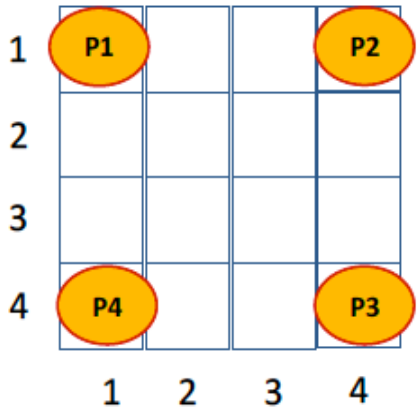
CS6364 - Artificial Intelligence
Fall 2020

Homework 2

Name: Rajarshi Chattopadhyay
Net ID: RXC170010
Email: rxc170010@utdallas.edu

PROBLEM 1: 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**)

Answer 1: Representing the game-
The game consists of a problem space, an initial state, and a set of goal states. A problem space is a mathematical abstraction in a form of a tree:

- the root represents current state
- **each node represents the state of the game**
- edges represent moves
- leaves represent final states

Each node of the game that represents the state of the game can be visualized as follows:

0,0	*0,1*	*0,2*	*0,3*
1,0	*1,1*	*1,2*	*1,3*
2,0	*2,1*	*2,2*	*2,3*
3,0	*3,1*	*3,2*	*3,3*

We **represent each node of the game as a 2D array having 4 rows (0:3) and 4 columns (0:3)**

Each item of the 2D array has value None for empty positions and a value 1/2/3/4 corresponding to the position of player 1/2/3/4 respectively.

Example,

```
node = [[1, None, None, 2], [None, None, None, None], [None, None, None,
None], [4, None, None, 3]]
```

The **Initial Node** of the game can be visualized as:

P1			P2
P4			P3

```
initial_node = [[1, None, None, 2], [None, None, None, None], [None, None,
None, None], [4, None, None, 3]]
```

```
1          None       None       2
None       None       None       None
None       None       None       None
4          None       None       3
```

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!

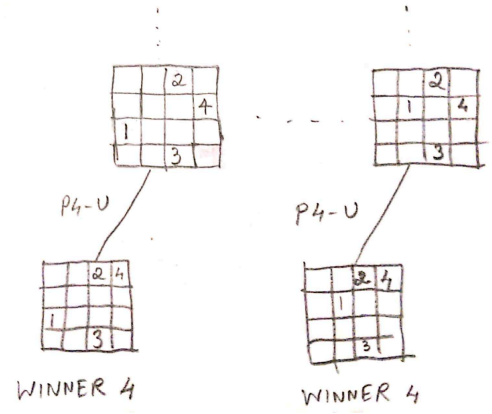
Program for Problem 1A: ***gen_tree.py***

Output for Problem 1A: ***output.txt***

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.

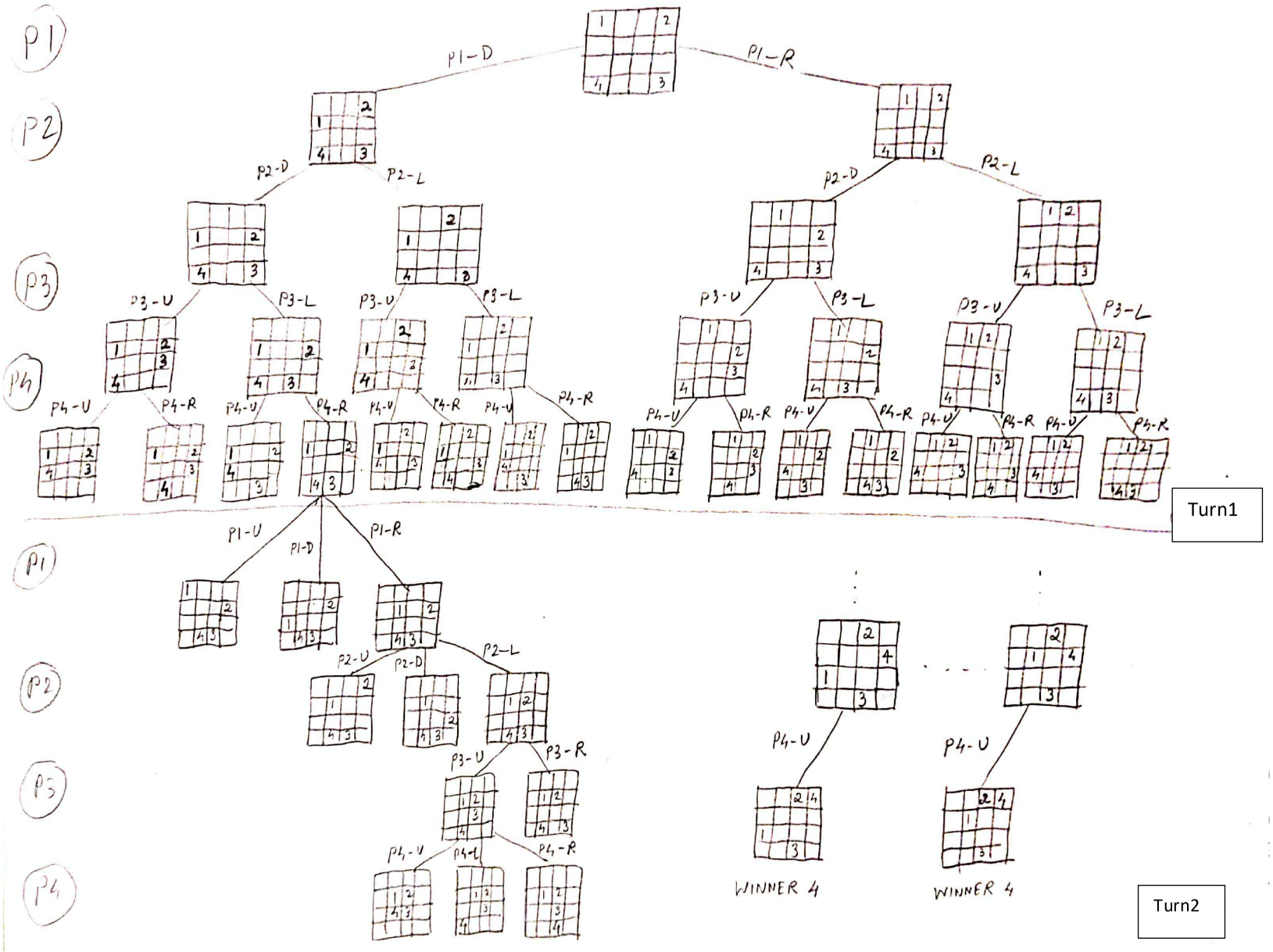
Answer 2:

