Initial design

Data Structures Required

To hold player information, a struct to hold pertinent information:

```
typedef struct player_t {
    char *id;
    char *name;
    char *position
    char *college
    llist_t *teams; // Pointer to linked-list of teams they have played for
} player_t;
```

Players will be stored in a hash-table for constant time look up for --player and --search options

```
typedef struct hash_t {
    player_t **players; // Entries in the hashtable are of type player_t
    hash_f hash_function;
    uint32_t max_cap;
    uint32_t curr_size;
}hash_t;
hash_t player_table;
```

To hold team information:

```
typedef struct team_t {
    char *team_name;
    int year;
    llist_t roster;  // Pointer to linked-list of players on that specific
team
} team_t;
```

Teams will also be stored in a hashtable for constant time look up for the --roster option

```
struct hash_t {
    team_t **teams;
    hash_f hash_function;
    uint32_t max_cap;
    uint32_t curr_size;
}hash_t;
hash_t *team_table;
```

Major Functions Pseudocode

--player

```
For player in player_table:
    if player name or player id equals optarg:
        print player id, player name, player position
        iter linked-list of player teams:
            print year and team
```

--search

```
for each item in player_table:
    if item exists:
        if strncmp of player name or id match optarg:
            print player id, name and college
```

--stats

Breadth First Search

```
Set total_score to 0
get index from hashed optarg
starting vertex set to team_table at index
BFS
At each level of BFS:
    increment count if valid
    print level and count
    total_score += (count * level)
    incremement level
    reset count to 0

print the total score divided by total players found
```

--distance

Breadth First Search w/ Early Exit

```
Conduct normal breadth first search with starting index of optarg 1, and
end (early exit condition)
optarg 2.

3 print statements based on start of search, end, or middle node.
if start node:
    print statement a
```

```
else if end node:
    print statement b
else:
    print statement c
```

--teams

```
Add team_table[0] to a queue
For each team in team_table:
    if team_name not in queue:
        Add team to queue

while the queue is not empty:
    dequeue team
    print team name
```

--roster

```
get index by hashing team + year
go to team_table at index
if exists:
    iterate over team_table players list
    print player
```

Project Flow

1. Validate File argument
2. Validate options (should be 1) as well as the optional arguments
3. Create hashtables
4. Parse the file and populate hashtables
5. Based on command option, call option-specific print function
6. Clean memory and close file
7. Exit