CS3610 Project 4

Due: Monday March 16 - 11:59pm

Your employer has asked you to keep track of the best basketball players in the league. Players are ranked by the number of points they have scored. Your employer wants updates on the list every week so you need to keep your rankings organized at all times. Fortunately, the max heap data structure can help you perform this task with ease.

You typically only need to add players to your heap at the beginning of a season. This means you need to spend O(n) time adding the n players of the league to your heap. Of course, new players could come in mid-season, but this should not be a problem with the add function you design. Players may also leave a team, which is why you will also define a remove function.

The difficult task is finding a way to keep the top players organized while continually updating each of the n players' points after each game. Fortunately, a heap makes this task easy and efficient. After a player is found in the list, it takes at most $O(lg(n))$ time to update a player's score and maintain the heap structure. This is preferable to using $O(n)$ to update and maintain a player's score with a sorted list.

When asked to print the top player, it should just take constant time to read from the top of your heap. When asked to print all the players by rank from greatest to least, you can use heap sort in $O(nlg(n))$ time.

Input Format:

The input has been optimized to read from a file. A data file has been provided with the starter code. The format is as follows:

'a' for adding a player followed by a name 'Player-X' followed by an integer value 'i' for number of points scored.

'r' for removing a player followed by a name 'Player-X'

'u' for updating a player score followed by a name 'Player-X' followed by an integer value 'i' for number of points to add to current score.

'o' for ordering and printing the players of the league by rank from greatest to least.

't' for printing the top player of the league

'q' to quit the program.

Output Format:

If adding an already entered player to the heap, print the following message:

```
Player already in heap
```

If attempting to remove or update a player not in the heap, print the following message:

```
Player not found
```

For printing the top player or a list of all players, give the name of the player followed by the number of points scored:

```
  Player-X: 100
```

Examples:

To begin testing, type the commands from the command line or save a sequence of commands in a test file (see input1.dat as an example).

```
$ ./a.out < input1.dat
```

Input 1:

```
a
Michael
50
a
John
30
a
Ted
48
t
u
Joe
3
u
Ted
3
t
o
q
```

Output 1:

```
Michael: 50
Player not found
Ted: 51
Ted: 51
Michael: 50
John: 30
```

Turn In:

Email your source code to pg219709@ohio.edu with the subject ”CS3610 Project 4”. If you have multiple files, package them into a compressed tar file (.tar.gz).

As an example, if I had my source files in a directory named ”project4/”, I could package the files inside with the following command

```
$ tar -zcvf cs3610_project4.tar.gz project4/
```

Now my files reside in cs3610_project4.tar.gz, which you will email to me. To verify that the files were packaged correctly, you can unpackage the .tar.gz file with the following command.

```
$ tar -xvf cs3610_project4.tar.gz
```