Example of P4 winning



The game tree covering the first turn and part of the second turn is drawn.

It covers all the possible moves played by all 4 players during their first move and part of the possible moves during their second move.


P1
P2
P3
P4
P1-D
P1-R
P2-D
P2-L
P3-U
P3-L
P4-U
P4-R
Turn1
P1-U
P1-D
P1-R
P2-U
P2-D
P2-L
P3-U
P3-R
P4-U
P4-L
P4-R
WINNER 4
WINNER 4
Turn2

First 31 nodes generated in the first level (first turn played by each player)

[Current_player=1 | Father_node=None | Action=None | Current_node=[[1, None, None, 2], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] |]

[Current_player=1 | Father_node=[[1, None, None, 2], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=down | Current_node=[[None, None, None, 2], [1, None, None, None], [None, None, None, None], [4, None, None, 3]] |]
[Current_player=2 | Father_node=[[None, None, None, 2], [1, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=down | Current_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [4, None, None, 3]] |]
[Current_player=3 | Father_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [4, None, None, 3]] | Action=up | Current_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, 3], [4, None, None, None]] |]
[Current_player=4 | Father_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, 3], [4, None, None, None]] | Action=up | Current_node=[[None, None, None, None], [1, None, None, 2], [4, None, None, 3], [None, None, None, None]] |]
[Current_player=4 | Father_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, 3], [4, None, None, None]] | Action=right | Current_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, 3], [None, 4, None, None]] |]
[Current_player=3 | Father_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [4, None, None, 3]] | Action=left | Current_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [4, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [4, None, 3, None]] | Action=up | Current_node=[[None, None, None, None], [1, None, None, 2], [4, None, None, None], [None, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [4, None, 3, None]] | Action=right | Current_node=[[None, None, None, None], [1, None, None, 2], [None, None, None, None], [None, 4, 3, None]] |]
[Current_player=2 | Father_node=[[None, None, None, 2], [1, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=left | Current_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [4, None, None, 3]] |]
[Current_player=3 | Father_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=up | Current_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, 3], [4, None, None, None]] |]
[Current_player=4 | Father_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, 3], [4, None, None, None]] | Action=up | Current_node=[[None, None, 2, None], [1, None, None, None], [4, None, None, 3], [None, None, None, None]] |]
[Current_player=4 | Father_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, 3], [4, None, None, None]] | Action=right | Current_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, 3], [None, 4, None, None]] |]
[Current_player=3 | Father_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=left | Current_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [4, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [4, None, 3, None]] | Action=up | Current_node=[[None, None, 2, None], [1, None, None, None], [4, None, None, None], [None, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [4, None, 3, None]] | Action=right | Current_node=[[None, None, 2, None], [1, None, None, None], [None, None, None, None], [None, 4, 3, None]] |]

