#### Jonathan Liao

#### CS 305 HW 5 Report

**1. Questions (include these in your write-up):**

1a. Meets all specifications. I am a little concerned about my AI however: the AI isn’t very smart sometimes, I feel as though my tree may not have been scored correctly. Sometimes the AI will be extremely clever, and other times it will play really dumb.

1b. COMPUTER WINS:

[CAMPUS\liao18@egr1 hm5]$ ./game

Computer goes first.

[X] [ ] [ ]

[ ] [ ] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 1

Enter column (0 to 2): 1

[X] [X] [ ]

[ ] [O] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 2

Enter column (0 to 2): 2

Computer wins.

[X] [X] [X]

[ ] [O] [ ]

[ ] [ ] [O]

Thanks for playing.

[CAMPUS\liao18@egr1 hm5]$

TIE GAME:

[CAMPUS\liao18@egr1 hm5]$ ./game

Human player goes first.

[ ] [ ] [ ]

[ ] [ ] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 0

Enter column (0 to 2): 0

[O] [X] [ ]

[ ] [ ] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 2

Enter column (0 to 2): 2

[O] [X] [ ]

[ ] [X] [ ]

[ ] [ ] [O]

Enter row (0 to 2): 2

Enter column (0 to 2): 1

[O] [X] [ ]

[ ] [X] [ ]

[X] [O] [O]

Enter row (0 to 2): 0

Enter column (0 to 2): 2

[O] [X] [O]

[ ] [X] [X]

[X] [O] [O]

Enter row (0 to 2): 1

Enter column (0 to 2): 0

Tie game.

[O] [X] [O]

[O] [X] [X]

[X] [O] [O]

Thanks for playing.

[CAMPUS\liao18@egr1 hm5]$

HUMAN WINS

[CAMPUS\liao18@egr1 hm5]$ ./game

Human player goes first.

[ ] [ ] [ ]

[ ] [ ] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 1

Enter column (0 to 2): 1

[X] [ ] [ ]

[ ] [O] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 0

Enter column (0 to 2): 1

[X] [O] [X]

[ ] [O] [ ]

[ ] [ ] [ ]

Enter row (0 to 2): 2

Enter column (0 to 2): 1

Congratulations! You win!

[X] [O] [X]

[ ] [O] [ ]

[ ] [O] [ ]

Thanks for playing.

[CAMPUS\liao18@egr1 hm5]$

2. (1 pt) Draw the non-null children and their non-null descendants of the following game tree node:

board:

o x x

o x

x o

score: ‘T’

player: ‘X’

board: board:

o x x o x x

o x o o x

x o o x o

score: ‘X’ score: ‘O’

player: ‘O’ player: ‘O’

board:

o x x

o x x

x o o

score: ‘X’

player: ‘X’

3. Copied from the UNIX terminal with code modified to print out tree. Assuming O (player) goes first

[O] [ ] [ ]

[ ] [ ] [ ]

[ ] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [O] [ ]

[ ] [ ] [ ]

[ ] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [O]

[ ] [ ] [ ]

[ ] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [ ]

[O] [ ] [ ]

[ ] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [ ]

[ ] [O] [ ]

[ ] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [ ]

[ ] [ ] [O]

[ ] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [ ]

[ ] [ ] [ ]

[O] [ ] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [ ]

[ ] [ ] [ ]

[ ] [O] [ ]

Player: 'O'

Score: 'T'

[ ] [ ] [ ]

[ ] [ ] [ ]

[ ] [ ] [O]

Player: 'O'

Score: 'T'

4. This is assuming that all of the nodes have their correct win scores, because the tree is dependent on its children to calculate each nodes’ score. After a move is made, all of the prior node’s children can be deleted since they are no longer possible board combinations.

For example, suppose we start at the root node. There are nine other children because they each represent a possible gameplay scenario of where the first player places the first letter. Once a child is selected that represents the player’s decision to place the first letter, all of the other children are now invalid gameplay scenarios: the tree path can only continue from the selected child and down its children.

All of the children of N (except for M) can be deleted since those are no longer possible board combinations.

