Homework Assignment 2

Part B

Iterative Deepening Minimax Algorithm with Alpha-Beta Pruning
Due: October 18th at 1pm

This part has a programming task, and a written part which is based on the programming question. Please use *handin* to turn in the program part, and turn in the written part separately in class (preferably typed).

Problem Description

The PEG game is a simple 2 player board game. One player will be trying to collect as many points as possible by capturing blue cells, while the opponent will be trying to collect as many points as possible by capturing red cells. The green cells are neutral. The objective is for a player to collect more points by capturing their target cells (and the associated points) than their opponent.

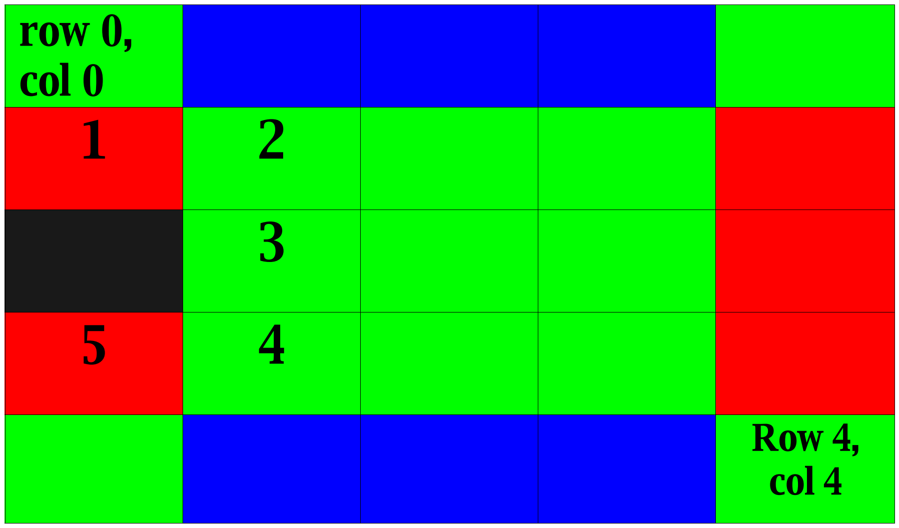

Figure 5. 5x5 Peg game board

Objective: Collect more points than your opponent by capturing target cells.
Game Setup: The board is setup by setting each cell in the NxN matrix to be:

- **nutral** – no points for either player
- **blue cell** – a cell that has some value for the first player (max)(usually 1)*
- **red cell** – a cell that has some value for the second player (min)(usually -1)*
- **played** – a cell that is not available for selection be either player

*note that cells may have varying values and need not always be worth 1. However the maximum points for any cell will be no greater than 50.

Game Play: The first player to move (max) may select any un-played cell in the board. Players then alternate taking turns. Each player can only select a cell that is not played and is adjacent to the last players move (In Figure 5. the player moving next can only select

from cells labeled 1, 2, 3, 4 and 5.). A running score is kept. When either player moves to a blue or red cell, the score is updated by adding in the cells value.
Game End: When a player has no legal move the game ends.

A **winner** is declared based upon the score. If the score is greater than 0 then player 1 is declared the winner. If the score is less than 0 then player 2 is declared the winner. Otherwise the game is a **Draw**.

Program a computer player for the game using ***iterative deepening minimax algorithm with alpha-beta pruning***. You are given a maximum of 10 **seconds** to make a move (during the tournament), however the timeout limit can be modified by the instructors when evaluating your player.

Your computer player will be used to play against other human or computer players (from other classmates or instructors).Your programmed player must be able to play as player 1 (the player who makes the first move and try to capture the blue cells) or player 2 (the player trying to capture the red cells).

Program Specification

A controller program has been written for the Peg game. It will create and manage a game board. A game board can range from a 3X3 board to a 15X15 board. The controller program is written in JAVA and you need to implement your computer player in the form of a JAVA class. The name of your player class should take the form:

CS540<section><class login>

For example, CS5402yong, if you are in section 2 and have a login of yong. Rename the PlayerTemplate.java file to have this form and rename the class in that file to have the same form. That is the file you will need to modify to make your player.

Your program should run on any of the UNIX machines in the lab. It will be invoked as part of the controller program as follows:

```
java HW2 <Timeout><Play Mode><Player Name1><PlayerName2><Board
Size><Board Layout>
```

where,

<Timeout> is used to specify the time limit for each move in terms of **number of minutes**. This can range from 0.1 to 3.
<Play Mode> is used to specify whether the game is a computer-computer, human-computer, or human-human game. Valid values are: **cc, hc, hh**.
<Player Name1> is the same as the name of the java class you created for your

computer player. Leave this parameter out if the play mode is 2 human players (hh).

<Player Name2> is the same as the name of the java class you created for your computer player. Leave this parameter out if the play mode is 2 human players (hh) or one human player (hc).

<Board Size> is any number between 3 and 15 (including 3 & 15).

<Board Layout> the name of a file that specifies a custom game board. This is a text file that contains only integers. This parameter is optional. Check boardlayoutNUM.txt files for examples

examples would be:

```
java HW2 3 cc CS5401pradheep CS5402yong 5
java HW2 0.1 hc CS5402yong 5 BoardLayout5.txt
java HW2 0.5 hh 5 BoardLayout5.txt
```

Notes

[1] Pay attention to what heuristic you will use to select your first move if you are player 1.

[2] Your program needs to provide 2 public int variables for the controller to obtain your next move directly. The variables should be called rowNextMove and colNextMove. For example, as shown in Fig. 5, the top leftmost cell is represented by rowNextMove = 0 and colNextMove = 0.

[3] DO NOT under any condition cause the controller program to crash. Pay attention to your recursive functions and make sure they terminate properly. If your program causes the controller to crash, you will FAIL this programming part of this homework.

[4] Make sure to check for legal move and any return of illegal move to the controller program will terminate the game with you loosing that game.

[5] The game board is represented as an **int[][] gameBoard** in the controller program. Your program needs to call a method, **int[][] getBoard()**, to get a copy(pointer) of the current game board to compute the next move.

[6] Remember the controller program will have a timeout mechanism and your program must be able to provide a legal move after the timeout.

[7] Your computer player needs to provide a public Boolean variable called ***minFlag*** that the game controller can set to indicate whether your player is a min player or a max player. If ***minFlag*** is set with a true value, your computer player is a min player, otherwise it is a max player.

[8] When reading the game board, all cells with value less than or equal to -55 have been occupied by previous player moves.

Written Part:
Question 8

(a) Describe the heuristic (EVAL function) you used at non-terminal states.

(b) Describe the heuristic you used for selecting the “first” move of the game (when you are the first player to move).

(c) For a 15x15 board size PEG game, assume a constant branching factor of 8, assume the game tree is complete (all branches are present, every branch has depth 15*15), if in each state the game board is stored using an array of type int[15][15] (assuming 4 bytes to store an integer value), and the program can visit 10 million states per second, calculate the amount of memory and time needed for standard minimax.