[Current_player=1 | Father_node=[[1, None, None, 2], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=right | Current_node=[[None, 1, None, 2], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] |]
[Current_player=2 | Father_node=[[None, 1, None, 2], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=down | Current_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [4, None, None, 3]] |]
[Current_player=3 | Father_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [4, None, None, 3]] | Action=up | Current_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, 3], [4, None, None, None]] |]
[Current_player=4 | Father_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, 3], [4, None, None, None]] | Action=up | Current_node=[[None, 1, None, None], [None, None, None, 2], [4, None, None, 3], [None, None, None, None]] |]
[Current_player=4 | Father_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, 3], [4, None, None, None]] | Action=right | Current_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, 3], [None, 4, None, None]] |]
[Current_player=3 | Father_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [4, None, None, 3]] | Action=left | Current_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [4, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [4, None, 3, None]] | Action=up | Current_node=[[None, 1, None, None], [None, None, None, 2], [4, None, None, None], [None, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [4, None, 3, None]] | Action=right | Current_node=[[None, 1, None, None], [None, None, None, 2], [None, None, None, None], [None, 4, 3, None]] |]
[Current_player=2 | Father_node=[[None, 1, None, 2], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=left | Current_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] |]
[Current_player=3 | Father_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=up | Current_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, 3], [4, None, None, None]] |]
[Current_player=4 | Father_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, 3], [4, None, None, None]] | Action=up | Current_node=[[None, 1, 2, None], [None, None, None, None], [4, None, None, 3], [None, None, None, None]] |]
[Current_player=4 | Father_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, 3], [4, None, None, None]] | Action=right | Current_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [None, None, None, 3], [None, 4, None, None]] |]
[Current_player=3 | Father_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [4, None, None, 3]] | Action=left | Current_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [4, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [4, None, 3, None]] | Action=up | Current_node=[[None, 1, 2, None], [None, None, None, None], [4, None, None, None], [None, None, 3, None]] |]
[Current_player=4 | Father_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [4, None, 3, None]] | Action=right | Current_node=[[None, 1, 2, None], [None, None, None, None], [None, None, None, None], [None, 4, 3, None]] |]

Number of levels covered: 1
Number of nodes generated: 31

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:

- When Player 1 wins, Player 2 receives utility=10, Player 3 receives utility=30, and Player 4 receives utility=10;
- When Player 2 wins, Player 1 receives utility=100, Player 3 receives utility=150, and Player 4 receives utility=200;
- When Player 3 wins, Player 1 receives utility=150, Player 2 receives utility=200, and Player 4 receives utility=300;
- 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.

Programming Assignment for Problem 1: (**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* = …| Father node (if not initial node) = …| Action= ????| Current game node =…| if the current game node corresponds to a winning situation for one of the players, write WINS[PLAYER???]| MINIMAX = ?????]

Answer 3:

