** Program name: Langton's Ant - Project 1

** Author: Nicole Beecher

** Date: 10/08/2017

Program Design

Create a class that contains:

1. The board
2. The ant’s location
3. The ant’s orientation

1. The board:
    a. Get user input for board size – board[x][y] – ask user for x and y values
        i. Set board to default all spaces to white
    b. Get user input for ant’s starting location on the board
        i. Set default direction the ant is facing to north
    c. Print board after each ant move
2. The ant’s location
    a. Get user input for ant’s location
    b. Depending on ant’s orientation and color of block she’s on – check next block movement for color or border (y + 1 or x + 1)
    c. Save next block color in a temporary color
    d. Move ant if permissible
    e. Change previous block to white or black (x -1 or y -1)
3. The ant’s orientation
    a. Set default to north
    b. After ant moves forward update ants orientation
    c. If next block is a border then choose random direction for ant and try to move again

Functions:

Board(rows, columns, ant x position, ant y position)

Void createBoard()

Void printBoard()

Void runAnt()

Free memory

runAnt():

```
    If block is white and ant is facing north then move ant col++ change direction to east
    If block is white and ant is facing east then move ant row++ change direction to south
    If block is white and ant is facing south then move ant col-- change direction to west
    If block is white and ant is facing west then move ant row-- change direction to north
    If block is black and ant is facing north then move ant col-- change direction to west
    If block is black and ant is facing east then move ant row-- change direction to north
    If block is black and ant is facing south then move ant col++ change direction to east
    If block is black and ant is facing west then move ant row++ change direction to south
    *If block is white then turn it black once ant moves or if the block is black then change it to white
    *Will need to check next block for border – if it is then set direction to random and move ant again
```

Test Table

Test Case	Input Values	Function	Expected Outcomes	Observed Outcomes
Input ant starting point greater than rows or cols	Input > rows	Main() while loop	Loop back to input asking for a number less than row	Loop back to input asking for a number less than row
Input ant starting at 0 (in border)	Input < 1	Main() while loop	Loop back to input asking for a number less than row	Ant was not placed on board
Input in range	Input > 1 and Input < rows	Main()while loop	Program continues	Program continues
Check ant's movement when it hits border	Input starting point at board[1][1] (upper left corner)	runAnt() if else main()while loop	Ant turns preset direction when it hits walls and tries again	Ant got stuck in a hitting wall, turning different direction, and hitting wall again. Over and over
Check ant's movement when it hits border	Input starting point at board[1][1] (upper left corner)	runAnt() if else main()while loop	Ant turns random direction when it hits walls and tries again	Ant was able to move out of corner and move forward
Ask user for row input	Testing print to screens	Main()while loop	"Row: "	"Row: Please enter number…"
Check ants first movement	Board size, ant starting location	runAnt() if else	Ant's first move will be right from facing north and on white space	Ants move was right

Check ants second movement		runAnt() if else	Ant's second move will be right from facing east and on white space	Ants move was right
Check ants third movement		runAnt() if else	Ant's third move will be right from facing south and on white space	Ants move was right
Check ants fourth movement		runAnt() if else	Ant's second move will be right from facing south and on white space	Ants move was right
Check ants fifth movement		runAnt() if else	Ant's fifth move will be right from facing west and on white space	Ants move was right
Check ants sixth movement		runAnt() if else	Ant's sixth move will be left from facing north and on black space	Ants move was left

Reflection

The greatest lesson I learned from this project is plan, plan, plan. The first day I sat down and started to write my board class with no thought to how I was going to do it beforehand. I know the assignment told me to not make this mistake and plan out my design first but "who has time for that?" I thought. I quickly learned I made the wrong decision and planning out my project would save me time overall. I grabbed a notebook and started with the steps I need to do and then use those steps as my functions in my board class. I eventually wrote, doodled, created a workflow on paper before I did any coding for each function. This helped me keep my thoughts organized and allowed me to implement my ideas from paper to code so much easier. Lesson learned: do not wing it, plan it.

I learned many things about coding in general from this project. This project was nothing compared to anything I've ever coded before in CS 161 so it was challenging. I learned how to write validations using an infinite loop and break when the correct input was typed in. I also learned during validations how to turn input into a string and check it for validity.

A part of the code I really struggled on for a while was how to create a border. I know it wasn't required but I figured it would be easier to check for a border then turn my ant's direction. I thought I could create another array that is larger than my board array and have it be a boarder but then how would I check ahead in my array to the next cell to see if it was a border or not. I finally decided on adding 2 to my rows and columns and creating a border within my board array. The first and last row and column are the border and I implemented code to check for the border symbol.

Above in my test plan I tested starting my ant in a corner to see how she could maneuver out. Originally in my code if my ant hit a wall I had a preset direction change. This caused my ant to get stuck. I changed my code to pick a random direction change if she hit a border and now she can maneuver corners without getting stuck.

Above in my test plan I tested my text outputs for each user input step. I noticed when I got to the part where I asked the user to input a row that my while loop for validation would print my error before I typed anything at all. This confused me since I was using this same loop for all my input and it was working fine for everything else. After searching the internet for people who had a similar issue I found out if I had a cin.ignore line of code in my while loop it would clear out anything my program might be holding on to. I gave it shot and it worked. Instead of my program printing out "Row: Please enter number less than or equal to rows and greater than 0:" it simply printed out "Row: " then waited for user to input a value.

3 Add route
2.4 retire route

Menu
Play
Quit

bool gameStatus

Main

```
while (gameStatus == true) {
    system ("cls")   // clears screen (menu)
```

run game function

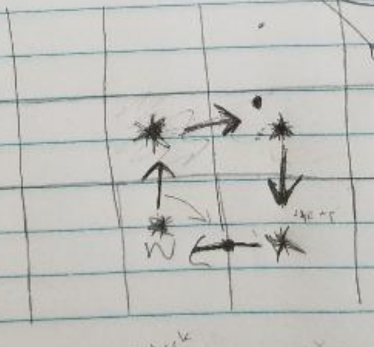
```
if ( * is facing north AND space = ' white ' 
~~int col2 = col + 1;~~
if (board[row][col] == 'white') {
    board[row][col] == 'black')
if (board[row][col] == 'black') {
    board[row][col] == 'white')
col++;
    board [row] [col] = '*'
```

tempf = black
move right ant or north
temp → tempb = black

set to white to start game

look ahead store color
~~tempf = white~~
move ant
tempB if white
change [illegible]

tempf = black



tempf → D
↓
tempB
change to other color

If face = [+ tempF = white

Tempf
move ant
tempb = temf
tempf
change array color a tempB

tempf = black
~~tempcolor~~ = black
move ant
tempB = tempcolor want
change got to black