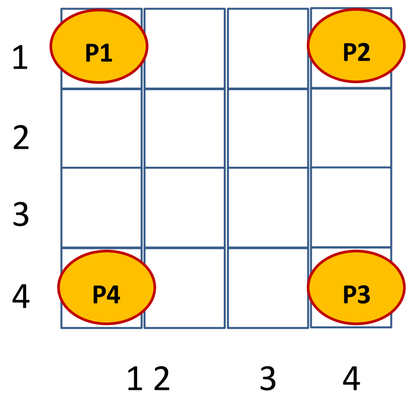
The University of Texas at Dallas
CS 6364
Artificial Intelligence
Fall 2020
Instructor: Dr. Sanda Harabagiu
Grader/ Teaching Assistant: Maxwell Weinzierl

Name: Pratik Deshpande NetID:prd190001

<u>Homework 2: 100 points (20 points extra-credit)</u>

Issued September 22, 2020
Due October 7, 2020 before midnight
Submit only in eLearning

<u>PROBLEM 1:</u> Multi-player Games (**60 points**)
Four players are playing a game which is illustrated in the following Figure:



Player 1 is initially positioned in square [1,1], Player 2 is positioned in square [1,4], Player 3 is positioned in square [4,4] and Player 4 is positioned in square [1,4]. The player that will reach first the diagonally opposite position will win the game. Player 1 will start the game, followed by Player 2, Player 3 and Player 4.
All Players can move either up, down, left or right, if the square in that direction is available and not occupied by any other player.
For example, Player 1 initially can move only to the right or down.

Question 1: How do you represent each node of the game? (**3 points**) How is the initial node of the game represented? (**1 point**)

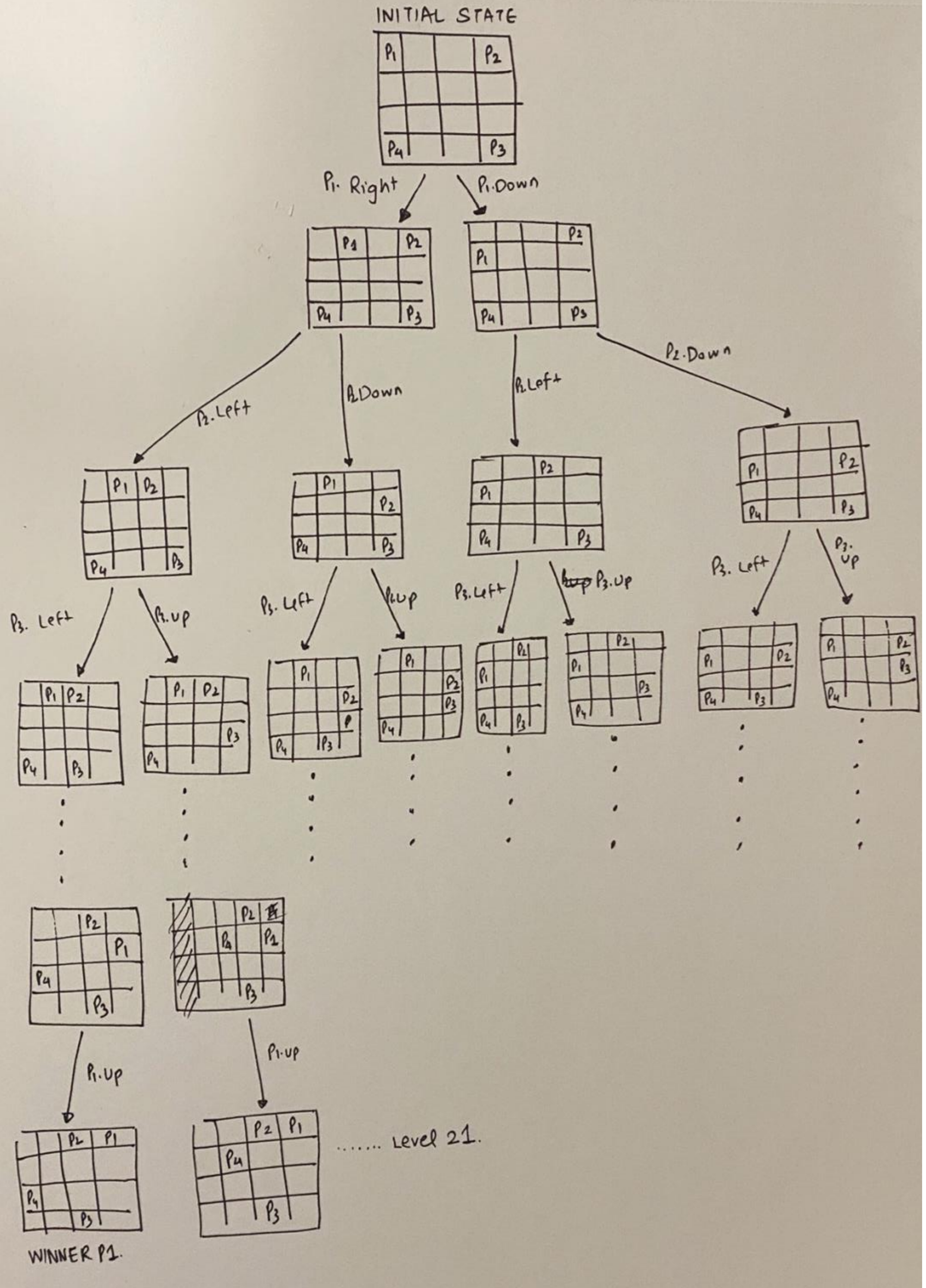
```
ROOT = {"State": {1:(1,1), 2:(4,1), 3:(4,4), 4:(1,4)}, "Parent": "None", "Level": 0, "Action": "None",
"CurrentPlayer": 0,
   "Repeated": False, "Terminating": False, "Winner": "None", "WinStates" : {1:(4,4), 2:(1,4), 3:(1,1), 4:(4,1)},
"Utility": {}}
```

