Project - Tic-Tac-Toe game with Java & Prolog

Course/Section: CS6364/501

Last Name: Polavarapu

First Name: Vinay Rahul

NetID (UTD email): vxp220014@utdallas.edu

Do not make any of these problems or your answers to be posted or available in Internet. Do not share with any others. All of the assignment should be done by yourself.

Submission: Submit (upload) softcopy of (1) a word document [this file with your answers and program listing and log of compile-run with test cases], and (2) a zip file of all the files (including all the programs and files that you worked and are needed to compile or run it) via elearning.

Scoresheet

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TTT game  100% | #1  5% | #2 coding  60% | #3 log files  10% | #4  5% | #5  10% | #6  10% |
|  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| No or Poor Documentation  (penalty max 50%) | Demo to TA  (penalty max 50%) |
|  | Please check with TA for demo schedule & appointment. |

\*\* Demo is required (check for the demo schedule from TA to make a reservation) or -50%.

Your documentation here should be well-organized and presented, to follow the flow of your work with ease. Place course information and your name & email (UTD) to the header and page number in footer. Keep 1" margin each side and font-size of 10, single-spaced.

A poor documentation (a word file – this file with your answers) may result in 0 for documentation (-50%).

**Note.** (If your program run is not manageable) Some problems as your program run may take too long to complete (or aborted out of memory or overflow, etc.). If your program runs over 30+ minutes, or producing over a few hundreds of solutions (or the depth of search tree is too big, etc.), please stop and make a note of it in your submission (here and to TA during Demo) and/or to adjust your search to be a bit manageable. One thing that you can do is to output first few dozens (to show that your program is working or to make the length of the move or depth to be shorter, etc.), in order to make the run manageable. Make a note of this and clearly state it in your documentation for the run and solutions and to TA during the demo. Another option is to make your program smarter, to be manageable (instead of running in brute-force manner).

Warning: All of the assignment should be done by yourself and for this course. Do not make any of the problems and course-materials, or your answers to be posted, do not share or make it available in Internet.

Project - Tic-Tac-Toe game with Java & Prolog

Design and implement ttt game to play N games and to keep the score of player1 and player2.

Task#0.

#1. Create project folder (ttt-netid where netid is your netid).

Answer -

A screenshot of a computer

Description automatically generated

Note. Your final project submission to elearning will be: (1) this document and (2) zip file of your project folder containing all your work (including source codes, java files, terminal session log files, etc.). Use protocol(“ttt-log.txt”) for your prolog program to start your terminal session log/history also to be kept in the log-file provided for each task.

#2. Unzip the sample ttt program files in the project folder (tictactoe.java and ttt.pl).

Answer –

A screenshot of a computer

Description automatically generated

Note that this java program (using Connect) may generate exception message (but ignored it) if your Java Run-Time Library installed is above Java v8. For this lab, first, create a folder (to unzip and place the programs here) and all the lab work is to be done here. When you run the jar file to start tic tac toe game. It will ask two paths for: (1) swipl and (2) ttt.pl. The first path for (1) swipl is found in the folder where you install swipl prolog. For my case, it is c:\swipl\bin\swipl and the second path for (2) ttt.pl would be c:\prolog\ttt-game\ttt.pl.

#3. Copy the java program (ttt1.java and its jar file: ttt1.jar and Connect1.java). Update the programs to do the following tasks.

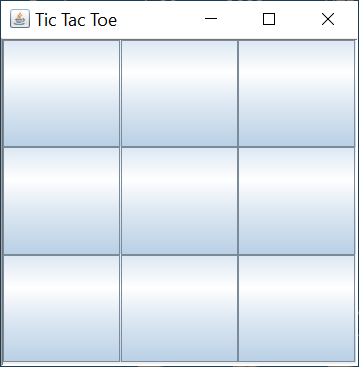
Tasks

#1. Initialize the path names for (a) swipl and (b) ttt.pl so that you do not have to update them in the beginning of the program run. For example,

String prolog = "C:/swipl/bin/swipl";

String ttt = "C:/ttt-rkm010300/ttt.pl";

A screenshot of a computer error

Description automatically generated 

Answer –

A screenshot of a computer error

Description automatically generated

#2. Design and implement ttt-board to add and do the following items.

(1) to display your netid (next to Tic Tac Toe) on the top

(2) to add three buttons:

(a) to start (a new ttt game – one game),

(b) to start N games (to complete N games for computer x computer)

(c) to stop/reset (to end a game).

