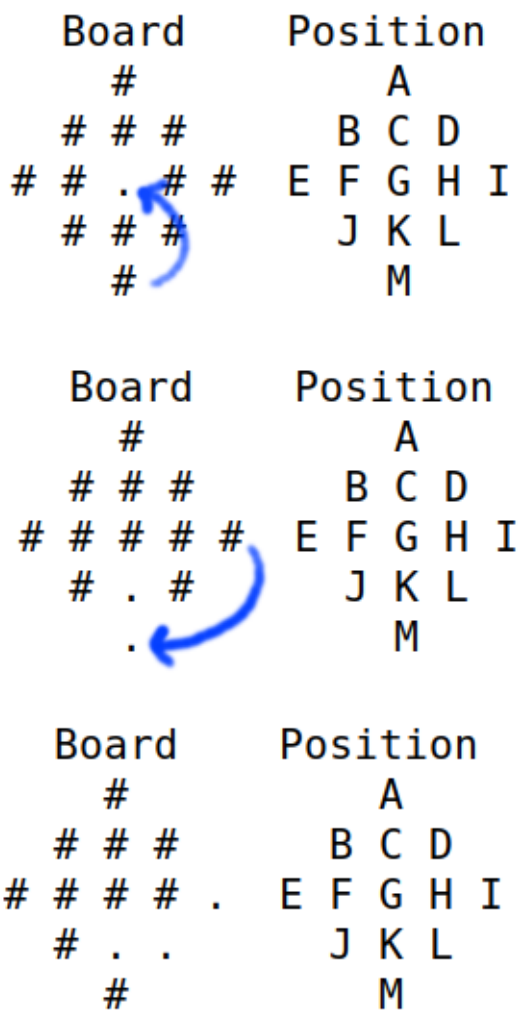
The Great 13 Puzzle

1 Introduction

You will be implementing *The Great 13 Puzzle*. This is a one player game, where the goal is to clear the peg board. Pegs are only removed if they are "hopped" by another peg on the board. The example on the right illustrates this in action. The board is a diamond shape, with pegs represented by the symbol #. Empty spaces are indicated by "." The positions on the board are labelled on the right.



2 Rules of the Game

- A peg may be removed if it is *jumped*. The diagram on the right shows the removal of the K peg by *jumping* the M peg over it.
- Valid jumps are done with pegs that "hop" over another peg, and land in an empty space. That is allowed to be vertical, horizontal, or diagonal. You cannot jump a peg on a curved path. For example, the next jump is the I peg, over the L peg, and into the empty space in M. We cannot jump the I peg over the H peg, and land in the K space.
- The game is won when there is only one peg remaining on the board.

3 How Your Program will work

Below is an overview of the expected functionality of your program.

1. Tell the user how to play the game, by calling `void display_instructions`.
2. Display the board, in the starting state, along with the legend on the right.
3. Prompt the user to enter a command, that moves a peg, as a 3 character sequence.
4. If the command is not valid, display the appropriate message and go to Step 3.
5. If the command was X, then exit the game, or if the command was R, then go to Step 2.
6. Display the board, after the user's command has executed.
7. If the game is over, inform the user, then quit. Otherwise, go to Step 3.

4 How the User Will Give Input

The user will give a sequence of *3* characters, for example *mkg* to indicate which peg will move (M), which peg is jumped (K), and where the peg will land (G). Note that *the user is allowed to give upper case or lower case letters* so MKG, mkg, mKg, MkG, ... etc are referring to the same sequence.

5 Starting Materials

There are three things you can find on Blackboard, but we wont release the starter code immediately.

1. The starter code called *program2.cc*, which is what you will modify to create your solution.
2. This pdf, *program2.pdf*.
3. A zip file called *examples.zip*. It provides examples of the expected output, when the game is played.

6 Other Important Printed Messages to the User

When you prompt the user for input, display the following message.

```
Enter positions from, jump, and to (e.g. EFG):
```

When the user gives the character 'X' (or 'x') as their input, you will print the following message and exit.

```
Exiting.
```

When the user gives the character 'R' (or 'r') as their input, you will print the following message.

```
Restarting game.
```

When the user wins the game, you will print the following message and then exit.

```
Congrats you win!
```

7 Messages on Invalid Commands

Note: In the cases where the user's command will generate more than one error, only print out the error with top priority. The examples below are listed from top priority to bottom.

When the command does not have 3 characters, and is not 'X' or 'R', print the following.

```
*** Invalid sequence. Please retry.
```

When a command has positions not in range A $\rightarrow$ M, print the following message.

```
*** Given move has positions not on the board! Please retry.
```

When the command has a peg jump over an empty space, print the following message.

```
*** Must jump a piece. Please retry.
```

When the command uses an empty space as the jumping peg, print the following message.

```
*** Source needs a piece. Please retry.
```

When the user enters a command that would jump the peg into a position, that is already occupied, then print the following message.

```
*** Destination must be empty. Please retry.
```

When the command refers to a sequence of positions that is not valid (for example IHK), print the following.

```
*** Move is invalid. Please retry.
```

8 How to Submit

You can take the starter code *starter.cc*, and rename it, but make sure all your work is in one file. Note that the autograder might fail if you have spaces or parenthesis in the name of your file. You will submit this to **Gradescope** under the assignment called *Program 2*. We will have an autograder help us handle the large number of submissions, and to give you quick feedback on your score. Note that at the time of release, the autograder might not be ready.