Programming Assignment for Problem 1:

A. Write code that generates the game tree for this multi-player game. (**15 points**) The output should use the following format:
[*Current player* =???| Father node (if not initial node) =???| Action= ????| Current game node =???| if the game node is repeated, write REPEATED; if the current game node corresponds to a winning situation for one of the players, write WINS[PLAYER???]]
Hint: Successors of repeated game nodes should not be considered!

- **Program in Folder**

Question 2: Draw the game tree based on the output of your code. (**11 points**) Note: You can draw the game tree on a piece of paper, take a picture and attach it to your answers, which will be returned in a PDF file.



INITIAL STATE
P1. Right
P1.Down
P2.Left
P2.Down
P2.Left
P2.Down
P3. Left
P3.UP
P3. Left
P3.UP
P3.Left
~~up~~ P3.UP
P3. Left
P3. UP
P1.UP
P1.UP
...... Level 21.
WINNER P1.

Question 3: If Player 1 wins, the utility should be 200. If Player 2 wins, the utility should be 300. If Player 3 wins, the utility should be 400 and if Player 4 wins, the Utility should be 500. In any case, because this is not a zero-sum game, the other players also receive utility values:

③ When Player 1 wins, Player 2 receives utility=10, Player 3 receives utility=30, and Player 4 receives utility=10;

③ When Player 2 wins, Player 1 receives utility=100, Player 3 receives utility=150, and Player 4 receives utility=200;

③ When Player 3 wins, Player 1 receives utility=150, Player 2 receives utility=200, and Player 4 receives utility=300;

③ When Player 4 wins, Player 1 receives utility=220, Player 2 receives utility=330, and Player 3 receives utility=440;

If the minimax values of the repeated states shall be ignored, compute the MINIMAX values in all other states of the game, using the following program assignment.

<u>Programming Assignment for Problem 1:</u> (**15 points**)

B. Write code that computes the MINIMAX values for all non-repeated game nodes and display the values of MINIMAX in the following format:

[*Current player*   Fa he node if no ini ial node   Ac ion C en game node if he c en game node corresponds to a winning situation for one of the players, write WINS[PLAYER???]| MINIMAX = ?????]