(3) to display who is the player for player1 (X) human and player2 (O) computer

(4) to have an option to select player1 (X) to be human or computer.

You may use the same prolog code to run for player1 in case of computer.

(5) at the end of each game, display who is the winner

(6) to get an integer N (1-10) from user to play N times.

In addition, display the game # (1-N), and the number of win for player1 and for player2.

(7) at the end of N games, it displays who is the overall winner (player1 or player2).

and to display the total winning score of each player

Answer – (1) And (2) – following are screenshots –

Netid following Tic Tac Toe –

Complete window –

A screenshot of a computer screen

Description automatically generated

On the bottom you can find the buttons.

(5) displaying the winner

A screenshot of a computer

Description automatically generated

(6) Take n input from user – for generating n games.

A screenshot of a computer error

Description automatically generated

Here I entered 5 for number of games!

#3. Your java and prolog programs should provide the log of each step into its own log file

(ttt-java-log.txt and ttt-prolog-log.txt) keeping the records to show the current state of the ttt game:

(a) in the beginning – game starts, game 1 of total N=10 games.

(b) at the end of each game – game ends, for game 1 of total N=10 games,

(c) who is winner – player1 (X) wins

(d) total count of wins for each player

(e) for each step (of X or O), for example, display Player1 X takes the cell (1, 2), etc.

#4. Compile your java program and have a jar file (ttt1.jar) and run of the program.

#5. Your test cases are:

Test case 1. Run human x computer – 1 time

Test case 2. Run human x computer – 3 times

Test case 3. Run computer x computer – 1 time

Test case 4. Run computer x computer – 100 times.

#6. Submit to elearnng:

(1) your word document (this document) with what is done, step by step, each step with subheading, including the listing of the programs of java and prolog, and compile and run of the program) – see below,

and

(2) the zip file folder (TicTacToe2.java and TicTacToe2.jar, all programs, terminal session log/history files of your run).

(3) do the demo of your program & run to TA during office hours. Please wait for TA announcement on demo schedule and guideline.

# Test case 1. Run human x computer – 1 time

A screenshot of a computer

Description automatically generated

Run – Computer wins!

# Test case 2. Run human x computer – 3 times

# A screenshot of a computer Description automatically generated

Run 1 – Computer Wins!

A screenshot of a computer

Description automatically generated

Run 2 – Computer wins

A screenshot of a computer

Description automatically generated

Run 3 – Computer wins!

# Test case 3. Run computer x computer – 1 time

# Test case 4. Run computer x computer – 100 times.

(You may skip most of the output except half page for the beginning and half page for the end of the program run. Your zip file should contain the full log file of this run for TA to check as needed).

Listing of ttt1.java program

import java.awt.\* ;

import java.awt.event.\* ;

import java.io.\* ;

import java.net.\* ;

import javax.swing.\* ;

import javax.swing.border.\* ;

public class Tttboard extends JFrame

                       implements ActionListener {

   JButton b11,b21,b31,

           b12,b22,b32,

           b13,b23,b33,start,reset,n\_game ;

   boolean myturn ;

   BufferedReader br ;

   BufferedWriter bw ;

   Thread connection ;

   Process prologProcess ;

   String prolog ;

   String ttt ;

   /\*\*

     \*  Create a tic tac toe game,

     \*  prolog is the prolog command (e.g. "/opt/local/bin/swipl").

     \*  ttt is the locator for ttt.pl (e.g. "/javalib/TicTacToe/ttt.pl").

     \*/

   public Tttboard(String prolog, String ttt) {

      this.prolog = prolog ;

      this.ttt = ttt ;

      b11 = new JButton("") ;

      b21 = new JButton("") ;

      b31 = new JButton("") ;

      b12 = new JButton("") ;

      b22 = new JButton("") ;

      b32 = new JButton("") ;

      b13 = new JButton("") ;

      b23 = new JButton("") ;

      b33 = new JButton("") ;

      start = new JButton("Start");

      reset = new JButton("reset");

      n\_game = new JButton("n\_game");

      b11.setActionCommand("(1,1).") ; // prolog reads pair term

      b21.setActionCommand("(2,1).") ;

      b31.setActionCommand("(3,1).") ;

      b12.setActionCommand("(1,2).") ;

      b22.setActionCommand("(2,2).") ;

      b32.setActionCommand("(3,2).") ;

      b13.setActionCommand("(1,3).") ;

      b23.setActionCommand("(2,3).") ;

      b33.setActionCommand("(3,3).") ;

      start.setActionCommand("start");

      reset.setActionCommand("reset");

      n\_game.setActionCommand("n\_game");