Program for Problem 1B: ***gen_tree_minimax.py***

Output for Problem 1: ***output_minimax.txt***

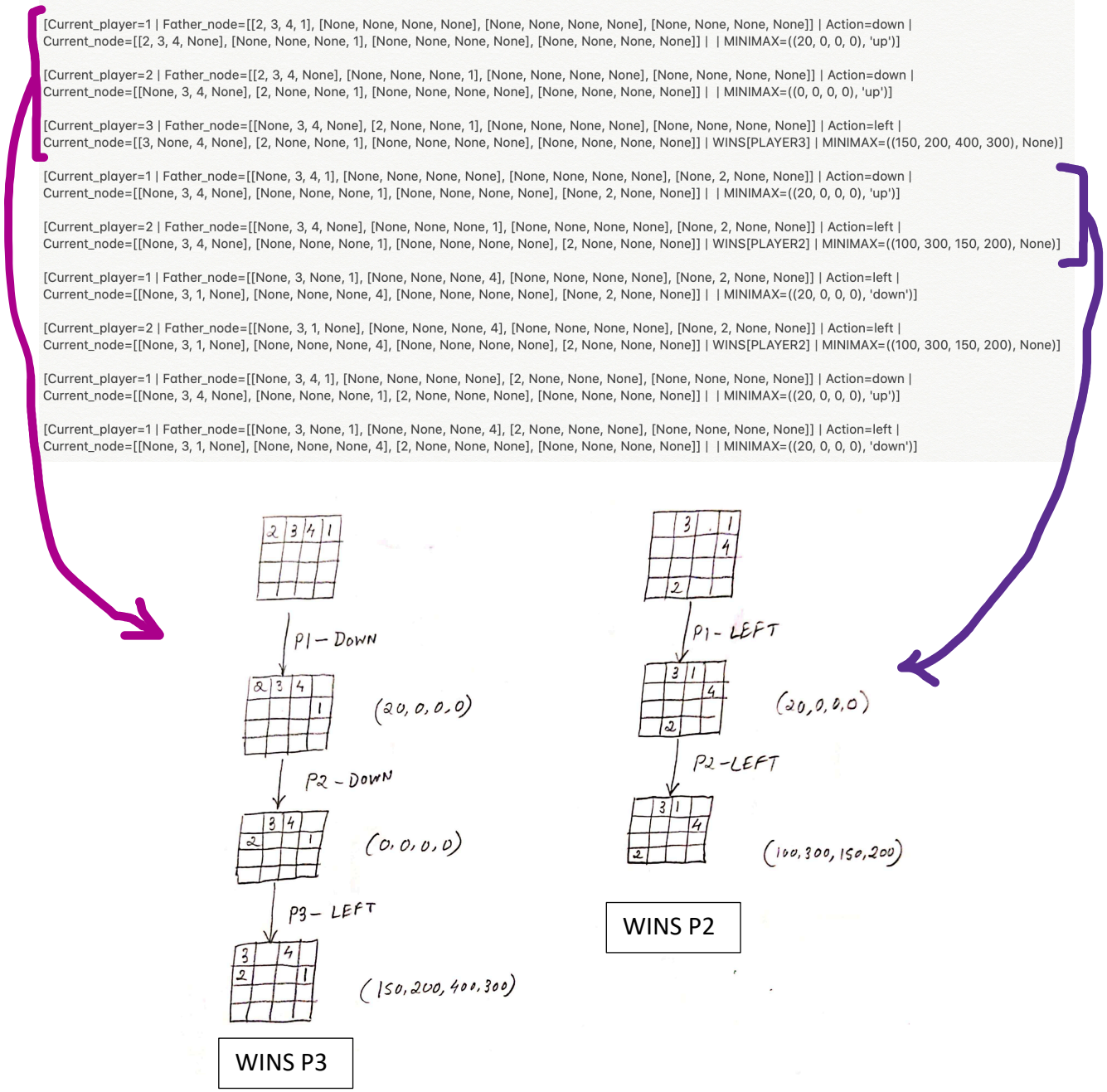
NOTE: In this program, a depth of 6 has been used to calculate bounded minimax values for a node. This was done keeping in mind the capabilities of the development infrastructure.

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**)

Answer 4:

