Zach Mitchell
CS162 Section400, Spring 17
Reflection of Langton's Ant

Broadly Speaking, the main things I need to develop are, and their functions will be:

1. Menu
    1. Guides the user to starting a game
    2. Gives them choices for size of board, where the ant will start, or **random start**
    3. Asks the user if they want to take another step
    4. Asks the user if they want to play again
    5. Asks the user if they want to quit
    6. Input validation
2. Board
    1. Initializes to size specified by user, filled with ' ' character for whitespace
    2. Has member 'isBlack' and 'isWhite' to keep track of the color of each cell
    3. Prints board to screen
    4. New and Delete (destructor)
3. Ant
    1. Remembers it's position, has a start position
    2. Remembers it's direction, has a start direction(maybe?)
    3. Looks ahead to the next position and makes decision based on 'isBlack'/white to turn left/right
    4. Knows the bound of the board and will turn around or loop when it strikes side of board

What is the basic overview of the progression at each step, the meat of the program?:
1. <u>Look ahead</u>, isBlack?
2. <u>Move</u> into space
3. Based on color in 1, <u>change direction</u>
4. Based on color of previous, <u>change color of previous</u>

Had to change, as the ant will be taking up the spot, so cannot read either black/white. Got around this by reading the color of the next cell and saving that information for the change afterward in the bool _isBlack :

1. Start knowing the entire board is white (ex: bool isBlack() > false)
2. know the direction always starts facing left
3. based on color and direction turn (turn left for black, right for white)
4. Change color of current cell
5. Look to the 'next' cell, checking color (bool isBlack())
6. Move into next cell
7. Loop until steps completed

Zach Mitchell
CS162 Section400, Spring 17
Reflection of Langton's Ant

Menu must contain:

Explanation for user
Do you want to play?
    Yes - start game
    No -> end game (jump to end)
Inquire as to specifics of setup (size, steps, **random**)
run program
Ask if they want to see it again
    Yes - start back at beginning
    No - end game

Road blocks:

Should a board object take an ant object as a param, or vice versa?

Decided an ant should take a board as a parameter, as I believe it would be easier to sense bounds.

Issue wrestling with direction. Created an enum to track, with switch

What to do when hitting the outer bounds of the board:

This took a lot of trial and error. To be honest I was tied at first to making the ant hit the wall and then turn around, but didn't like that because of the rules of the game, he would keep running along the edge of the wall. I guess that is closer to reality, but I liked the interesting design patterns so instead opted to have the Ant wrap around the edge of the board.

At first I fiddled a lot with my 'move' functions thinking that would be the only place that could cause issue (if the ant moved off the board). But after painstaking error checking (I really should have used gdb, but it doesn't work on my Mac so kept pounding away) I realized that my functions that kept checking the next cell for black or white were what was causing me to have segfaults. Once I tooled everything to look at the opposite side of the board and to move to the opposite side of the board, I was set.

I created the menu and input validation function on the main.cpp because they were the generic functions that I'll reuse in later projects.

From here I just had to make it presentable to the user adding things like a visual x-y axis descriptor and converting the y to be more attuned to what someone who's taken pre-algebra would.

My input validation still only works with ints.