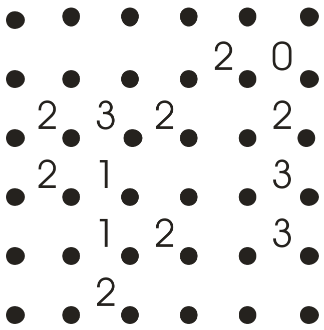
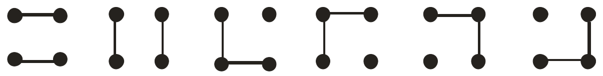
Term Project
Fall 2014

This semester you will implement a game called “Slither”. You will find a web-based version of this game at http://www.puzzle-loop.com. Use examples from the site to test and debug your solution. This game is played on a 5x5 to 25x30 grid with “dots” at each of the grid corners. Some of the squares contain a number between 0 and 3 inclusively. The number identifies the number of lines that will surround that particular square. Your objective is to find a single looped path with no crosses or branches.

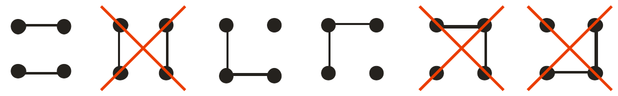
Consider the following board:



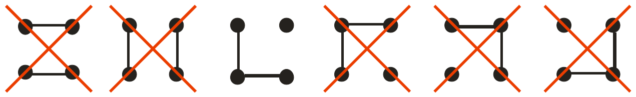
In the first row, the first three squares contain no number. These squares can have 0 to 3 lines on their sides. The last two squares of the row contain 2 and 0, so we know the square with 2 could possibly be:



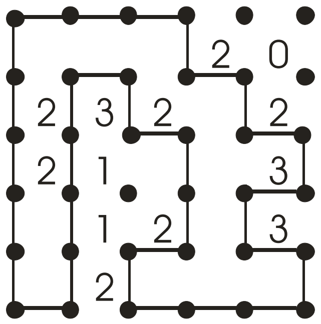
but, because the next square is labeled with a 0, we can eliminate all combinations (#2, 5, and 6) with a vertical edge on the right:



and we can eliminate #1 and #4 because they will not be able to generate a closed loop (due to the restrictions of no edges on the square to the right), leaving us with only one labeling:



Applying other restrictions, we eventually can generate the solution:



To simplify the drawing of our results, we will substitute a “+” for each “dot, “|” for the vertical line and “-” for the horizontal line, making our display of the final solution:

```
+-+-+-+ + +
|     |2 0
+ +-+ +-+ +
|2|3|2  |2
+ + +-+ +-+
|2|1  |  3|
+ + + + +-+
| |1 2| |3
+ + +-+ +-+
| |2|     |
+-+ +-+-+-+
```

CAP 5635 students are required to use LISP in developing their solution. CAP 4621 students can use a commonly used language of their choice (verify that the language is acceptable with the TAs if using other than Lisp, C, C++, or Java. A 10% bonus will be given for using LISP). All solutions will be individual – no group work or assistance from anyone inside or outside of the class.

Project Time Table

The following are important dates when project reports or code submissions are required.

Oct. 24 Initial submission of project

This submission will show that your program can play this game with no intelligence. Your program should provide a description of the game to the user, request and read the name of a file containing a set of data describing the board, should allow a human to specify moves, should make the move and display the new game board, should recognize when a game is completed (a solution has been generated), and should ask if another game is desired (repeating this process if required). Each move will be specified as a triple – two numbers separated by a blank followed by blank and a letter: N N L. No other punctuation or characters is allowed. The two numbers will identify the row then column (starting in the upper left corner) where a line will be added and the letter will identify the line’s position on the square (T for top, B for bottom, L for left, and R for right). For example: 21 T specifies the second row, first column, and top side (shown in red with yellow highlight on the example below). This could have also been specified as: 1 1 B.

Any other added features ("bells and whistles") which you feel should be part of the user interface should be demonstrated here (e.g., numbering the rows and columns are *highly* recommended). Each student will turn in a textual copy of their program (not compiled) on this date with a README.TXT plain text file describing how to execute your program. The main routine for your program should be called “slither”. To demonstrate your program, use the following board (or one of its rotations):

```
+ + +  which has a single solution:  +-+-+
 3 3                                |3 3|
+ + +                                +-+-+

+ + +                               + + +
```

Oct. 27 Written report

A plain text, PDF, or Word file of your written report should be turned in on this date (1-2 pages in length). This report will detail what intelligence you intend to implement in your program. Note: you might not be able (due to time until the end of the semester, time limits for a move, and space within the machine) to fully implement all of this. Your report should therefore be structured to contain the following two sections:

1. This is what I will definitely try to implement and
2. This is what I would love to implement provided I don't have any major difficulties.

Nov. 3 Search Implemented (nothing to submit!)

On this date I *encourage* you to have a search program implemented that performs a blind search for a solution. *No submission of code is required on this date!!* This is merely a road marker for you. From this date until the final submission, you can concentrate on implementing your intelligence.

Dec. 3 Final Code Completed (submitted by Dec. 8) - All projects must be fully implemented and running for final submission

This submission will be used to show what you have as a final program. You will be graded on the program playing a legal game (developing/finding a correct solution), any changes which have been made since the first demonstration, and how well the program plays. Your program should allow humans to play manually or the computer to play automatically. During automatic play, your program should display the initial board, the final board, a list of moves made for the solution, and the total CPU time used to generate that solution. You are to submit files containing your program, the data sets, and a README.TXT plain text file describing how to load your program and any other directions required to execute your program.

Dec. 3 – Dec. 8 Tournament

I will post a series of data sets that you will be required to run your program against. You must provide a table of tournament results (submitted under the assignment “Tournament Results”) identifying the time taken to generate a solution for each board. You must provide the move sequence generated by each of these data sets as a separate file. Each solution must be found in no more than 10 CPU seconds (I reserve the right to change this value higher or lower!).

We will verify your results by testing your program ourselves on one or more of the tournament data sets to verify your results.

Dec. 10 Final written report

This report will detail the entire project. As a minimum this report should include:

1. A description of the game.
2. A description of the implementation approach that was taken.
3. A description of all programs/classes/methods/functions written and all major variables used.
4. A flowchart of the program.
5. A calling hierarchy or UML diagram showing the various programs/methods/functions and how the various programs/methods/functions call each other.
6. A description of what intelligence was implemented and what was not. Discuss why

the non-implemented intelligence was not implemented.

7. A discussion of what you would do differently if you were able to start over.

A table of the CPU times needed to solve each board and the move sequences that were generated by your program when generating each solution should be submitted under the "Tournament Results" assignment. A complete code listing of your program and the tournament boards should be submitted under the "Final Code Submission" assignment.

Grading

The grading of the project will be according to the following scale:

Initial Demonstration 10%
Intelligence Report 15%
Play against tournament data sets 25%: 15% for playing all data sets and reporting your results and 10% for the intelligence of play
Final Demonstration 15%
Final Report 35%