Project 1

Due June 19

1 Description

For purposes of this project, a map is a n (row) by m (column) grid where each grid square is one of the following types of terrain:

- road (‘r’) - cost: 1
- field (‘f’) - cost: 2
- hills (‘h’) - cost: 5
- mountain (‘m’) - cost: 10
- water (‘w’) - impossible to navigate; you can’t swim

Map files are plain text files. A very simple example of a map file (an ‘L’ shaped island in the middle of water, with a hill on the north end):

```
wwwwww
whwwww
wfwwww
wffffw
wwwwww
```

Possible movement directions are up, down, left and right (‘u’, ‘d’, ‘l’, ’r’). Moving off the edge of the map is impossible, e.g. you cannot move up if you are in the top row. The cost of movement is ‘paid’ when *entering* a square. For example, starting in square $(1, 1)$ (the hill on the north end of the island; maps are 0-indexed), the following sequence of moves:

```
["d", "d", "r", "r", "r"]
```

Will move you to square $(3, 4)$, with a total cost of 10.

Your assignment is to write a program that, given a map file, initial row and column, and goal row and column, produces a *least cost* sequence of moves to the goal *if possible*. You must provide a file `solution` that, when executed on the command line with the following arguments:

```
$ ./solution map.txt initial_row initial_col goal_row goal_col
```

Prints a result to standard output that is either a JSON array that is a minimal-cost sequence of moves to the goal or the JSON value `null` if no such sequence is possible.

Submit your `solution` file (and any other files needed for your code to work) to the submit server (`submit.cs.umd.edu`)

2 Other details

- **Start early. Do not put this project off until the last minute.**
- All code submitted must be your own. You may use standard library functions/modules.
- We reserve the right to test your code on maps several thousand by several thousand grid squares. Whatever approach you take must be efficient enough to succeed within a few seconds of run time.
- Test your code! It's not hard to find a path to the goal - finding an optimal path is more difficult.
- Your solution file must be executable; if you're using a scripting language (Python is recommended, but not required), the first line should be an appropriate shebang (`https://en.wikipedia.org/wiki/Shebang_(Unix)`), e.g.

  ```
  #!/usr/bin/env python3
  ```

- If you're using Java (not recommended, but acceptable), then 'wrap' your jar so we can execute it with `$ ./solution map.txt ...` For example, your `solution` file could be a simple shell script:

  ```
  #!/usr/bin/env bash
  java -jar yourJarFile.jar $@
  ```

- The following programming languages are ‘officially sanctioned’. Talk to us before using anything different (remember, we do need to be able to execute your code on our machines):
  - Java
  - Python 2 or 3 (make sure you use the correct shebang)
  - Ruby
  - OCaml (make sure `$ ./solution ...` works)
  - Bash (to ‘wrap’ your code files, if necessary)