      Font f = new Font("monospaced",Font.PLAIN,64) ;

      b11.setFont(f) ;

      b21.setFont(f) ;

      b31.setFont(f) ;

      b12.setFont(f) ;

      b22.setFont(f) ;

      b32.setFont(f) ;

      b13.setFont(f) ;

      b23.setFont(f) ;

      b33.setFont(f) ;

      // start.setFont();

      // reset.setFont(f);

      // n\_game.setFont(f);

      b11.addActionListener(this) ;

      b21.addActionListener(this) ;

      b31.addActionListener(this) ;

      b12.addActionListener(this) ;

      b22.addActionListener(this) ;

      b32.addActionListener(this) ;

      b13.addActionListener(this) ;

      b23.addActionListener(this) ;

      b33.addActionListener(this) ;

      start.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                startNewGame(prolog, ttt);

            }

        });

        reset.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                resetGame();

            }

        });

        newGame.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String n = JOptionPane.showInputDialog("Enter the number of games:");

                if (n != null && !n.trim().isEmpty()) {

                    try {

                        int numberOfGames = Integer.parseInt(n.trim());

                        startNGames(numberOfGames);

                    } catch (NumberFormatException ex) {

                        JOptionPane.showMessageDialog(null, "Please enter a valid number.");

                    }

                }

            }

        });

    });

      JPanel panel = new JPanel() ;

      panel.setLayout(new GridLayout(4,3)) ;

      panel.add(b11) ;

      panel.add(b21) ;

      panel.add(b31) ;

      panel.add(b12) ;

      panel.add(b22) ;

      panel.add(b32) ;

      panel.add(b13) ;

      panel.add(b23) ;

      panel.add(b33) ;

      panel.add(start);

      panel.add(reset);

      panel.add(n\_game);

      //this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE) ;

      this.setTitle("Tic Tac Toe vxp220014") ;

      Border panelborder = BorderFactory.createLoweredBevelBorder() ;

      panel.setBorder(panelborder) ;

      this.getContentPane().add(panel) ;

      this.setSize(400,400) ;

      this.setLocation(900,300) ;

      this.myturn = true ;

      Connector connector = new Connector(54321) ;

      connector.start() ;

      Socket sock ;

      try {

         sock = new Socket("127.0.0.1",54321) ;

         br = new BufferedReader(new InputStreamReader(sock.getInputStream())) ;

         bw = new BufferedWriter(new OutputStreamWriter(sock.getOutputStream())) ;

      } catch(Exception x) { System.out.println(x) ; }

      connection = new Thread() {

         public void run() {

            while(true) {

               try{

                  String s = br.readLine() ;

                  computer\_move(s) ;

               } catch(Exception xx) { System.out.println(xx) ; }

            }

         }

      } ;

      connection.start() ;

      Thread shows = new Thread() {

         public void run() {

            setVisible(true) ;

         }

      } ;

      EventQueue.invokeLater(shows);

      // Start the prolog player

      try {

         prologProcess =

           Runtime.getRuntime().exec(prolog + " -f " + ttt) ;

      } catch(Exception xx) {System.out.println(xx) ; }

      // On closing, kill the prolog process first and then exit

      this.addWindowListener(new WindowAdapter() {

         public void windowClosing(WindowEvent w) {

            if (prologProcess != null) prologProcess.destroy() ;

            System.exit(0) ;

         }

      }) ;

   }

   void startNewGame(String prolog, String ttt) {

      connection.stop() ;      new Tttboard(prolog,ttt);

   }

   void resetGame() {

      connection.stop() ;      new Tttboard(prolog,ttt);

   }

   void startNGames(int n) {

      for(int i=0;i<n;i++) {connection.stop() ;      new Tttboard(prolog,ttt);

      }

   }

//       /opt/local/bin/swipl   /javalib/TicTacToe/ttt.pl

   public static void main(String[] args) {

      String prolog = "C:\\Program Files\\swipl\\bin\\swipl" ;