5 a. I spent about 6 hours working on this assignment plus the writeup.

b. Debugging this code was challenging. There were many possible sources of error.

**Appendix A:** I verify that the code and this write-up were authored by me. I have documented the help I have received in comments in the code files. I have not distributed my code to anyone else except via this homework submission.

**Appendix B**:

/\* CS 305 HW5

\* Spring 2016

\* Author: Jonathan Liao

\* tic tac toe game loop

\* gametree.c

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h> // for random #s

#include "gametree.h"

/\* in gametree.c -- students define \*/

GameNode \* createNode(char b[3][3], int row, int col, char player) {

if (b[row][col] != ' ') {

return NULL;

}

GameNode \* new\_Node = malloc(sizeof(GameNode));

new\_Node->board[3][3];

int i;

int j;

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

new\_Node->board[i][j] = b[i][j];

}

}

//value declarations for struct Node

new\_Node->board[row][col] = player;

new\_Node->score = '#';

new\_Node->player = player;

if(winner(new\_Node->board, new\_Node->player) == 1) { //winner scenario. No more children

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

new\_Node->children[i][j] = NULL;

}

}

}

else { //not a winning scenario. Add more children/possible outcomes

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

if(new\_Node->player == 'X') { //if it was X's turn, it's now O's

new\_Node->children[i][j] = createNode(new\_Node->board, i, j, 'O');

}

if(new\_Node->player == 'O') { //if it was O's turn, it's now X's

new\_Node->children[i][j] = createNode(new\_Node->board, i, j, 'X');

}

}

}

}

return new\_Node;

}

GameNode \* computerMove(GameNode \* tree) {

if(tree == NULL) {

return NULL; //there's nothing in this tree

}

//find best scoring tree

//first iteration to find an 'X' which is best option for computer

int i;

int j;

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

if(tree->children[i][j] != NULL && tree->children[i][j]->score == 'X') { //found a child with a score of X

return tree->children[i][j];

}

}

}

//if we made it here, the first search failed to find a child with 'X'

//second interation to find a 'T' which is second best option for computer

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

if(tree->children[i][j] != NULL && tree->children[i][j]->score == 'T') { //found a child with a score of T

return tree->children[i][j];

}

}

}

//if we made it here, the first and second search failed to find a child with a 'X' or 'T' respetively

//third and last interation to find an 'O' which is last option for computer

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

if(tree->children[i][j] != NULL && tree->children[i][j]->score == 'O') { //found a child with a score of O

return tree->children[i][j];

}

}

}

}

void calculateScores(GameNode \* tree) { //takes the completed tree that has all board game combinations and gives each node a score.

if(tree == NULL) {

return; //there's nothing in this tree

}

if(isLeaf(tree) == 1) { //is a leaf

//these two if cases test for which player won each case

if(winner(tree->board, tree->player) == 1) {

tree->score = tree->player; //current player won

return;

}

if(winner(tree->board, tree->player) == 0) {

tree->score = 'T'; //no winner here, it's a tie

return;

}

}

else {//not a leaf

int i;

int j;

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++) {

if(tree->children[i][j] != NULL) {

calculateScores(tree->children[i][j]); //calculate on next children

}

}

}

////counters for the case statements

int x;

int y;

int child;

int playerS = 0;

int playerO = 0;

int playerT = 0;

char oppositePlayer;

for(x = 0; x < 3; x++) {

for(y = 0; y < 3; y++) {

if(tree->children[x][y] != NULL) {

if(tree->children[x][y]->score != tree->player) {

//case 1:child has opposite player score

playerO++;

oppositePlayer = tree->children[x][y]->score;

}

if(tree->children[x][y]->score == tree->player) {

//case 2:child has the same score as the player

playerS++;

}

else {

//case 3:child has T

playerT++;

}

}

}

}

//actual assignments of scores occurs here from counters

if(playerO > 0) { //at least one child had an score with opposite player

tree->score = oppositePlayer;

return;

}

if(playerT == 0 && playerO == 0) { //all children have score of same player

tree->score = tree->player;

return;

}

else { //there was a mix of T's and current player letter scores. Or it was all T's

tree->score = 'T';

return;

}

}

}