "Canada", "Argentina", "Belgium", etc.)
  - Only team names up to 25 characters in length are acceptable (including the null character - So think of this as 24+1). Each team name must be unique
- Group seeding (eg. "A1", "B3", "F2", etc.)
  - Only groups A-H and seeds 1-4 are acceptable. (So only A1, A2, A3, A4, B1, B2, ..., H3, H4). Each group seeding must be unique
- Primary kit (uniform) colours (eg. "Red", "Orange", "Yellow", "Green", "Blue", "Indigo", and "Violet")
  - Only the values 'R', 'O', 'Y', 'G', 'B', 'I', 'V' are acceptable.

Your program will then use an array of structures to represent up to 32 possible teams.

## Implementation

Your program should continuously prompt the user for one of five possible commands:

1. Insert a new team (using command `i`)
  - Prompt the user for the team code
    - Assume the user will enter one integer
    - This must be unique in your database and cannot conflict with an existing team code. If the number is less than 0 or greater than 31, or if there is a conflict with an existing code, or if the database is full, tell the user the error. The user can try again or you can return the user to the main prompt
  - Prompt the user for the name of the team
    - Assume the user will enter a string of characters of any length
    - If the team name is longer than the acceptable length, you should accept as many characters as you can and ignore any additional characters. If there is any other issue, tell the user the error. The user can try again or you can return the user to the main prompt
  - Prompt the user for the group seeding of the team
    - Assume the user will enter two characters: a letter representing the group, and a number representing the seeding
    - If the letter is not A-H or if the number is not 1-4, tell the user the error. The user can try again or you can return the user to the main prompt
  - Prompt the user for the area of the primary kit (uniform) colour
    - Assume the user will enter one character value

- If the character is not in the list 'R', 'O', 'Y', 'G', 'B', 'I', 'V', tell the user the error. The user can try again or you can return the user to the main prompt
- 2. Search for a team in the database and print it out (using command `s`)
  - Prompt the user for the team code
    - If the team code is found, print out all the values for this team only (see the print command below for more details)
    - If the team code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
- 3. Update a team in the database (using command `u`)
  - Prompt the user for the team code
    - If the team code is found, prompt the user to update all the values for the team (see the insert command above for details)
    - If the team code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
- 4. Print the entire list of teams (using command `p`)
  - Print out a table listing all the teams in your database with all the attributes:
    - Team Code
    - Team Name
    - Group Seeding
    - Primary Kit Colour
- 5. Quit the program (using command `q`)
  - Yes, all data is lost when quitting your program. You do not need to maintain the data across multiple runs.

## PART 2

As in part 1, each election team will be stored as a structure with the following attributes:

- Team code (eg. 0, 1, 2, 3, 4, etc.)
  - Each team code must be unique. Only integers 0-31 are acceptable
- Team name (eg. "Australia", "Cameroon", "Canada", "Argentina", "Belgium", etc.)
  - Only team names up to 25 characters in length are acceptable (including the null character - So think of this as 24+1). Each team name must be unique
- Group seeding (eg. "A1", "B3", "F2", etc.)
  - Only groups A-H and seeds 1-4 are acceptable. (So only A1, A2, A3, A4, B1, B2, ..., H3, H4). Each group seeding must be unique
- Primary kit (uniform) colours (eg. "Red", "Orange", "Yellow", "Green", "Blue", "Indigo", and "Violet")
- Only the values 'R', 'O', 'Y', 'G', 'B', 'I', 'V' are acceptable..

Now, you will *also* need to store each team's player as a structure with the following attributes:

- Player code (eg. 0, 1, 2, 3, 4, etc.)

- Each player code must be unique. There is no limit to the number of players but each code will be greater than or equal to 0
- Player name (eg. "John Smith", "Jane Doe", etc.)
  - Only names up to 50 characters in length are acceptable (including the null character - So think of this as 49+1)
- Player age (eg. 17, 18, 19, 20, etc.)
  - Only integers 17-99 are acceptable
- Professional club affiliation (eg. "Manchester City", "Los Angeles FC", "Toronto FC", "AFC Ajax", etc.)
  - Only names up to 50 characters in length are acceptable (including the null character - So think of this as 49+1)

Your program will then use a linked list of structures to represent all teams and a linked list of structures to represent all players.