      String ttt = "C:\\ttt-vxp220014\\ttt.pl" ;

      boolean noargs = true ;

      try {

         prolog = args[0] ;

         ttt = args[1] ;

         noargs = false ;

      }

      catch (Exception xx) {

         System.out.println("usage: java TicTactoe  <where prolog>  <where ttt>") ;

      }

      if (noargs) {

         Object[] message = new Object[4] ;

         message[0] = new Label("  prolog command") ;

         message[1] = new JTextField(prolog) ;

         message[2] = new Label("  where ttt.pl ") ;

         message[3] = new JTextField(ttt) ;

         try {

            int I = JOptionPane.showConfirmDialog(null,message,"Where are Prolog and ttt.pl? ",JOptionPane.OK\_CANCEL\_OPTION) ;

            if (I == 2 | I == 1) System.exit(0) ;

            System.out.println(I) ;

            new Tttboard(((JTextField)message[1]).getText().trim(),((JTextField)message[3]).getText().trim()) ;

         } catch(Exception yy) {}

      }

      else

         new Tttboard(prolog,ttt) ;

   }

   void computer\_move(String s) { // " x ## y '

      String[] c = s.split(",") ;

      int x = Integer.parseInt(c[0].trim()),

          y = Integer.parseInt(c[1].trim()) ;

      //System.out.println(x+","+y) ;

      if (x == 1) {

         if (y == 1) b11.setText("O") ;

         else if (y == 2) b12.setText("O") ;

         else if (y == 3) b13.setText("O") ;

      }

      else if (x == 2) {

         if (y == 1) b21.setText("O") ;

         else if (y == 2) b22.setText("O") ;

         else if (y == 3) b23.setText("O") ;

      }

      else if (x == 3) {

         if (y == 1) b31.setText("O") ;

         else if (y == 2) b32.setText("O") ;

         else if (y == 3) b33.setText("O") ;

      }

      if (winner()) {

         JFrame frameC = new JFrame("Winner!");

         frameC.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

      // Create a label.

      Label label = new Label("Winner - Computer");

      // Add the label to the frame.

      frameC.setSize(200,200);

      frameC.setLocation(900,300) ;

      frameC.add(label);

      // Display the frame.

      frameC.setVisible(true);

      connection.stop() ;}

      else  myturn = true ;

   }

   /\*\*

     \* Java player

     \*/

   public void actionPerformed(ActionEvent act) {

      if (!myturn) return ; // otherwise

      String s = ((JButton)act.getSource()).getText() ;

      if (!s.equals("")) return  ;

      ((JButton)(act.getSource())).setText("X") ;

      try {

         bw.write(act.getActionCommand() + "\n") ;

         bw.flush() ;

      } catch(Exception xx) { System.out.println(xx) ; }

      myturn = false ;

      if (winner()){JFrame frameh = new JFrame("Winner!");

      frameh.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

      // Create a label.

      Label label = new Label("Winner - human");

      // Add the label to the frame.

      frameh.setSize(400,400);

      frameh.setLocation(900,300) ;

      frameh.add(label);

      // Display the frame.

      frameh.setVisible(true); connection.stop() ;}

   }

   /\*\*

A screen shot of a computer program

Description automatically generated/\*

If Java player closes GUI, then Prolog process is terminated.

Java process monitors "win" status of both players, signals a win,

and closes the connector and prolog player.

Prolog justs plays given position.

Write all of this up; it is interesting.

\*/

Listing of Connector1.java program

import java.io.\*;

import java.net.\*;

import java.util.\* ;

import javax.swing.\* ;

/\*\*

  \*  A Connector is a server that listens for I/O connections

  \*  at a port.  Each client gets a Socket and buffered

  \*  i/o streams bundled in a Transducer object.

  \*  Text read at any input is rebroadcast to other clients.

  \*  <pre>

  \*  =====================================

  \*   Copyright 1999-- John R. Fisher

  \*   jrfisher@csupomona.edu

  \*  =====================================

  \*  </pre>

  \*  @author jrfisher@csupomona.edu

  \*/

public class Connector extends Thread {

   int clientNum ;

   int port ;

   ServerSocket portalSocket ;

   Vector collaborators ; // Object output streams for clients

   public Connector(int port) {

      this.clientNum = 1 ;

      this.port = port ;

      this.collaborators = new Vector() ;

   }

   public void run() {

      // Catch big exceptions that prevent server from continuing.

