TCSI201 11/30/09 Honors Intro. Comp. Sci.

Final Homework/Project Assignments.

(1) **Team Project. Due Monday 12/7 9:00PM.**

For team credit, each team member present a web site with url

http://www.albany.edu/~netid/tcsi201 that explains and presents the applets that the team developed for the labs. Collaboration is expected, but individuals will be questioned on the basic structure of web sites on the final (significance of directories, index.html, public_html, Unix protection settings, applet tag, a for anchor tag, html tag.)

Supplemental Reading: Any of thousands of web sites or books about html and elementary web development.

(2) **Individual *-Pattern Matcher mini-project. Due Wednesday, 9:20AM**

1. Complete the Matcher.java program by implementing the case when the pattern begins with * which was covered in class. Either create two versions or give the user the option to choose between the "greedy" matcher and the "patient" matcher algorithm variants. In the case when the pattern is *prest and the subject is Csrest, where C is any character, prest is the rest of the pattern and srest is the rest of the subject, the greedy version will first try to match * with C and *prest with srest before trying * with nothing and prest with Csrest. The patient version reverses the order of testing these two cases.

2. Draw 8 activation trees preferably on paper for the runs of the two algorithms on following pairs of pattern and subject:

   - pattern:* subject:ABC
   - pattern:*Z* subject:ZXZ
   - pattern:*Z*A subject:XZYZB
   - Your choice of pattern with 2 *'s and an interesting subject, essentially different from the given three.

   For each pair, draw the activation tree for the "greedy" matcher and the (usually different) activation tree for the "patient" matcher. Use of your program to help figure out the activation trees is encouraged, but understanding the algorithm is critical to avoid utter confusion.

   Supplemental reading: Parts of Kernighan's paper that was handed out. Skip most of the C details but read the English.

(3) **Individual Tic-Tac-Toe project. Due 12/7, 10:00PM on Blackboard**

1. Read Chapter 10 of Kumar's Learning Computing with Robotics but concentrate on pages 250-263. Figure out which parts of Kumar's Tic-Tac-Toe player correspond with which parts of my Java example expressed in TTTFramework.java, Board.java and BadMoveException.java.

2. Kumar's openWins(board,player) used in evaluate(board) counts the number of rows, columns and diagonals in the given board that contain only spaces or the mark for the given player.
3. Begin the project by starting a document to record what you do, what you learn along the way, how you test it and what progress you make. Write it AS YOU GO ALONG. A coherent report that shows genuine learning will earn a score of at least 50%.

4. Implement a version where the computer plays randomly, as explained by Kumar.

5. Implement a version where the computer chooses its move by using the evaluation function, as explained by Kumar.

6. Implement a version where the computer chooses its moves by using full lookahead. I think it's easiest here to fashion the algorithm yourself based on the ideas from Kumar, rather than attempting to translate Kumar's code. Notice that lookahead is most naturally implemented recursively.

7. Of course, TEST all versions you try! Put a short account of your tests and results in the report.