## Implementation

Your program should continuously prompt the user for one of four possible commands:

1. Print help (using command `h`)
  - Print a simple message or messages describing how to use the program.
2. Quit (using command `q`)
  - Quit the program. Yes, all data is lost when quitting your program. You do not need to maintain the data across multiple runs.
3. Control teams (using command `t`)
  - See below
4. Control players (using command `p`)
  - See below

## Teams

As in worldCupDB.c (Part 1) all the same commands exist as before:

1. Insert a new team (using command `i`)
  - Prompt the user for the team code
    - Assume the user will enter one integer
    - This must be unique in your database and cannot conflict with an existing team code. If there is a conflict with an existing code, or if the database is full (you cannot allocate any more memory), tell the user the error. The user can try again or you can return the user to the main prompt
  - Prompt the user for the name of the team
    - Assume the user will enter a string of characters of any length
    - If the team name is longer than the acceptable length, you should accept as many characters as you can and ignore any additional characters. If there is any other issue, tell the user the error. The user can try again or you can return the user to the main prompt

- Prompt the user for the group seeding of the team
  - Assume the user will enter two characters: a letter representing the group, and a number representing the seeding
  - If the letter is not A-H or if the number is not 1-4, tell the user the error. The user can try again or you can return the user to the main prompt
- Prompt the user for the area of the primary kit (uniform) colour
  - Assume the user will enter one character value
  - If the character is not in the list "R", "O", "Y", "G", "B", "I", or "V", tell the user the error. The user can try again or you can return the user to the main prompt
- 2. Search for a team in the database and print it out (using command `s`)
  - Prompt the user for the team code
    - If the team code is found, print out all the values for this team only (see the print command below for more details)
    - If the team code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
- 3. Update a team in the database (using command `u`)
  - Prompt the user for the team code
    - If the team code is found, prompt the user to update all the values for the team (see the insert command above for details)
    - If the team code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
- 4. Print the entire list of teams (using command `p`)
  - Print out a table listing all the teams in your database with all the attributes:
    - Team Code
    - Team Name
    - Group Seeding
    - Primary Kit Colour

You should implement one additional command:

1. Delete team (using command `d`)
  - Prompt the user for the team code
    - If the team code is found, delete the team safely by removing the element from the linked list and `freeing` the memory
    - If the team code is not found, tell the user the error. The user can try again or you can return the user to the main prompt

## Players

Implement the following commands:

1. Insert a new player (using command `i`)
  - Prompt the user for the player code
    - Assume the user will enter one integer
    - This must be unique in your database and cannot conflict with an existing player code. If there is a conflict with an existing code, or if the database is

full (you cannot allocate any more memory), tell the user the error. The user can try again or you can return the user to the main prompt

- Prompt the user for the name of the player
    - Assume the user will enter a string of characters of any length
    - If the player name is longer than the acceptable length, you should accept as many characters as you can and ignore any additional characters. If there is any other issue, tell the user the error. The user can try again or you can return the user to the main prompt
  - Prompt the user for the age of the player
    - Assume the user will enter one integer
    - If the number is less than 17 or greater than 99, tell the user the error. The user can try again or you can return the user to the main prompt
  - Prompt the user for the professional club affiliation of the player
    - Assume the user will enter a string of characters of any length
    - If the professional club affiliation name is longer than the acceptable length, you should accept as many characters as you can and ignore any additional characters. If there is any other issue, tell the user the error. The user can try again or you can return the user to the main prompt
2. Search for a player in the database and print it out (using command `s`)
- Prompt the user for the player code
    - If the player code is found, print out all the values for this player only (see the print command below for more details)
    - If the player code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
3. Update a player in the database (using command `u`)
- Prompt the user for the player code
    - If the player code is found, prompt the user to update all the values for the player (see the insert command above for details)
    - If the player code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
4. Print the entire list of players (using command `p`)
- Print out a table listing all the players in your database with all the attributes:
    - Player Code
    - Player Name
    - Age
    - Club
5. Delete player (using command `d`)
- Prompt the user for the player code
    - If the player code is found, erase the player safely by removing the element from the linked list and `freeing` the memory
    - If the player code is not found, tell the user the error. The user can try again or you can return the user to the main prompt
6. Include a fifth member of the player structure called `team_code` and use the insert function to prompt the user for this data. Include a sixth command to the team menu called `r` to print all the players registered on the team the user chooses. For example, if player 23 is

registered in team code 12, then I know that player 23 is a player for team 12. Therefore, if I print the registrants of team 12, player 23 should show in the list. Note the following rules:

- The team code entered by the user must already exist as a valid team in your database. In other words, a player cannot be a member of a team which doesn't exist yet.
- It is not possible to remove a team that still has players registered for it. One must either re-assign or remove all players first, then the team could be removed.

## Other implementation notes

- You are welcome to create any number of helper functions you wish
- You are welcome to use any C libraries you wish but you should be able to get by with `stdio.h`, `stdlib.h`, and `string.h`
- You are welcome to use whatever wording you would like for your prompts. It does not have to precisely match the example below.