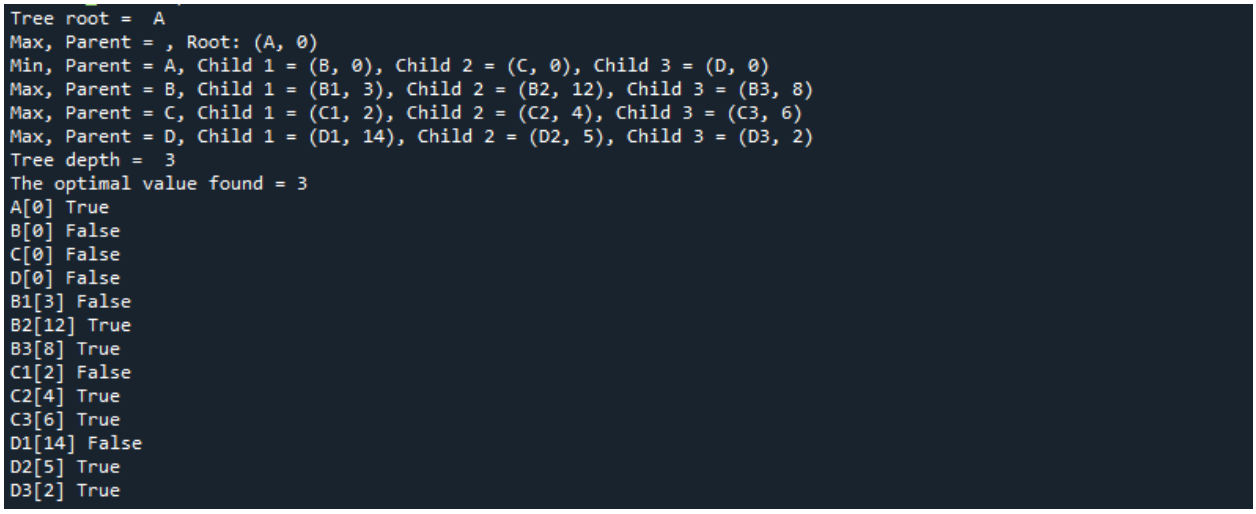
Anshul Chauhan & James Rozsypal
Project 3 Report
Date: 8/4/2021
CECS 451

- For this assignment we were tasked with creating a program that utilizes the minimax algorithm with alpha-beta pruning in order to find the optimal path through a tree. In order to do this we had to write a function called parse() to pull the data from the tree.txt files that were provided and a function called minimax() which performs the minimax algorithm with alpha-beta pruning.
- The parse() method works by reading each line of a tree text file using the python csv library. Each row of a line pulled from a tree's csv text file contains the nodes of the tree. If the line being read is the first line in the file then we know it contains the root node so we only have to get the parent, id, and value of the root node. After getting the root node we loop through the rest of the lines in the file which contain the children nodes, we pull the parent, id, and value of each child node until there are no more children to be read. After completing these steps a whole tree is parsed from a csv text file and saved as a Tree().
- The minimax() algorithm works by exploring the tree in a depth-first manner, performing moves until reaching a terminal state, and sending this value back up the tree. It calculates each node of the tree by selecting either the move with the maximum point, or the minimum point. The Alpha-beta pruning to a standard minimax calculation returns a similar move as the standard calculation does, yet it eliminates every one of the nodes which are not actually influencing an official choice yet making calculation moderate. Subsequently by pruning these nodes, it makes the algorithm quick.



```
Tree root =  A
Max, Parent = , Root: (A, 0)
Min, Parent = A, Child 1 = (B, 0), Child 2 = (C, 0)
Max, Parent = B, Child 1 = (D, 0), Child 2 = (E, 0)
Max, Parent = C, Child 1 = (F, 0), Child 2 = (G, 0)
Min, Parent = D, Child 1 = (H, 0), Child 2 = (I, 0)
Min, Parent = E, Child 1 = (J, 0), Child 2 = (K, 0)
Min, Parent = F, Child 1 = (L, 0), Child 2 = (M, 0)
Min, Parent = G, Child 1 = (N, 0), Child 2 = (O, 0)
Max, Parent = H, Child 1 = (H1, 10), Child 2 = (H2, 5)
Max, Parent = I, Child 1 = (I1, 7), Child 2 = (I2, 11)
Max, Parent = J, Child 1 = (J1, 12), Child 2 = (J2, 8)
Max, Parent = K, Child 1 = (K1, 9), Child 2 = (K2, 8)
Max, Parent = L, Child 1 = (L1, 5), Child 2 = (L2, 12)
Max, Parent = M, Child 1 = (M1, 11), Child 2 = (M2, 12)
Max, Parent = N, Child 1 = (N1, 9), Child 2 = (N2, 8)
Max, Parent = O, Child 1 = (O1, 7), Child 2 = (O2, 10)
Tree depth =  5
The optimal value found = 8
A[0] True
B[0] False
C[0] False
D[0] False
E[0] True
F[0] False
G[0] True
H[0] False
I[0] False
J[0] True
K[0] True
L[0] False
M[0] False
N[0] True
O[0] True
H1[10] False
H2[5] True
I1[7] False
I2[11] True
J1[12] True
J2[8] True
K1[9] True
K2[8] True
L1[5] False
L2[12] True
M1[11] False
M2[12] True
N1[9] True
N2[8] True
O1[7] True
O2[10] True
```