CSCI 310 – Data Structures – Spring 2019
HW 01 – Knight's Tour (20 points)

One of the more interesting puzzlers for chess buffs is the *Knight's Tour* problem, originally proposed by the mathematician Leonhard Euler. The question is this: Can the chess piece called the *knight* move around an empty chessboard and touch each of the 64 squares once and only once?

The knight makes L-shaped moves (over two in one direction and then over one in a perpendicular direction). Thus, from a square in the middle of an empty chessboard, the knight can make eight different moves (numbered 0 through 7) as shown in the Figure 1.

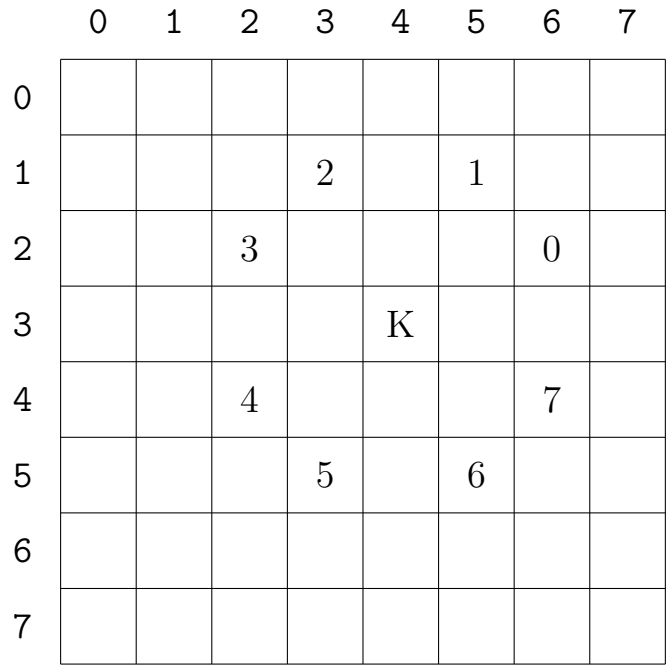

Figure 1: These are the legal moves for a knight on square $(3, 4)$.

1. (3 points) Draw an 8×8 chessboard on a sheet of paper and attempt a Knight's Tour by hand. Put a 1 in the first square you move to, a 2 in the second square, a 3 in the third, etc. Before starting the tour, estimate how far you think you will get, remembering that a full tour consists of 64 moves. How far did you get? Was this close to your estimate?

2. (7 points) Now let's develop a program that will move the knight around a chessboard. The board is represented by an 8×8 two-dimensional array called `board`. Each of the squares is initialized to zero. We describe each of the eight possible moves in terms of both their horizontal and vertical components. For example, a move of type 0, as shown in Figure 1, consists of moving two squares horizontally to the right and one square vertically upward. A type 2 move consists of moving one square horizontally to the left and two squares vertically upward. Horizontal moves to the left and vertical moves upward are indicated with negative numbers. The eight types of moves can then be described using two single-subscripted arrays, named `horizontal` and and `vertical`, as shown in Figure 2.

   Let the variables `currentColumn` and `currentRow` indicate the column and row of the knight's current position. To make a move of type `moveNumber`, where `moveNumber` is from 0 and 7, your program may use statements such as

```
nextColumn = currentColumn + horizontal[ moveNumber ];
nextRow = currentRow + vertical[ moveNumber ];
```

Move Type	0	1	2	3	4	5	6	7
horizontal	2	1	-1	-2	-2	-1	1	2
vertical	-1	-2	-2	-1	1	2	2	1

Figure 2: There are eight different types of moves.

Use a counter that runs from 1 to 64 to keep track of how many moves the tour was able to make. Record the latest count in each square the knight moves to. Test each potential move to see if (i) the knight already visited that square, and (ii) to make sure the knight does not move *off* the chessboard. Write a program that attempts the Knight's Tour by moving the knight around the chessboard. The program should show the move number on each square where the knight landed. Run your program. How many moves did the knight make? A sample run from one possible solution to Question 2 is shown in Figure 3.

```
 1  10  23  42   7   4  13  18
24  41   8   3  12  17   6  15
 9   2  11  22   5  14  19  32
 0  25  40  35  20  31  16   0
 0  36  21   0  39   0  33  30
26   0  38   0  34  29   0   0
37   0   0  28   0   0   0   0
 0  27   0   0   0   0   0   0

Number of moves: 42
```

Figure 3: Sample run for Question 2 where the knight started on square $(0, 0)$. In this tour, the knight was only able to visit 42 squares.

3. (7 points) After attempting to write and run a Knight's Tour program, you have probably developed some valuable insights. We will use these to develop a heuristic (or strategy) for moving the knight. Heuristics do not guarantee success, but a carefully developed heuristic greatly improves the chance of success. You may have observed that the outer squares are more troublesome than the squares nearer the center of the board. In fact, the most troublesome, or inaccessible, squares are the four corners. Intuition may suggest that you should attempt to move the knight to the most troublesome squares first and leave open those that are easiest to get to so when the board gets congested near the end of the tour there will be a greater chance of success. We may develop an *accessibility heuristic* by classifying each of the squares according to how accessible they are, and then always moving the knight to the square (within the knight's L-shaped moves, of course) that is most inaccessible. We label a double-subscripted array `accessibility` with numbers indicating from how many squares each particular square is accessible. On a blank chessboard, each center square is rated as 8, each corner square is rated as 2, and the other squares have accessibility numbers of 3, 4, or 6 as shown in Figure 4

	0	1	2	3	4	5	6	7
0	2	3	4	4	4	4	3	2
1	3	4	6	6	6	6	4	3
2	4	6	8	8	8	8	6	4
3	4	6	8	8	8	8	6	4
4	4	6	8	8	8	8	6	4
5	4	6	8	8	8	8	6	4
6	3	4	6	6	6	6	4	3
7	2	3	4	4	4	4	3	2

Figure 4: From how many other squares each square can be reached when the board is empty.

Write a version of the Knight's Tour using the accessibility heuristic. The knight should always move to the square with the *lowest* accessibility number. In case of a tie, the knight may move to any of the tied squares. Therefore, the tour may begin in any of the four corners. (Note: As the knight moves around the chessboard, your program should reduce the accessibility numbers as more squares become occupied. In this way, at any given time during the tour, each available square's accessibility number will remain equal to precisely the number of squares from which that square may be reached.) Run this version of your program. Did you get a full tour? A sample run of a possible solution for Question 3 is shown in Figure 5.

4. (3 points) Modify the program to run 64 tours, one starting from each square of the chessboard. How many full tours did you get?

What to Turn In

Each of the four parts for this assignment are to be submitted through Blackboard. For Part 1, scan your drawing and submit the scanned file. For each of the other three parts you should create a zip file containing the entire directory created by NetBeans and submit the zip file through Blackboard.

```
 1  22   3  18  25  30  13  16
 4  19  24  29  14  17  34  31
23   2  21  26  35  32  15  12
20   5  56  49  28  41  36  33
57  50  27  42  61  54  11  40
 6  43  60  55  48  39  64  37
51  58  45   8  53  62  47  10
44   7  52  59  46   9  38  63

Number of moves: 64
```

Figure 5: Sample run for Question 3. In this tour, the knight was able to visit all 64 squares.