      try { // 1

         portalSocket = new ServerSocket(port) ;

         while(true) {

            // Catch smaller exceptions so server itself can continue..

            try { // 2

               Socket soc = portalSocket.accept() ;

               BufferedWriter out =

                  new BufferedWriter(new OutputStreamWriter(soc.getOutputStream())) ;

               collaborators.add(out) ;

               System.out.println("Spawning Transducer for " + clientNum) ;

               Transducer b =

                  new Transducer(this,

                               new BufferedReader(new InputStreamReader(soc.getInputStream())),

                               out,

                               clientNum) ;

               b.start() ;

               clientNum++ ;

            }

            catch(Exception e2) {

               JOptionPane.showMessageDialog(null,e2.toString(),"CONNECTOR EXCEPTION #2",JOptionPane.WARNING\_MESSAGE) ;

            }

         }

      }

      catch (Exception e1) {

         // Could not make ServerSocket

         JOptionPane.showMessageDialog(null,e1.toString(),"CONNECTOR EXCEPTION #1",JOptionPane.WARNING\_MESSAGE) ;

      }

   }

   /\*\*

     \*  From the command line ...

     \*     java -classpath <path> Connector <port#>

     \*/

   public static void main(String[] args) {

      try {

         int port = Integer.parseInt(args[0]) ;

         Connector prtl = new Connector(port) ;

         prtl.start() ;

         //System.out.println("Starting portal on port " + prtl.portalSocket.getInetAddress() +  port) ;

      }

      catch(Exception e) {

         System.out.println(e) ;

         System.out.println("usage: java -classpath <> Connector <port>") ;

      }

   }

}

/\*\*

  \*  A Transducer is an attachment to a particular client.

  \*  The Transducer listens to this client and rebroadcasts its

  \*  contribution.

  \*/

class Transducer extends Thread {

   BufferedReader in ;

   BufferedWriter out ;

   int client ;

   Connector portal ;

   Transducer(Connector p, BufferedReader instream,

                    BufferedWriter outstream, int k) {

      this.portal = p ;

      in = instream ;

      out = outstream ;

      client = k ;

   }

   public void run() {

      while(true) {

         try {

            // if can read a good input ...

            String s = in.readLine() ;

            if (s == null) {    // THIS CLIENT IS GONE ...

               portal.collaborators.remove(out) ;   // remove this client from server

               break ;  // stop running this client

            }

            // try to tell everyone ...

            Iterator it = portal.collaborators.iterator() ;

            while(it.hasNext()) {

               // Avoid bad collaborators if possible or necessary ...

               BufferedWriter bw = null ;

               try {

                  bw = (BufferedWriter)(it.next()) ;

                  if (bw != out) {  // don't tell self

                     bw.write(s+"\r") ; // return for readLine at other end

                     bw.flush() ;

                  }

               }

               catch (Exception e2) {

                  JOptionPane.showMessageDialog(null,e2.toString(),"TRANSDUCER EXCEPTION #2",JOptionPane.WARNING\_MESSAGE) ;

               }

            }

         }

         // running exception not otherwise handled ...

         catch(Exception e1) {

            JOptionPane.showMessageDialog(null,e1.toString(),"TRANSDUCER EXCEPTION #1",JOptionPane.WARNING\_MESSAGE) ;

         }

      }

   }

}

Listing of ttt1.pl program

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Prolog TicTacToe alpha-beta expert

%% Design to play against human (Java GUI)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%  The empty Tac Tac Toe board

%%   Z1 Z2 Z3

%%   Z4 Z5 Z6   ~   [Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9]

%%   Z7 Z8 Z9

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:- dynamic board/1.

init:-

   retractall(board(\_)),

   assert(board([\_Z1,\_Z2,\_Z3,\_Z4,\_Z5,\_Z6,\_Z7,\_Z8,\_Z9])).

:- init.

%%%%%

%%  Generate possible marks on a free spot on the board.

%%  Use mark(+,+,-X,-Y) to query/generate possible moves (X,Y).

