COSC 2436 Homework #3

Overview:
You are asked to Create a C++ program that explores the classic computer science problem of finding a **Knight's Tour** of a given board. In the game of chess, the knight piece makes a special L-shaped move, as noted below. A knight's tour is a path through an NxM board where the knight visits every square exactly once with no repeats. A closed tour is one where the knight returns to its starting square forming a continuous loop through the board.

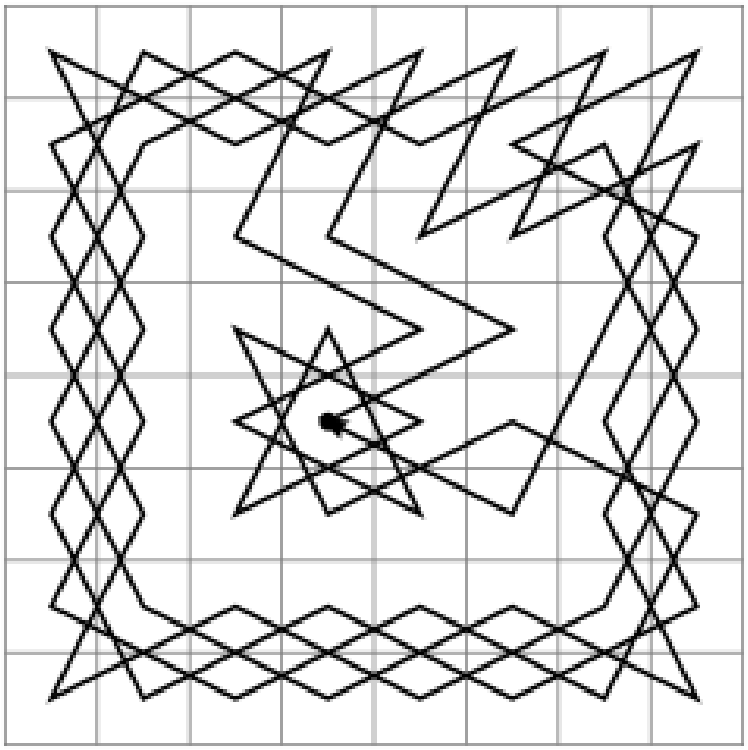

Figure 1 - Example closed Knight's tour on an 8x8 board.

We are interested in using backtracking to find closed tours on special boards. A stack will provide the mechanism for attempting to solve this problem and allow for the backtracking as needed. We will also need to queue the possible moves for the knight, with the priority in the order shown below.
In addition, we will consider the possibility of having holes in the board that are not considered as part of the solution. These holes should not be visited by the knight.

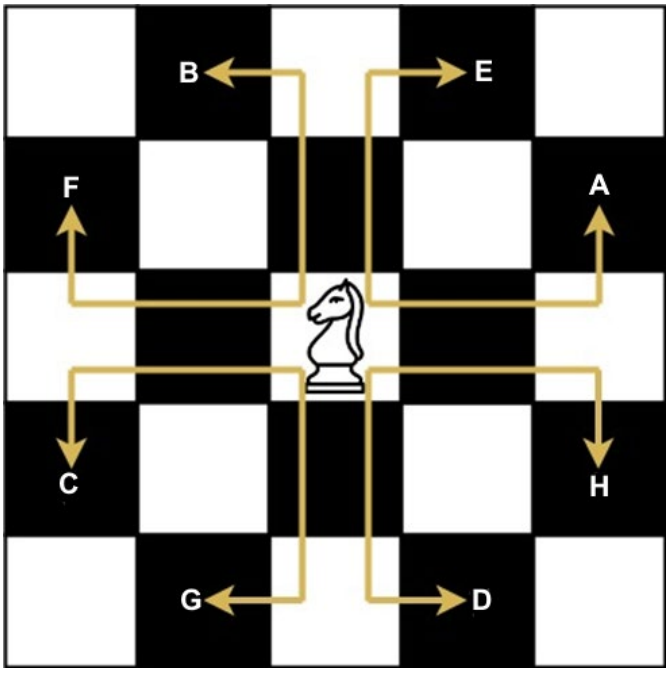

Figure 2 - Knight's moves with order of precedence indicated.

