Artificial Intelligence Homework 2

1. Consider setting up a simplified version of the game of *Pacman* as a 2-player, zero-sum, turn-taking game. (For those of you not familiar with Pacman, we will explain the simplified version here that we will use:)

   Pacman is played on a 2-d grid with m rows and n columns. You can imagine that some movements around the grid are blocked by walls so that the grid forms somewhat of a maze. The actual positions of the walls are not important for this problem; just imagine that moving from each cell on the grid to all four adjacent cells is not always possible.

   One player (let's say MAX) controls "Pacman," who is always located in one grid cell, but can move left, right, up, or down around the grid, unless blocked by a wall or the grid boundaries. Pacman's job is to eat all of the dots on the grid (each grid cell may or may not contain a dot, and when Pacman moves into a cell with a dot, he "eats" it and it disappears). On MAX's turn, Pacman may move one cell left, right, up, or down, unless blocked.

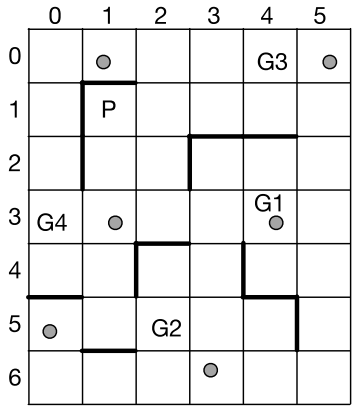
   The other player (MIN) controls the ghosts. There are four ghosts on the board. Like Pacman, each ghost occupies one cell of the grid. Ghosts can also move left, right, up, or down, unless blocked by a wall or the grid boundaries (just like Pacman). The job of the Ghosts is to capture Pacman, which is done by touching him (if a ghost occupies the same square as Pacman, then that counts as a touch). On MIN's turn, **all** four ghosts may individually move one cell left, right, up, or down, unless blocked. All the ghosts move simultaneously, but each ghost may move in whatever direction it wants (or rather, the directions that MIN wants).

   This game is fully observable, so Pacman knows where all the ghosts are, and the ghosts know where Pacman is.

   Pacman (MAX) wins if he eats all the dots on the board before getting captured by a ghost. The ghosts (MIN) win if one of them can capture Pacman before he eats all the dots.

   We will represent a state by the following pieces of information together:
   - a (row, col) coordinate of where Pacman is on the board.
   - four (row, col) coordinates, called (gr1, gc1), (gr2, gc2), etc, representing where the ghosts are.
   - a map called `dots` from (row, col) coordinates to Booleans, specifying whether each (row, col) cell contains a dot or not.

   Here is a sample grid:



In this grid, m=7 and n=6. (row, col) = (1, 1). (gr1, gc1) = (3, 4), (gr2, gc2) = (5, 2). [Similarly for the other 2 ghosts.] The dots map would have keys for each (row, col) on the board, but the only values of "true" would be for the (row, col) keys where the gray dots are. All the other (row, col) keys would have "false" values.

Answer the following questions about this 2-player game:

a) What is the largest number of possible actions the MAX player could have from a state? Briefly say how you came up with this number.

b) What is the largest number of possible actions the MIN player could have from a state? Briefly say how you came up with this number.