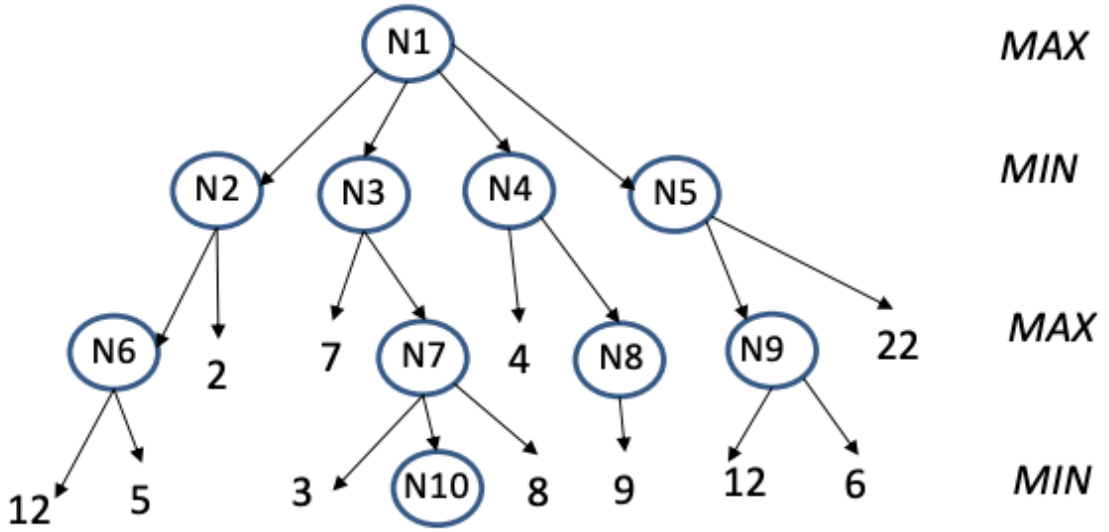
Some example game nodes having win and normal states are as below:

```
[Current_player=2 | Father_node=[[None, None, None, None], [None, None, None, 1], [None, None, None, None], [None, 2, 3, 4]] | Action=left |
Current_node=[[None, None, None, None], [None, None, None, 1], [None, None, None, None], [2, None, 3, 4]] | WINS[PLAYER2] | MINIMAX=((100, 300, 150, 200), None)]

[Current_player=1 | Father_node=[[None, None, None, None], [None, None, 1, None], [2, None, None, None], [None, None, 3, 4]] | Action=right |
Current_node=[[None, None, None, None], [None, None, None, 1], [2, None, None, None], [None, None, 3, 4]] |  | MINIMAX=((0, 0, 0, 0), 'up')]

[Current_player=1 | Father_node=[[2, 3, 4, 1], [None, None, None, None], [None, None, None, None], [None, None, None, None]] | Action=down |
Current_node=[[2, 3, 4, None], [None, None, None, 1], [None, None, None, None], [None, None, None, None]] |  | MINIMAX=((20, 0, 0, 0), 'up')]

[Current_player=2 | Father_node=[[2, 3, 4, None], [None, None, None, 1], [None, None, None, None], [None, None, None, None]] | Action=down |
Current_node=[[None, 3, 4, None], [2, None, None, 1], [None, None, None, None], [None, None, None, None]] |  | MINIMAX=((0, 0, 0, 0), 'up')]

[Current_player=3 | Father_node=[[None, 3, 4, None], [2, None, None, 1], [None, None, None, None], [None, None, None, None]] | Action=left |
Current_node=[[3, None, 4, None], [2, None, None, 1], [None, None, None, None], [None, None, None, None]] | WINS[PLAYER3] | MINIMAX=((150, 200, 400, 300), None)]

[Current_player=1 | Father_node=[[None, 3, 4, 1], [None, None, None, None], [None, None, None, None], [None, 2, None, None]] | Action=down |
Current_node=[[None, 3, 4, None], [None, None, None, 1], [None, None, None, None], [None, 2, None, None]] |  | MINIMAX=((20, 0, 0, 0), 'up')]

[Current_player=2 | Father_node=[[None, 3, 4, None], [None, None, None, 1], [None, None, None, None], [None, 2, None, None]] | Action=left |
Current_node=[[None, 3, 4, None], [None, None, None, 1], [None, None, None, None], [2, None, None, None]] | WINS[PLAYER2] | MINIMAX=((100, 300, 150, 200), None)]

[Current_player=1 | Father_node=[[None, 3, None, 1], [None, None, None, 4], [None, None, None, None], [None, 2, None, None]] | Action=left |
Current_node=[[None, 3, 1, None], [None, None, None, 4], [None, None, None, None], [None, 2, None, None]] |  | MINIMAX=((20, 0, 0, 0), 'down')]

[Current_player=2 | Father_node=[[None, 3, 1, None], [None, None, None, 4], [None, None, None, None], [None, 2, None, None]] | Action=left |
Current_node=[[None, 3, 1, None], [None, None, None, 4], [None, None, None, None], [2, None, None, None]] | WINS[PLAYER2] | MINIMAX=((100, 300, 150, 200), None)]

[Current_player=1 | Father_node=[[None, 3, 4, 1], [None, None, None, None], [2, None, None, None], [None, None, None, None]] | Action=down |
Current_node=[[None, 3, 4, None], [None, None, None, 1], [2, None, None, None], [None, None, None, None]] |  | MINIMAX=((20, 0, 0, 0), 'up')]

[Current_player=1 | Father_node=[[None, 3, None, 1], [None, None, None, 4], [2, None, None, None], [None, None, None, None]] | Action=left |
Current_node=[[None, 3, 1, None], [None, None, None, 4], [2, None, None, None], [None, None, None, None]] |  | MINIMAX=((20, 0, 0, 0), 'down')]
```



PROBLEM 2: Alpha-Beta Search (26 points)

Find the move ordering (killer move) that allows you to prune the largest number of subtrees when using alpha-beta pruning on the following game tree:

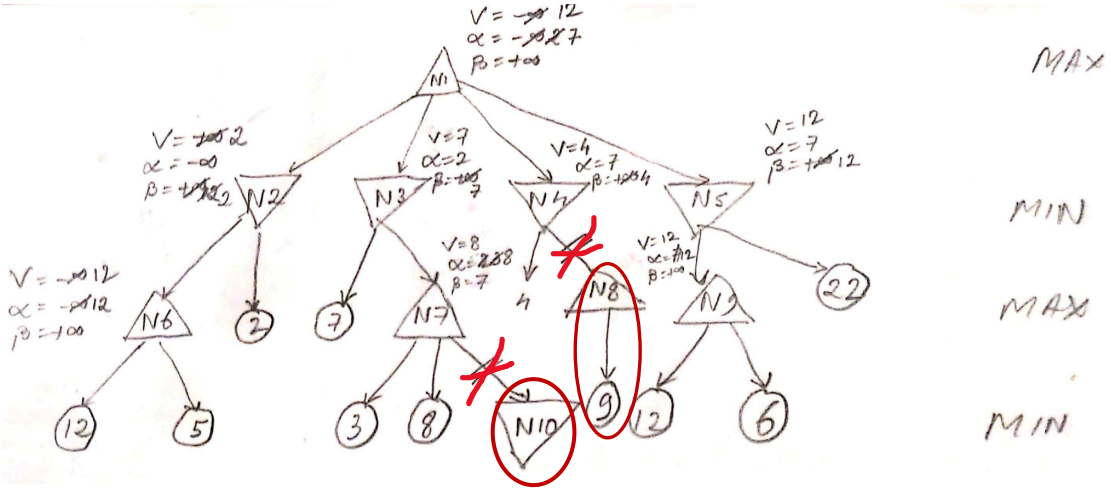


You will receive **10 points** if you find the killer move and explain why it is a killer move. You should produce a trace of alpha-beta pruning using the format showing how the killer move operates (**10 points**):

N1:	alpha = -∞	beta = +∞	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.