%%%%%

mark(Player, [X|\_],1,1) :- var(X), X=Player.

mark(Player, [\_,X|\_],2,1) :- var(X), X=Player.

mark(Player, [\_,\_,X|\_],3,1) :- var(X), X=Player.

mark(Player, [\_,\_,\_,X|\_],1,2) :- var(X), X=Player.

mark(Player, [\_,\_,\_,\_,X|\_],2,2) :- var(X), X=Player.

mark(Player, [\_,\_,\_,\_,\_,X|\_],3,2) :- var(X), X=Player.

mark(Player, [\_,\_,\_,\_,\_,\_,X|\_],1,3) :- var(X), X=Player.

mark(Player, [\_,\_,\_,\_,\_,\_,\_,X|\_],2,3) :- var(X), X=Player.

mark(Player, [\_,\_,\_,\_,\_,\_,\_,\_,X|\_],3,3) :- var(X), X=Player.

%%%%%

%%  Move

%%%%%

move(P,(1,1),[X1|R],[P|R]) :- var(X1).

move(P,(2,1),[X1,X2|R],[X1,P|R]) :- var(X2).

move(P,(3,1),[X1,X2,X3|R],[X1,X2,P|R]) :- var(X3).

move(P,(1,2),[X1,X2,X3,X4|R],[X1,X2,X3,P|R]) :- var(X4).

move(P,(2,2),[X1,X2,X3,X4,X5|R],[X1,X2,X3,X4,P|R]) :- var(X5).

move(P,(3,2),[X1,X2,X3,X4,X5,X6|R],[X1,X2,X3,X4,X5,P|R]) :- var(X6).

move(P,(1,3),[X1,X2,X3,X4,X5,X6,X7|R],[X1,X2,X3,X4,X5,X6,P|R]) :- var(X7).

move(P,(2,3),[X1,X2,X3,X4,X5,X6,X7,X8|R],[X1,X2,X3,X4,X5,X6,X7,P|R]) :- var(X8).

move(P,(3,3),[X1,X2,X3,X4,X5,X6,X7,X8,X9|R],[X1,X2,X3,X4,X5,X6,X7,X8,P|R]) :- var(X9).

%%%%%

%%  Record a move: record(+,+,+).

%%%%%

record(Player,X,Y) :-

   retract(board(B)),

   mark(Player,B,X,Y),

   assert(board(B)).

%%%%%

%%  A winning line is ALREADY bound to Player.

%%  win(+Board,+Player) is true or fail.

%%    e.g., win([P,P,P|\_],P).  is NOT correct, because could bind

%%%%%

win([Z1,Z2,Z3|\_],P) :- Z1==P, Z2==P, Z3==P.

win([\_,\_,\_,Z1,Z2,Z3|\_],P) :-  Z1==P, Z2==P, Z3==P.

win([\_,\_,\_,\_,\_,\_,Z1,Z2,Z3],P) :-  Z1==P, Z2==P, Z3==P.

win([Z1,\_,\_,Z2,\_,\_,Z3,\_,\_],P) :-  Z1==P, Z2==P, Z3==P.

win([\_,Z1,\_,\_,Z2,\_,\_,Z3,\_],P) :-  Z1==P, Z2==P, Z3==P.

win([\_,\_,Z1,\_,\_,Z2,\_,\_,Z3],P) :-  Z1==P, Z2==P, Z3==P.

win([Z1,\_,\_,\_,Z2,\_,\_,\_,Z3],P) :-  Z1==P, Z2==P, Z3==P.

win([\_,\_,Z1,\_,Z2,\_,Z3,\_,\_],P) :-  Z1==P, Z2==P, Z3==P.

%%%%%

%%  A line is open if each position is either free or equals the Player

%%%%%

open([Z1,Z2,Z3|\_],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([\_,\_,\_,Z1,Z2,Z3|\_],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([\_,\_,\_,\_,\_,\_,Z1,Z2,Z3],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([Z1,\_,\_,Z2,\_,\_,Z3,\_,\_],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([\_,Z1,\_,\_,Z2,\_,\_,Z3,\_],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([\_,\_,Z1,\_,\_,Z2,\_,\_,Z3],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([Z1,\_,\_,\_,Z2,\_,\_,\_,Z3],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

open([\_,\_,Z1,\_,Z2,\_,Z3,\_,\_],Player) :- (var(Z1) | Z1 == Player),(var(Z2) | Z2 == Player), (var(Z3) | Z3 == Player).

%%%%%

%% Calculate the value of a position, o maximizes, x minimizes.

%%%%%

value(Board,100) :- win(Board,o), !.

value(Board,-100) :- win(Board,x), !.

value(Board,E) :-

   findall(o,open(Board,o),MAX),

   length(MAX,Emax),      % # lines open to o

   findall(x,open(Board,x),MIN),

   length(MIN,Emin),      % # lines open to x

   E is Emax - Emin.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% using minimax procedure with alpha-beta cutoff.

