Overall objective: get familiar with the use of several data structures

Notes to make:

* Cannot be recursive, must be iterative procedure with “back tracking “ involving a stack.
* The board state at each instant is an 8×8 array of integers all of whose elements are initially set at −1.
* As you move to a square, you replace the corresponding array element by the sequence number of the move made (such as 1 for the square visited after the first move, 2 for the second move etc.).
* At each instant the board state with any other necessary information is pushed on to the stack. This is to recover from the case when the knight is trapped somewhere, with no unvisited square to move to, without finding the complete solution. In such a case, you back-track along the traversed path by popping the states from the stack, until you find a square from which you can make a valid move.
* While back-tracking you should also unmark the the squares (i.e., put −1 at the corresponding elements of the current board state) so that you can visit those squares again.
* If, at a particular instant, the knight’s position on the board is given by (i,j), there may be at most eight possible moves for the knight which will move it to one of the squares (i − 2,j + 1),(i−1,j +2),(i+1,j +2),(i+2,j +1),(i+2,j −1),(i+1,j −2),(i−1,j −2),(i−2,j −1).
* Also, remember that in the successful trajectory, the knight can visit a square only once.
* The project also requires the use of a singly linked list to store a number of choices of initial conditions for the knight. Before solving the problem for any initial condition, the program should prompt the user incrementally to enter the initial conditions, but should not ask for the number of initial positions that the user wishes to enter. At the end of this data entry mode, the program should display all the positions entered. It should then further prompt to see if the user wants to add, delete, or modify the list of initial positions.
* (i) Implementation of a singly linked list (with associated operations) to store the user inputs concerning the initial positions of the knight.
* (ii) Solving the knight’s tour problem for each initial position using a stack, backtracking, andusing Warnsdoff’s heuristic for the first 32 moves.