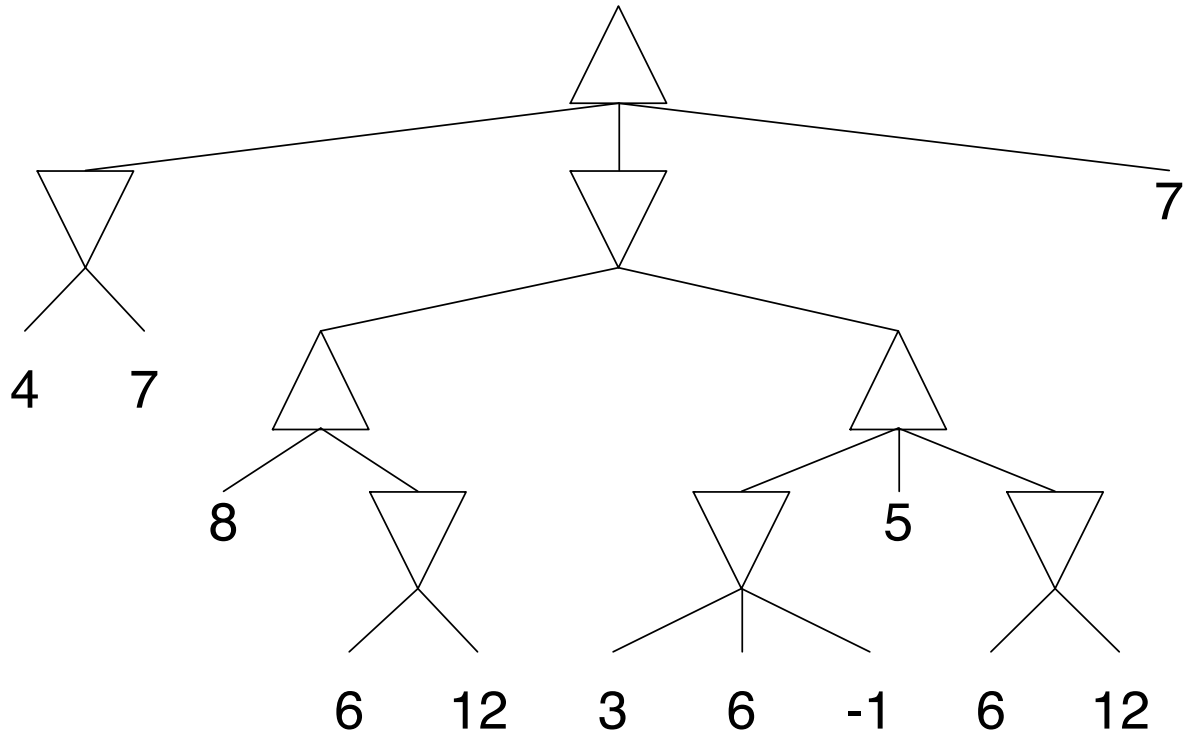
c) Give pseudocode or a mathematical definition of what the IS-TERMINAL(*s*) function might look like. Remember, argument *s* is a state. In other words, give me pseudocode or a mathematical function that returns true if the state is terminal, and false otherwise.

d) Give pseudocode or a mathematical definition of what the UTILITY(*s, p*) function might look like. Your utility function should prioritize winning more quickly. That is, a state where Pacman has eaten all the dots in 100 moves should get a higher utility value than one where he has eaten all the dots in 200 moves. Similarly, for ghosts, states where they capture Pacman in fewer moves should get "more negative" utility values. You may assume you have access to a variable with the number of moves made so far.

