CS 326-01-09S
OBJECT ORIENTED DESIGN
SUMMER 2009

Program 3

You are now to update your knight move program to write a knight move class. Previous assignments have allowed you flexibility in the naming and organizing of classes and methods. This one requires a strict adherence to the names and signatures listed. I will be writing the main class for this program, and my code will need to know EXACTLY how to access your methods and classes.

Create a folder called “PG3” in the root of your shared folder. Put all files pertaining to this project in this folder. When you are ready to have your program graded, place a (possibly empty) file called “DONE” into this folder.

Create a class titled “public class Board”. This class will not have a main method; I will supply this. This class will have the following public methods:

public void LoadBoard (bool [,] b);

This will tell your Board class what specific board you care to solve. It will accept a two-dimensional array of booleans. If the board is properly formatted, LoadBoard will return nothing (void). However, if the board is improperly formatted, LoadBoard will throw a MalformedBoardException. One possible malformed board would be one in which the rows are not all the same length; however, you should check for all other bad boards as well. Loading a Board should clear the initial position. We are no longer operating under the assumption that the rows and columns will have a maximum size of 10. They may be any length. Additionally, any configuration of trues and falses in the array is acceptable, provided the board structure is good. The first index in b corresponds to the row number; the second index corresponds to the column number.

public void LoadInitialPosition (Position P);

This will tell your board class what position in your board will be the start position. The Position class is the class defined in the last project, and it must be called exactly that; if you called it something else in the last project, you must change it. (You don't have to change the names of your stack and linked list classes, though; I won't use them directly.) Additionally, your Position class should contain SetRow, SetCol, GetRow, and GetCol methods called exactly that. If the initial position is out of bounds, your method should throw a BadPositionException, called exactly that. If there is no loaded board, your method should throw an UnloadedBoardException, called exactly that.

public Position[] solve ();

This method should find the longest path on the board (including the initial position), and return an array of Position corresponding to this path. If something (board or position) was unloaded, it should throw an UnloadedBoardException, called exactly that. If there is more than one maximal path, it doesn't matter which one is returned. Index 0 of this array should be the initial position, the last index of the array should be the final position in the longest path; in other words, the path should proceed “forwards” through the array.

I still want you to keep this project free of recursion. Use the stack classes you built in Program 2.