Move Ordering 1: **Expanding via N2 first**



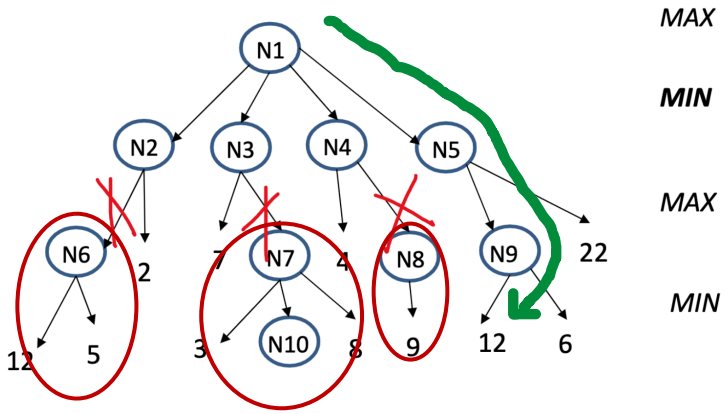
Trace of alpha-beta pruning using DFS: -

Node	Alpha	Beta	value	function
N1	-inf	+inf	maxvalue= -inf	
N2	-inf	+inf	minvalue= +inf	
N6	-inf	+inf	maxvalue= +inf	
N6	Max(12, 5) = 12	+inf	maxvalue= 12	Max(12, 5)
N2	-inf	Min(12, 2) = 2	minvalue= 2	Min(N6, 2)
N1	Max(-inf, 2) = 2	+inf	maxvalue= 2	Max(-inf, N2)
N3	2	+inf	minvalue= +inf	
N3	2	Min(+inf, 7) = 7	minvalue= 7	Min(+inf, 7)
N7	2	7	maxvalue= 2	
N7	Max(2, 3) = 3	7	maxvalue= 3	Max(2, 3)
N7	Max(3, 8) = 8	7	maxvalue= 8	Max(3, 8)
N7	8	7	maxvalue= 8	beta prune N10, as alpha8>beta7
N3	2	Min(7, 8) = 7	minvalue= 7	Min(7, N7)
N1	Max(2, 7) = 7	+inf	maxvalue=7	Max(N2, N3)
N4	7	+inf	minvalue= +inf	
N4	7	Min(+inf, 4) = 4	minvalue= 4	Min(+inf, 4)
N4	7	4	minvalue= 4	alpha prune N8, as alpha7>beta4
N1	Max(7, 4) = 7	+inf	maxvalue=7	Max(N2, N3, N4)
N5	7	+inf	minvalue= +inf	
N9	7	+inf	maxvalue= 7	
N9	Max(7, 12) = 12	+inf	maxvalue= 12	Max(7, 12)
N9	Max(12, 6) = 12	+inf	maxvalue= 12	Max(12, 6)
N5	7	Min(+inf, 12) = 12	minvalue= 12	Min(+inf, 12)
N5	7	Min(12, 22) = 12	minvalue= 12	Min(12, 22)
N1	Max(7, 12) = 7	+inf	maxvalue= 12	Max(N2, N3, N4, N5)
N1	**7**	**+inf**	**12**	**Max(2, 7, 4, 12)**

The subtrees that will be pruned are:

- **N10** = *Beta pruning* – Search stopped below N7 MAX node having alpha=8 greater than beta=7.
- **N8** and rest = *Alpha pruning* – Search stopped below N4 MIN node having alpha=7 greater than beta=4.