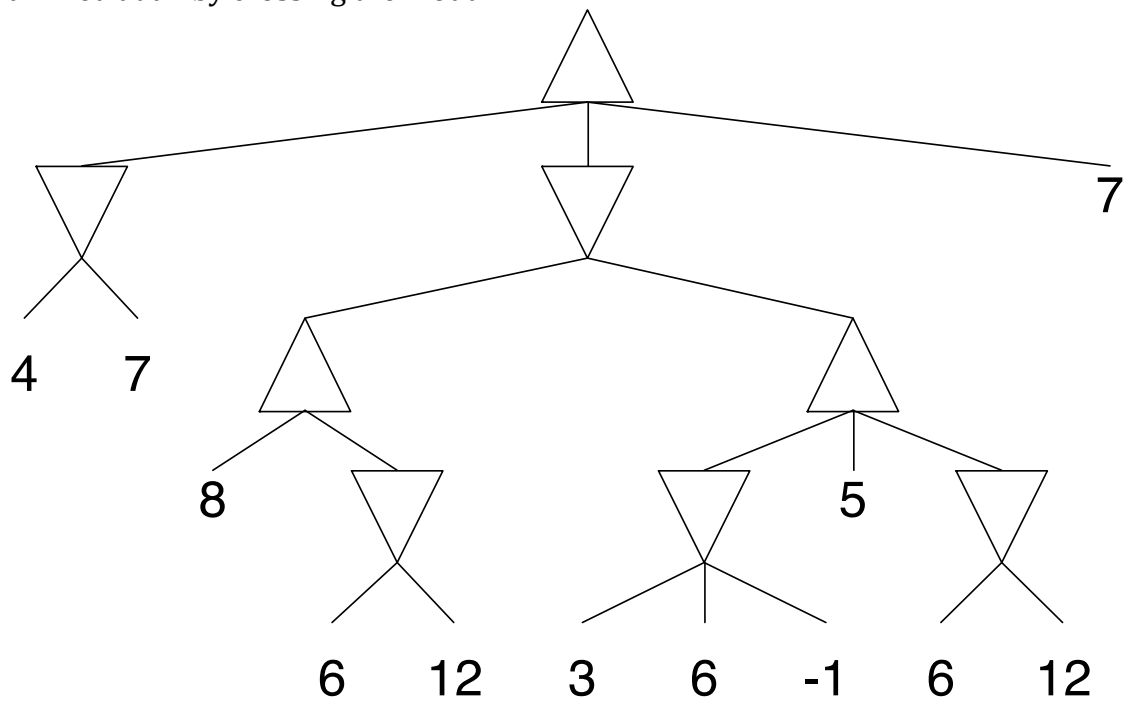
   Remember, argument *s* is a state, and *p* is the player whose perspective we're calculating the utility from, but just do it from MAX's perspective (so assume *p* = MAX). So if Pacman wins, the utility should be positive, if the ghosts win, it should be negative. I suppose a tie is possible if Pacman moves into a cell with the last edible dot, but a ghost is there. Like the previous question, you can write me some pseudocode or give this as a mathematical function.

2. *(For this problem, you will probably want to print out these two pages and turn them in with the rest of your written work, or else re-draw the trees yourself.)*

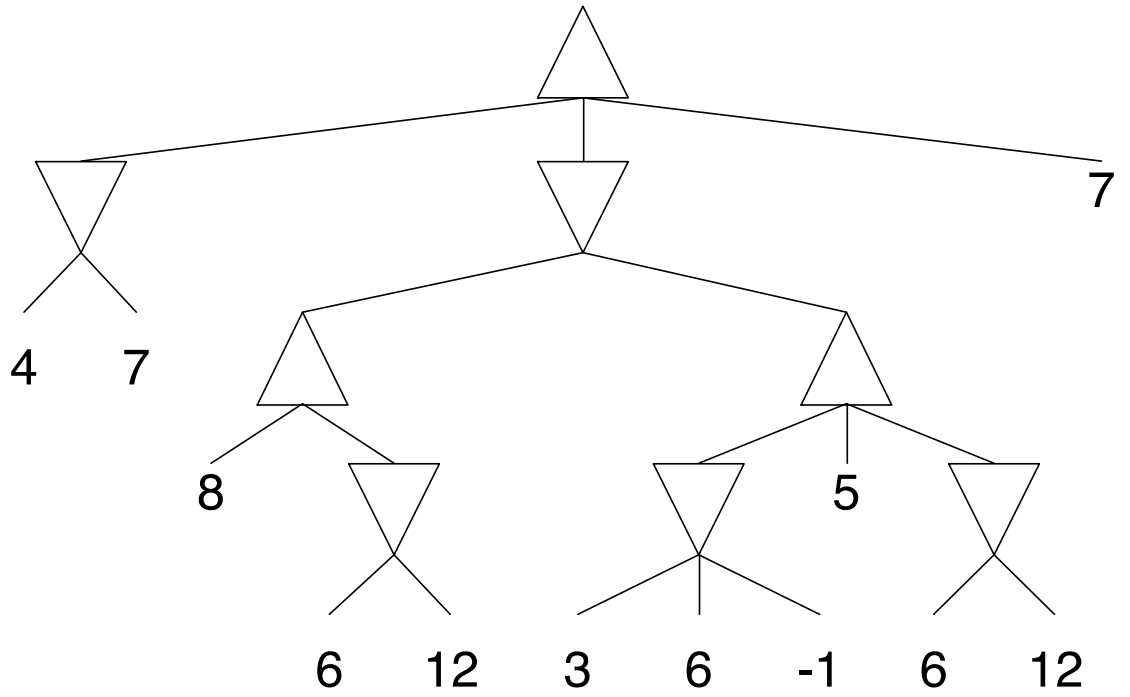
   Run **minimax** (no alpha-beta) on the following game tree, filling in values for each internal node. The first player is MAX (triangle pointing up). MIN nodes are downward-pointing triangles.