Question 4: Place the MINIMAX values for each non-repeated game state in the drawing of the game tree based on the output of your code. (**15 points**)

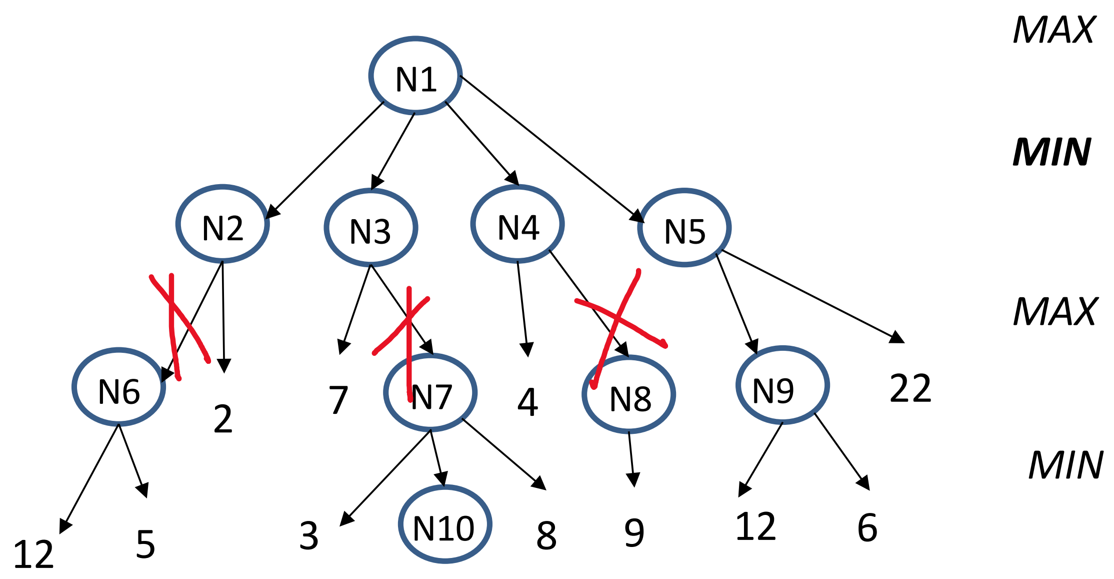
<u>Extra-credit:</u>

<u>Programming Assignment for Problem 1:</u>

C. If you allow an additional action for the game, namely that any player can jump over an occupied square, and represent this new action as : Jump+Down, Jump+Left, Jump+Right or Jump+Down, modify your program to generate the new game tree. Produce the output in the same format as when doing assignment A. (**10 points**) Comment on the difference between the game trees: Which player won faster in which variant of the game??? Are the repeated states any different? (**10 points**)

PROBLEM 2: Alpha-Beta Search (**26 points**)

Find the move ordering (killer move) that allows you to prune the largest number of



You will receive **10 points** if you find the killer move and explain why it is a killer move. You should produce a trace of alpha-beta pruning using the format showing how the killer move operates (**10 points)**:

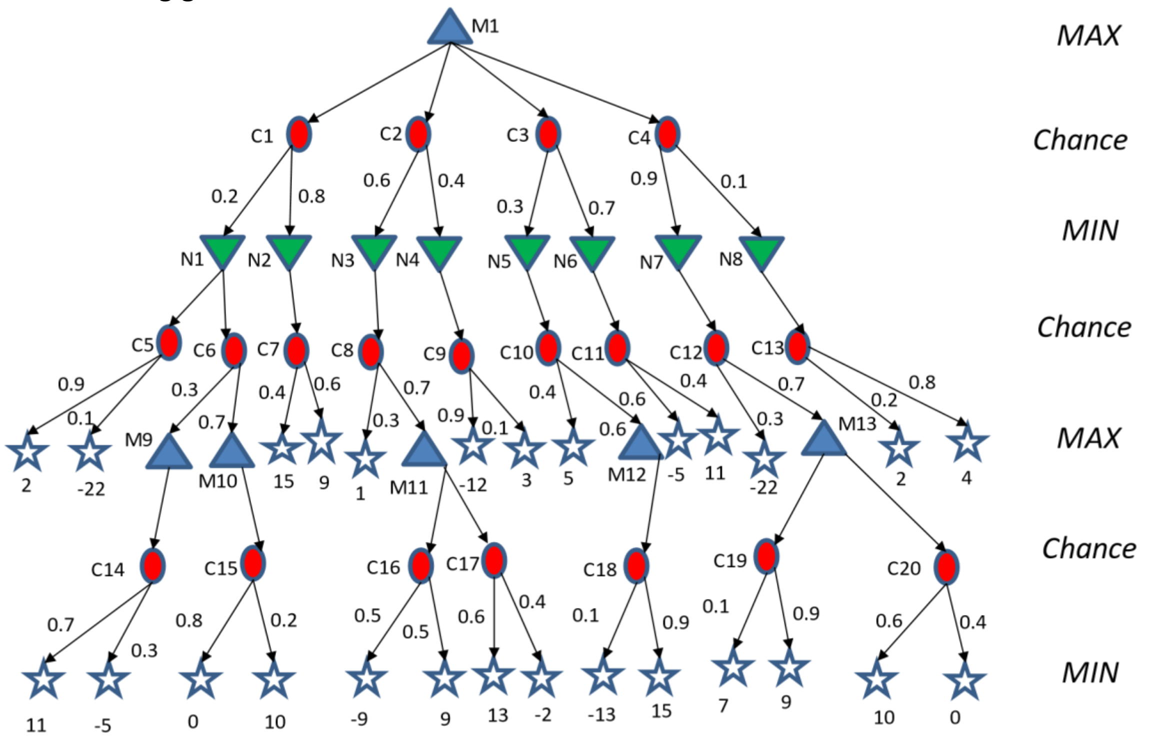
N1:	alpha = -f	beta = +f	maxvalue=???
NX??:	alpha = ??	beta = ??	minvalue=???
NY:	alpha = ???	beta = ??	maxvalue=??
???:	alpha = ???	beta = ??	minvalue=??
….			

Indicate on the Search Tree which subtrees shall be pruned and if it is alpha or beta pruning (**6 points)** when you indicate correctly which subtrees shall be pruned.

Answer:
Killer Move = N5 Expansion first
N9 = MAX(12,6) = 12
N5 = 12
Now if expansion of N4 means we know N4 < 4, we can prune, as N4 being Min will never be greater than 4
Similarly N3 < 7 we can prune it and N2 < 2 so we can prune it.

N1:	alpha = -infinity	beta= +infinity	max = -infinity	
N5:	alpha = -infinity	beta= +infinity	min = +infinity	
N9:	alpha = -infinity	beta= +infinity	max = -infinity	
12:	Return 12			
N9:	alpha=12	beta=+infinity	max=12	
6:	Return 6			
N9:	alpha=12	beta=+infinity	max=12	
N5:	alpha= -infinity	beta=12	min=12	
22:	Return 22			
N5:	alpha= -infinity	beta=12	min=12	
N1:	alpha=12	beta=+infinity	max=12	
N2:	alpha=12	beta=+infinity	min=+infinity	
2:	Return 2			
N2:	alpha=12	beta=2	min=2	**ALPHA-PRUNE**
N1:	alpha=12	beta=+infinity	max=12	
N3:	alpha=12	beta=+infinity	min=+infinity	
7:	Return 7			
N3:	alpha=12	beta=7	min=7	**ALPHA-PRUNE**
N1:	alpha=12	beta=+infinity	max=12	
N4:	alpha=12	beta=+infinity	min=+infinity	
4: Return 4				
N4:	alpha=12	beta=4	min=4	**ALPHA-PRUNE**
N1:	alpha=12	beta=+infinity	max=12	

PROBLEM 3: Chance Games (**14 points**) Given the following game tree:



Compute the *Expectiminimax* value in the following nodes, making sure you show how you computed the value:

Calculating the Expectiminimax Values for **Chance** nodes (C14 – C20):