% Computer (o) searches for best tic tac toe move,

% Human player is x.

% Adapted from L. Sterling and E. Shapiro, The Art of Prolog, MIT Press, 1986.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:- assert(lookahead(2)).

:- dynamic spy/0.  % debug calls to alpha\_beta

:- assert(spy).    % Comment out stop spy.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

search(Position,Depth,(Move,Value)) :-

   alpha\_beta(o,Depth,Position,-100,100,Move,Value).

alpha\_beta(Player,0,Position,\_Alpha,\_Beta,\_NoMove,Value) :-

   value(Position,Value),

   spy(Player,Position,Value).

alpha\_beta(Player,D,Position,Alpha,Beta,Move,Value) :-

   D > 0,

   findall((X,Y),mark(Player,Position,X,Y),Moves),

   Alpha1 is -Beta, % max/min

   Beta1 is -Alpha,

   D1 is D-1,

   evaluate\_and\_choose(Player,Moves,Position,D1,Alpha1,Beta1,nil,(Move,Value)).

evaluate\_and\_choose(Player,[Move|Moves],Position,D,Alpha,Beta,Record,BestMove) :-

   move(Player,Move,Position,Position1),

   other\_player(Player,OtherPlayer),

   alpha\_beta(OtherPlayer,D,Position1,Alpha,Beta,\_OtherMove,Value),

   Value1 is -Value,

   cutoff(Player,Move,Value1,D,Alpha,Beta,Moves,Position,Record,BestMove).

evaluate\_and\_choose(\_Player,[],\_Position,\_D,Alpha,\_Beta,Move,(Move,Alpha)).

cutoff(\_Player,Move,Value,\_D,\_Alpha,Beta,\_Moves,\_Position,\_Record,(Move,Value)) :-

   Value >= Beta, !.

cutoff(Player,Move,Value,D,Alpha,Beta,Moves,Position,\_Record,BestMove) :-

   Alpha < Value, Value < Beta, !,

   evaluate\_and\_choose(Player,Moves,Position,D,Value,Beta,Move,BestMove).

cutoff(Player,\_Move,Value,D,Alpha,Beta,Moves,Position,Record,BestMove) :-

   Value =< Alpha, !,

   evaluate\_and\_choose(Player,Moves,Position,D,Alpha,Beta,Record,BestMove).

other\_player(o,x).

other\_player(x,o).

spy(Player,Position,Value) :-

   spy, !,

   write(Player),

   write(' '),

   write(Position),

   write(' '),

   writeln(Value).

spy(\_,\_,\_). % do nothing

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% For testing, use h(+,+) to record human move,

%%% supply coordinates. Then call c (computer plays).

%%% Use s to show board.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h(X,Y) :-

   record(x,X,Y),

   showBoard.

c :-

   board(B),

   alpha\_beta(o,2,B,-200,200,(X,Y),\_Value),

   record(o,X,Y),

   showBoard.

showBoard :-

   board([Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9]),

   write('    '),mark(Z1),write(' '),mark(Z2),write(' '),mark(Z3),nl,

   write('    '),mark(Z4),write(' '),mark(Z5),write(' '),mark(Z6),nl,

   write('    '),mark(Z7),write(' '),mark(Z8),write(' '),mark(Z9),nl.

s :- showBoard.

mark(X) :-

   var(X),

   write('#').

mark(X) :-

   \+var(X),

   write(X).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%  Play tic tac toe with the Java GUI

%%%  using port 54321.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

connect(Port) :-

   tcp\_socket(Socket),

   gethostname(Host),  % local host

   tcp\_connect(Socket,Host:Port),

   tcp\_open\_socket(Socket,INs,OUTs),

   assert(connectedReadStream(INs)),

   assert(connectedWriteStream(OUTs)).

:- connect(54321).  % Comment out for testing

ttt :-

   connectedReadStream(IStream),

   read(IStream,(X,Y)),

   record(x,X,Y),

   board(B),

   alpha\_beta(o,2,B,-200,200,(U,V),\_Value),

   record(o,U,V),

   connectedWriteStream(OStream),

   write(OStream,(U,V)),

   nl(OStream), flush\_output(OStream),

   ttt.

:- ttt.             % Comment out for testing