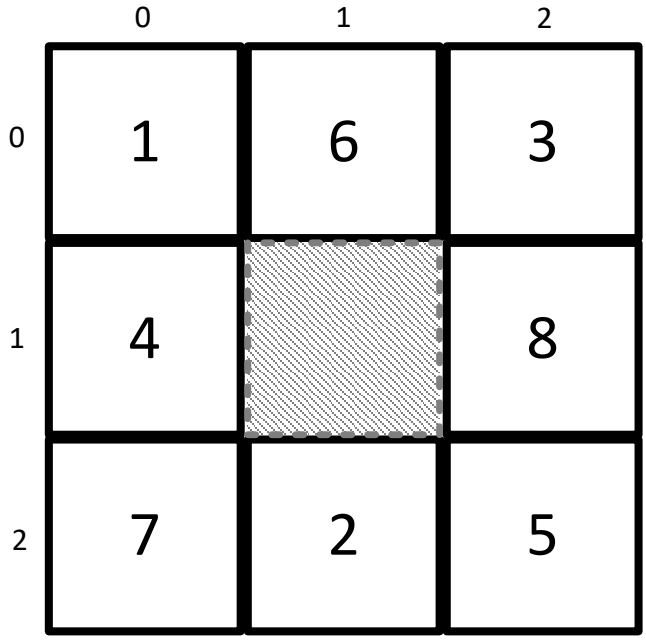

Figure 3 - Closed tour through a 3x3 grid with a hole at (1,1) starting from (0,0).

Input files
- The input files contain several lines indicating the details of a particular board. All input files are well formed with valid values. There is no need for exception handling on the inputs and all files follow the same format.
- The first line provides the size of the board, as integers, separated by a single space. These values are the number of rows and number of columns respectively and will always be positive integers.
- The second line provides, the starting location (row and column) as integers, separated by a single space. These values are always valid for the given board and are used for the start of numbering.
- The third line provides, the number if holes or excluded squares on the board as a single non-negative integer. This value could be zero and is always a reasonable value for the given board.
- The next lines in the file contain the locations of any holes/excluded tiles. One pair of integers (row and column) is provided on each line, with the values on each line separated by a single space. The number of lines corresponds to the number of holes listed on the third line in the file. These number pairs are always valid for the board and do not repeat.

For example, the **input1.txt** contains:

```
3 3
0 0
1
1 1
```

This corresponds to a board with 3 rows and 3 columns. The starting point for any closed tour should be row 0 and column 0. It includes one (1) hole at location (1,1) in he board. This corresponds to the image presented above. This board does contain a closed tour, as shown.

Output files
- If the requested board contains a closed tour, output the message, “A closed tour exists!” If no tour exists for the given board configuration, output the message, “A closed tour does not exist!”
- If a closed tour exists, output on the next line, the board with the tour indicated . The order of the cell-visits should be indicated with a number, taking at most 2 spaces. Any holes in the board should be marked with two “X” characters and may not be visited.

For example, the **ans1.txt** contains:

```
A closed tour exists!
+--+--+--+
| 1| 6| 3|
+--+--+--+
| 4|XX| 8|
+--+--+--+
| 7| 2| 5|
+--+--+--+
```

Note that the order of visiting the cells is important. Be sure to use the order listed above as the preference for visiting cells on the tour.

Reminder

- Turn in your lab assignment to our Linux server.
- Make sure to only have one (1) .cpp file with the main() function in your working directory, otherwise your program will fail the grading script.
- Create a folder under your root directory, name the folder hw3 (case sensitive), copy all your .cpp and .h files to the folder (ArgumentManager.h is also needed)
- Only include the necessary files (.cpp and .h files) in your working directory in your final submission.
- To test your program, you may wish to copy the input files and answer files onto the server and run your program. Do not include any outputs files and after verifying that the code passes the tests, **delete any output*.txt files**.

Please reach out to myself or the TAs for any clarifications or typos.