3. Run **minimax with alpha-beta pruning** on the same tree, with left-to-right node expansion (that is, consider the children of a node in left-to-right order, as we've done normally). Show the values of alpha and beta at each node (you may want to copy this tree onto another sheet of paper if you need more room), which values get passed up the tree by using arrows, and which nodes are not examined at all by crossing them out.



4. Run **minimax with alpha-beta pruning**, but now consider the children of a node in right-to-left order.



5. For this problem, you will generate a game tree from scratch and then apply minimax.

   Consider the following game, which we will call simply "Seven." The game is played between two players, who alternate saying numbers to each other. The first player always says "1", and the second responds by adding either 1 or 2 to the number, which they then say out loud. The first player must then add 1 or 2 to whatever the second player said, and so on. The number may never go higher than 7, and the player who is forced to say 7 loses.

   Here is an example:

   P1: 1
   P2: 3
   P1: 4
   P2: 6
   P1: 7 ← P1 is forced to say 7, and therefore loses.

   a) Draw the entire game tree for this game, using the notation from class and the book. In other words, use upward-pointing triangles for MAX nodes, and downward for MIN. Inside each triangle (a state), put the number that the player at that state starts with. Because the first player must always say "1", we will make MAX the second player, who gets to pick between saying "2" and "3". Therefore the root node at the top of the tree is an upward pointing triangle for MAX with "1" in it. The "1" should have two downward-pointing triangle children, illustrating MAX's two possible choices. Each leaf node at the bottom of the tree will have a "7" in it. Note that there will be lots of duplicate nodes, and furthermore, some upward-pointing triangles and downward-pointing triangles will have the same number in them.

      You may find this easier to draw on a piece of paper in landscape orientation, rather than portrait.

   b) Label each "7" state (the terminal states) with a +1 if MAX wins, and a -1 if MIN wins. Hint: if it's an upward-pointing triangle with a 7, that means the *other* player (MIN) was forced to say 7, so these are wins for MAX.

      Run minimax on the entire tree, labeling each node off to the side with a +1 or -1 showing the minimax value at that node (similar to how we did this in class with the Nim tree). Make it clear which numbers are the minimax values and which are the numbers of the game itself. For instance, you could circle the minimax values since the states are in triangles. Of course, you should end up with a single minimax value at the root "1" node.

   c) Assuming both players play optimally, who will win?

   d) You will notice, assuming you did part (b) correctly, that each number in the game (1 through 7) is either a "winning number" or a "losing number." For instance, you can see in the game tree that 7 is winning number because all the nodes with 7s are wins for the player in that state (upward-pointing triangles with 7 are wins for MAX, downward-pointing triangles with 7 are wins for MIN). Also, note that 6 is a "losing number" for the exact opposite reason.

      What are the other winning and losing numbers? (In other words, for the numbers 1 through 5, tell me if each one is a winning or losing number.)