Move Ordering 2: **Expanding via N5 first**



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	

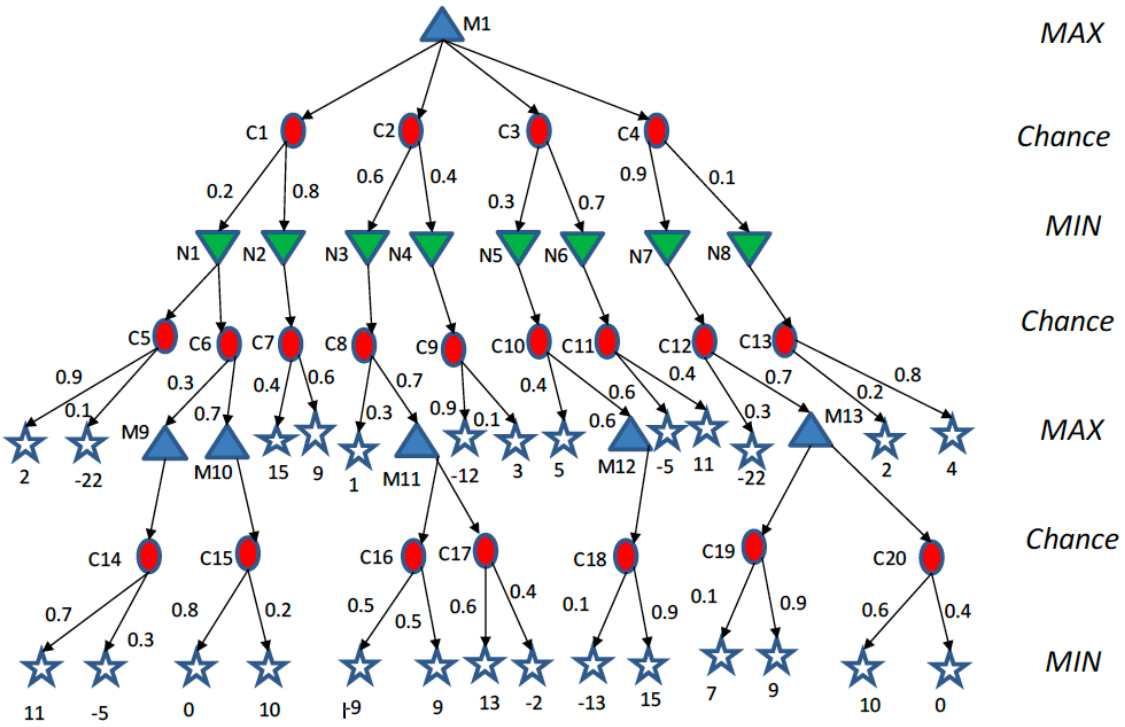
We get maximum number of prunes here. So, this is the best expansion move.

The killer move is N1 → N5 → N9 → 12

At N1, which is a MAX node, it will go to the node with the max value out of N2, N3, N4, N5, i.e. max(2, 7, 4, 12). So, the next move should be to get to N5.
At N5, which is a MIN node, the next move that will be possible is to the min out of N9 and 22.
At N9, which is a MAX node, its value will be max(12, 6) i.e. 12.
Hence, at N5, the node that will be selected will be min(N9, 22), i.e. min(12, 22), i.e. 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:

(**1 point**) In M1:

(**1 point**) In M9:

(**1 point**) In M10:

(**1 point**) In M11:

(**1 point**) In M12:

(**1 point**) In M13:

(**1 point**) In N1:

(**1 point**) In N2:

(**1 point**) In N3:

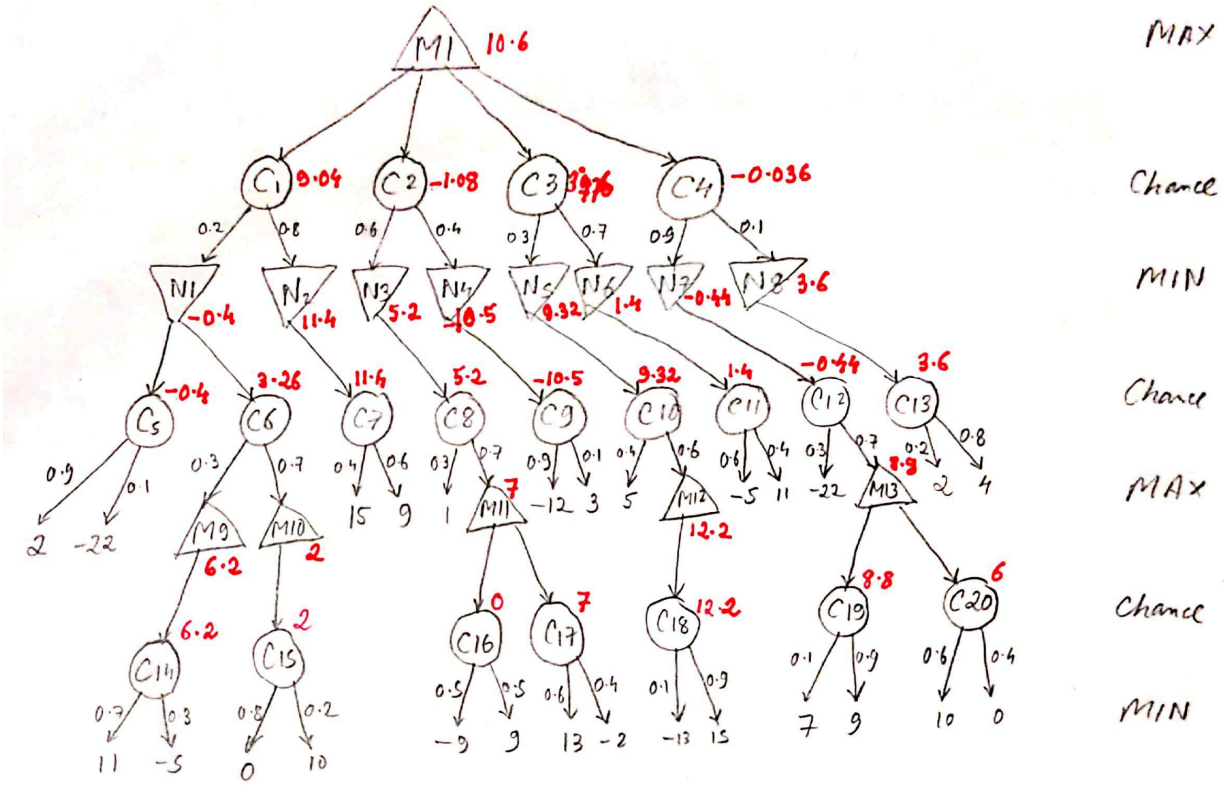
(**1 point**) In N4:

