Coding Project 2 (CP2):
Extending OO design, using dynamic memory

CP2 will continue to store **`Team`** objects in a single league (which may or may not be an instance of a **`League`** C++ class), and **`Player`** objects within each team. So the core structure of a main program **`.cpp`** file that implements a menu for user input, along with classes for **`Team`**, **`Player`**, and (optionally) **`League`**, should remain. Therefore, you may begin CP2 with your working version of CP1, with a working version of CP1 that I will provide for you, or from scratch. You may begin CP2 with some other student's working CP1 code.

CP2 will require additions *and changes* to whatever code you choose to begin with. How users interact with the program through the top-level menu will change, and new options must be supported. The menu commands and their usage are specified below, and a sample executable will "document" the required behavior by example.

In addition to the functionality changes, there is one important implementation constraint:

> For full credit on CP2 *you may not use* static arrays, vectors, or any other STL containers. Everything should be stored in dynamically allocated storage that your code manages, without any hard limits on any container sizes, including player rosters and the league's teams. In other words, *you should use dynamic arrays* (or linked lists, if you are comfortable enough to do so before we have covered them fully). Your dynamic array should mimic the functionality of the vector<> container in the sense that it should automatically grow if necessary.

Menu Changes

The menu will change in two or three ways (depending on whether you implement the optional part of the assignment):

1. Existing menu options will read more information on the line with the command, rather than on separate lines. For example, instead of typing `Player` and then being prompted to input the name(s), jersey number, and team on separate lines, the user of your program will now enter everything on one line.

2. There will be additional menu options to support added functionality, as described below.

3. [OPTIONAL] There will be a sub-menu for entering information about trades.

New Functionality

CP2 should support free agents, releasing and signing players, and unique jersey numbers.

- A free agent is a player who is not currently on a team.

- Releasing a player removes them from a team roster and puts them in a list of free agents
- Signing a player moves them from the list of free agents onto a team roster
- CP2 should be smarter about jersey numbers. Two players on the same team may not have the same jersey number. When necessary, your program should assign jersey numbers automatically, and should remember players' preferred numbers, as described below. Your program should also remember all of each players' teams and associated jersey numbers across team changes.
- [optional] A trade moves some number of players between teams.

Top Level Menu

At the top level, a user may enter the commands as described below. The **[**, **]**, **<**, and **>** characters are never intended to be typed by the user. They delineate the fields of input, with `[<sample>]` indicating an optional parameter, because it appears within brackets.

`Quit`
Behaves exactly as CP1.

`Team <location> <team-nickname>`
Behaves exactly as CP1, except that the team information is specified on one line. Checks to ensure that `<team-nickname>` is not already in use by some other team; if not, a new team is added to the league. Locations and nicknames may not contain whitespace.

Example:
`Team Phoenix Suns`
Team Phoenix Suns added to the league

`Player <firstname> <lastname> <number> [<team-nickname>]`
Behaves exactly as CP1, except that the player information is specified on one line. (Let's stop using "active" in CP2, for simplicity.) The `<team-nickname>` is an optional parameter. If no team is specified, or if the specified team does not exist, the player should be added to the list of free agents. If a team is specified, the player should be added to that team. If the specified jersey number is taken, your program should assign the lowest available positive jersey number to the player, but should remember that the player's preferred number is `<number>`.[1]

[1] If the player is later (i) released and signed by, or (ii) traded to, another team, and the player's preferred number is available on that team, then the player should be assigned that number. Otherwise, the player should retain whichever number they had on their previous team.

Examples:
`Player LeBron James 23`
Player LeBron James added to free agents

`Player Chris Paul 3 Suns`
Player Chris Paul added to Suns

`League`
Same functionality as CP1, except that the number of players on each team should be displayed, along with the team name.

Example:
`League`
Boston Celtics (4 players)
Phoenix Suns (2 players)
Miami Heat (0 players)

`Roster [<team-nickname>]`
Behaves as CP1, if a team nickname is specified. If `<team-nickname>` is not specified, your program should print the list of free agents. If `<team-nickname>` does not match an existing team, your program should report an error and should not show any players.

Examples:
`Roster Suns`
Phoenix Suns
  Players:
Paul, Chris (#3)
Booker, Devin (#1)

`Release <lastname> <team-nickname>`
Removes the specified player from the specified team, and places the player onto the list of free agents. You may assume that the last name uniquely identifies a player within the league. In other words, all inputs that we will test will give all players different last names. Your program should check that the specified team exists, and that the specified player is currently on that team, and report an error if appropriate.

Example:
`Release Paul Suns`
Player Chris Paul released from the Phoenix Suns, added to free agents

---

Sign <lastname> <team-nickname>

Adds the specified player from the list of free agents to the specified team. If the player is not a free agent, or the team does not exist, then your program should report an error. The player should be given their preferred jersey number on their new team, if possible. If not, the player should be given their most recent jersey number, if they were on a team previously. Otherwise, the player should be given the lowest jersey number that is not currently in use on team `<team-nickname>`.

Example:

```
Sign Paul Celtics
Player Chris Paul added from free agents to the Boston Celtics
```

Career <lastname>

Should show a list of teams that the specified player has played for, including their jersey numbers. Note that your program should look through the list of free agents and all of the teams, to find the player. Note also that you will have to store with a player the history of their jersey numbers and teams.

Example:

```
Career Smith
Celtics (#4)
Heat (#6)
Mavericks (#4)
Celtics (#2)
```

Trade **(OPTIONAL)**

Should read one or more `Move` lines that follow, until another `Trade` command appears. Only if all `Move` commands are legal (that is, none of them cause an error) should any of the `Move` commands be enacted. In other words, any `Move` that is not legal should cause your program to ignore all of the `Moves` associated with the trade.

Move <lastname> <old-team-nickname> <new-team-nickname>

Moves the specified player from one team to another. Checks to ensure that both teams exist and that the specified player is on the first specified team.

Example:

```
Trade
Beginning trade
Move Harden Nets 76ers
Move Simmons 76ers Nets
Trade
Trade completed successfully
```

```
Trade
```
Beginning trade
```
Move James Nets Knicks
```
Error: No player James on the Knicks
```
Move Singh Fakers
```
Error: Team Fakers does not exist
```
Move Barrett Knicks Lakers
Trade
```
Trade unsuccessful

Your program should report all errors associated with the trade. A trade does not have to “make sense” in any way other than the teams should exist and the players should be on the specified teams. In other words, a legal trade could comprise a series of moves among several teams, with some teams not getting any players, or it could be one or more Moves with all players going from one team to another, etc.