Tucker Lavell
CS162 Fall 2017
Project 1: Langton's ant
Documentation

Problem

Create a Langton's Ant simulation based off user inputs. When the program starts it should have a main menu asking the user what they want to do. If the user chooses start a Langton's Ant simulation, it should then ask the user for some inputs about location and size of board. The board should be printed at every step. The program must also properly handle cases where the ant makes it to the boundaries of the simulation board. When the simulation is finished running, the user should be able to see the final board showing the ants path and a choice to restart the program or quit. To further the user experience and prevent as many errors as possible, the whole program should have a robust input validation.

Pseudocode

```
int numRows, numCols, numSteps, startRow, startCol;
Ant ant(numRows, numCols, numSteps, posRow, posCol); // startRow/startCol
        Move
                If (board[posRow][posCol] == BLACK
                        turnLeft
                Else
                        turnRight
                Flip color of slot
                Add the change of the Row to the ants position
                Add the change of the Column to the ants position
                Check bounds against new position
                        Wraps on the board if "out of bounds"
        turnLeft(row, col)
                Depending on orientation of the ant, makes a left turn
        turnRight
                Depending on orientation of the ant, makes a right turn
```

Test Table

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes

Make everything zero	numRows = 0;
	numCols = 0;
	numSteps = 0;
	startRow = 0;
	startCol = 0;
		move();	Reask for good input on all 5.	Reask for good input on all 5.

Starting Row and Column out of scope	numRows = 5;
	numCols = 5;
	numSteps = 10;
	startRow = 6;
	startCol= 6;
		move();	Reask for startRow and startCol.	Reask for startRow and startCol

Try to hit a boundary	numRows = 5;
	numCols = 5;
	numSteps = 10000;
	startRow = 3;
	startCol= 2;
		move();	Wrap to the other side of the board and continue normally.	Breaks at edge. Took fine tuning to get it to wrap.

Reflection

This assignment was extremely difficult. I greatly underestimated how long this project would take, and my work shows it. This is one of the sloppiest pieces of code I have ever written and the documentation could be much better. I basically had to do this entire assignment in just a few days. My biggest takeaway from this project was learning the real workload of this class. A project plan is by far the hardest thing I have had to do so far in any of my classes. The ant was difficult to get started, it was hard to figure out how to track the color of the squares and the direction the ant was facing. It took me numerous attempts to figure out how to get the ant to wrap around the board when it went out of bounds. In my code I got X and Y backwards, so I had to go change all my variable names from posX and posY, to indicate which was the rows and which was the columns. turnRight and turnLeft also went through many modifications before they finally worked as expected.