CMPT 225 Assignment 5 (3%)

This is an individual endeavour. Submit your solutions by Wednesday, April 3, 2019 3:00pm.

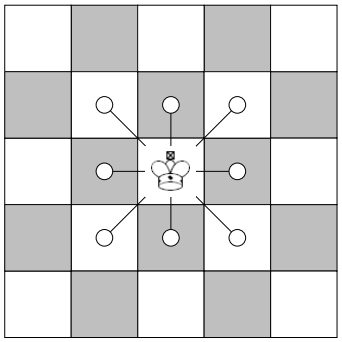
As usual, you should refer to the basecode and the sample input/output to resolve any clarifications about your code's usage/format.

0. [0 marks] *The Goal*

   This last Assignment is meant to be a quickie. The goal is to get some exposure to **applying** the Standard Template Library data collections to solve problems. This is a departure from past Assignments where your focus was on their **implementation**.

1. [60 marks] *Chess Puzzles*

   In the game of chess, all moves take place on a grid of squares. In one turn, the king may move one square horizontally, vertically or diagonally, up to 8 possibilities in all, but may not move to a square occupied by another piece, or off the board.

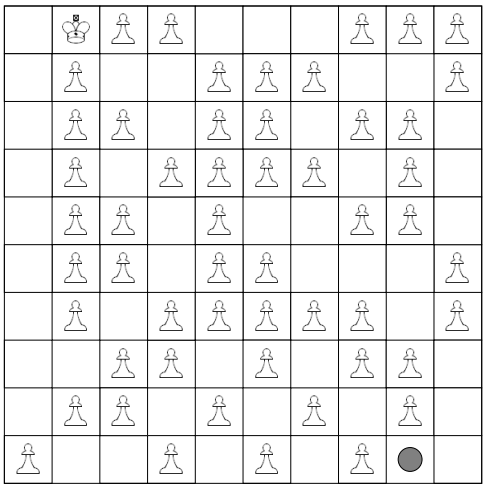
   

   Consider the challenge to move the king from its starting square to a goal square, the challenge being to do it in the fewest moves possible. The puzzle illustrated here can be solved in 14 moves. (You probably should verify this.)

   You will write a program that solves these sorts of puzzles.

   (a) [0 marks] Run `make` to build the base code. Then run it using "`./king < puzzle.in`".

   The file "`puzzle.in`" contains a sample board which you should use as a template to create your own test boards. It is reproduced below for convenience:

```
10
 k##   ###
 #  ###  #
 ## ## ##
 # #### #
 ## #  ##
 ## ##   #
 # ##### #
  ## # ##
 ## # # #
#  # # #o
```

The first line of the file is an integer n, which denotes that the puzzle occurs on an $n \times n$ board. (In general, n could be very large, perhaps on the order of 10^5.) Next come n lines of n characters each where:

- "`k`" represents the starting position of the king;
- "`o`" represents the goal position;
- "`#`" represents an obstacle; and
- " " represents an empty square.

(b) [20 marks] Complete `king.cpp` so that it correctly reports the minimum number of moves required to move the king from the starting position to the goal. The class `ChessMap` has been provided for you to organize the map and simplify the input routine. To access the `j`th character of row `i`, use either `map[i][j]` or `map[{i,j}]`.

The output? Simply output the number of moves, as an integer, on a line by itself. For example,

```
14
```

If the puzzle cannot be completed, your program should output

```
Impossible
```

Reminder: The point of this question is not that you should *implement* a queue ADT — instead you should *apply* an ADT to this problem. (And you certainly need a Queue ADT to implement a Breadth First Search.)

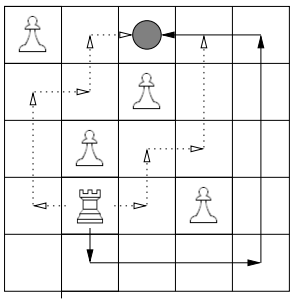
- You should use the C++ STL generic `queue`, whose interface can be found here: `http://www.cplusplus.com/reference/stl/queue/`
- You'll need to look up STL generic `pair` as well.
- You'll also need one of `vector` or `unordered_map`.

(c) [20 marks] Once your code for `king.cpp` is working, copy it to a new file named `king-moves.cpp`. Enhance your code in `king-moves.cpp` so that it prints out a sequence of moves that lead to an optimal solution. For example,

```
1, 2
1, 3
0, 4
0, 5
0, 6
1, 7
2, 6
3, 7
4, 6
5, 7
6, 8
7, 9
8, 9
9, 8
```

If there are multiple optimal solutions, output any one of them.

(d) [20 marks] Once again, copy your code from `king-moves.cpp` to `rook-moves.cpp`. Repeat the previous question, but using a rook instead of a king. A rook may move 1 or more squares in the horizontal of vertical direction, but may not jump over any obstacles or off the board.



Notes: As usual, your grade will be based on a blend of:

- **Correctness** — Did your program [efficiently] solve the stated problem?
- **Requirements** — Did your program make [good] use of the STL?
- **Style** — Did your code meet the coding style requirements?

One Last Reminder: Follow the instructions carefully. Changing `mapio.h` or `mapio.cpp` may result in a compile error when **we** try to build your code using **our** version of `mapio.h`/`mapio.cpp`. Code that doesn't build never receives a mark much higher than 0.