(**1 point**) In N5:

(**1 point**) In N6:

(**1 point**) In N7:

(**1 point**) In N8:



$C20 = (0.6\times10)+(0.4\times0) = 6$
$C19 = (0.1\times7)+(0.9\times9) = 8.8$
$C18 = (0.1\times-13)+(0.9\times15) = 12.2$
$C17 = (0.6\times13)+(0.4\times-2) = 7$
$C16 = (0.5\times-9)+(0.5\times9) = 0$
$C15 = (0.8\times0)+(0.2\times10) = 2$
$C14 = (0.7\times11)+(0.3\times-5) = 6.2$

$M19 = \max(C19, C20) = \max(8.8, 6) = 8.8$
$M12 = C18 = 12.2$
$M11 = \max(C16, C17) = \max(0, 7) = 7$
$M10 = C15 = 2$
$M9 = C14 = 6.2$

$C13 = (0.2\times2)+(0.8\times4) = 3.6$
$C12 = (0.3\times-22)+(0.7\times8.8) = -0.44$
$C11 = (0.6\times-5)+(0.4\times11) = 1.4$
$C10 = (0.1\times5)+(0.6\times12.2) = 9.32$
$C9 = (0.9\times-12)+(0.1\times3) = -10.5$
$C8 = (0.3\times1)+(0.7\times7) = 5.2$
$C7 = (0.4\times15)+(0.6\times9) = 11.4$
$C6 = (0.3\times6.2)+(0.7\times2) = 3.26$
$C5 = (0.9\times2)+(0.1\times-22) = -0.4$

$N8 = C13 = 3.6$
$N7 = C12 = -0.44$
$N6 = C11 = 1.4$
$N5 = C10 = 9.32$
$N4 = C9 = -10.5$
$N3 = C8 = 5.2$
$N2 = C7 = 11.4$
$N1 = \min(C5, C6) = \min(-0.4, 3.26) = -0.4$

$C4 = (0.9\times-0.44)+(0.1\times3.6) = -0.036$
$C3 = (0.3\times9.32)+(0.7\times1.4) = 3.776$
$C2 = (0.6\times5.2)+(0.4\times-10.5) = -1.08$
$C1 = (0.2\times-0.4)+(0.8\times11.4) = 9.04$

$M1 = \max(C1, C2, C3, C4) = \max(9.04, -1.08, 10.6, -0.036) = 10.6$

C20 = 6
C19 = 8.8
C18 = 12.2
C17 = 7
C16 = 0
C15 = 2
C14 = 6.2

M13 = max(C19, C20) = 8.8
M12 = C18 = 12.2
M11 = max(C16, C17) = 7
M10 = C15 = 2
M9 = C14 = 6.2

C13 = 3.6
C12 = -0.44
C11 = 1.4
C10 = 9.32
C9 = -10.5
C8 = 5.2
C7 = 11.4
C6 = 3.26
C5 = -0.4

N8 = C13 = 3.6
N7 = C12 = -0.44
N6 = C11 = 1.4
N5 = C10 = 9.32
N4 = C9 = -10.5
N3 = C8 = 5.2
N2 = C7 = 11.4
N1 = min(C5, C6) = -0.4

C4 = -0.036
C3 = 3.776
C2 = -1.08
C1 = 9.04

M1 = max(C1, C2, C3, C4) = 9.04

Therefore,

In M1: 9.04
In M9: 6.2
In M10: 2
In M11: 7
In M12: 12.2
In M13: 8.8
In N1: -0.4
In N2: 11.4
In N3: 5.2
In N4: -10.5
In N5: 9.32
In N6: 1.4
In N7: -0.44
In N8: 3.6