Expect-MiniMax(C14) = 0.7(11) + 0.3(-5) = 7.7 – 1.5 = 6.2
Expect-MiniMax(C15) = 0.8(0) + 0.2(10) = 0 + 2.0 = 2.0
Expect-MiniMax(C16) = 0.5(-9) + 0.5(9) = -4.5 + 4.5 = 0
Expect-MiniMax(C17) = 0.6(13) + 0.4(-2) = 7.8 – 0.8 = 7.0
Expect-MiniMax(C18) = 0.1(-13) + 0.9(15) = -1.3 + 13.5 = 12.2
Expect-MiniMax(C19) = 0.1(7) + 0.9(9) = 0.7 + 8.1 = 8.8
Expect-MiniMax(C20) = 0.6(10) + 0.4(0) = 6.0 + 0 = 6.0
Expect-MiniMax(M9) = max (C14) = max (6.2) = 6.2
Expect-MiniMax(M10) = max (C15) = max (2.0) = 2.0
Expect-MiniMax(M11) = max (C16, C17) = max (0, 7.0) = 7.0
Expect-MiniMax(M12) = max (C18) = max (12.2) = 12.2
Expect-MiniMax(M13) = max (C19, C20) = max (8.8, 6.0) = 8.8

Calculating the Expectiminimax Values for Chance nodes (C5 – C13):

Expect-MiniMax(C5) = 0.9(2) + 0.1(-22) = 1.8 – 2.2 = -0.4

Expect-MiniMax(C6) = 0.3(M9) + 0.7(M10) = 0.3(6.2) + 0.7(2.0) = 1.86 + 1.4 = 3.26

Expect-MiniMax(C7) = 0.4(15) + 0.6(9) = 6.0 + 5.4 = 11.4

Expect-MiniMax(C8) = 0.3(1) + 0.7(M11) = 0.3(1) + 0.7(7.0) = 0.3 + 4.9 = 5.2

Expect-MiniMax(C9) = 0.9(-12) + 0.1(3) = -10.8 + 0.3 = -10.5

Expect-MiniMax(C10) = 0.4(5) + 0.6(M12) = 0.4(5) + 0.6(12.2) = 2.0 + 7.32 = 9.32

Expect-MiniMax(C11) = 0.6(-5) + 0.4(11) = -3.0 + 4.4 = 1.4

Expect-MiniMax(C12) = 0.3(-22) + 0.7(M13) = 0.3(-22) + 0.7(8.8) = -6.6 + 6.16 = -0.44

Expect-MiniMax(C11) = 0.2(2) + 0.8(4) = 0.4 + 3.2 = 3.6

Expect-MiniMax(N1) = min (C5, C6) = min (-0.4, 3.26) = -0.4

Expect-MiniMax(N2) = min (C7) = min (11.4) = 11.4

Expect-MiniMax(N3) = min (C8) = min (5.2) = 5.2

Expect-MiniMax(N4) = min (C9) = min (-10.5) = -10.5

Expect-MiniMax(N5) = min (C10) = min (9.32) = 9.32

Expect-MiniMax(N6) = min (C11) = min (1.4) = 1.4

Expect-MiniMax(N7) = min (C12) = min (-0.44) = -0.44

Expect-MiniMax(N8) = min (C13) = min (3.6) = 3.6

Calculating the Expectiminimax Values for Chance nodes (C1 – C4):

Expect-MiniMax(C1) = 0.2(N1) + 0.8(N2) = 0.2(-0.4) + 0.8(11.4) = -0.08 + 9.12 = 9.04

Expect-MiniMax(C2) = 0.6(N3) + 0.4(N4) = 0.6(5.2) + 0.4(-10.5) = 3.12 – 4.2 = -1.08

Expect-MiniMax(C3) = 0.3(N5) + 0.7(N6) = 0.3(9.32) + 0.7(1.4) = 2.796 + 0.98 = 3.776

Expect-MiniMax(C4) = 0.9(N7) + 0.1(N8) = 0.9(-0.44) + 0.1(3.6) = -0.396 + 0.36 = -0.036

Expect-MiniMax(M1) = max (C1, C2, C3, C4) = max (9.04, -1.08, 3.776, -0.036) = 9.04

Software Engineering (includes documentation for your programming assignments)

Your README file must include the following:

x Your name and email address. x *Homework number* for this class (AI CS6364), and the *number of the problem* it solves.

x A description of every file for your solution, the programming language used, supporting files from the aima code used, etc. x How your code operates, in detail.

x A description of special features (or limitations) of your code.

Within Code Documentation:

x Methods/functions/procedures should be documented in a meaningful way. This can mean expressive function/variable names as well as explicit documentation.

x Informative method/procedure/function/variable names.

x Efficient implementation x
Don't hardcode variable values, etc