DESIGN

1. Write logic for movement of ant
   **Update ant position, space color, curr pos
   **check row and col to ensure stay on board (>0)
   a. If space == white, change to black
      If dir == up
         1. Move right 1 col
         2. Direction == up
      i. If dir == right
         1. Move down 1 row
         2. Dir == down
      Ii. if dir == down
         1. Move left 1 col
         2. Dir == left
      Iii. if dir == left
         1. Move up 1 row
         2. Dir == up
      iv.
   b. if space == black, change to white
      I. if dir == left
         1. Move down 1 row
         2. Dir == down
      Ii. if dir == down
         1. Move right 1 col
         2. Dir == right
      Iii. if dir == right
         1. Move up 1 row
         2. Dir == up
      Iv. if dir == up
         1. Move left 1 col
         2. Dir == left

2. List steps needed in main.cpp
   a. Size, location, number of steps input asked for from user
      i. INPUT VALIDATION
         1. Must be greater than 0
         2. Don’t allow characters
         3. Reprompt for input
   b. Menu options created/given to user
      i. START

    ii. QUIT
c. Create ant object
d. Make board
e. Draw board
f. Draw Ant
g. Create for loop which will encapsulate Ant logic movement(above), and executive however many times "steps" specified by user
h. Print board to screen
i. Deallocate memory

TEST CASE	INPUT VALUES	DRIVER FUNCTIONS	EXPECTED OUTCOMES	OBSERVED OUTCOMES
Input too low	Input < 0	main() cin >>	Loop back to input	Loop back to input
Input at 0	Input = 0	main() cin >>	Loop back to input	Loop back to input
Input letter	Input f	main() cin >>	Loop back to input	Loop back to input
Input correct range	Input = 10	main() cin >>	Continue to next input	Continue to next input
Input very high	Input = 100000	main() cin >>	Continue to next input	Continue to next input
Input too high	Input = 3000000000	main() cin >>	Loop back to input	Loop back to input

REFLECTION
My Ant movement code ended up being more complex than I designed. I added previous column and previous row variables in order to change the previous space's color. I couldn't figure out the menu for a while. I started off with an if/else statement and it didn't work out, so I learned how to use switch (wasn't very comfortable), and it worked better. Writing the input validation while loops took me a long time. I wasn't sure how to ensure the user didn't input a character instead of a number because I would use !(cin>>[variable]) to validate, but the program would crash if a character was entered. So I googled and used cin.fail(), cin.clear(), cin.ignore() instead which basically flushes the input (to the best of my understand). But, I had to do this repetitively and I

feel like there’s a way to eliminate separate while loops for every input. I had a lot of trouble with ensuring my brackets were correct. I eventually transferred everything to atom to check them. For example, the for loop encapsulated most of the code, and there were just many long nested statements. The ant class took pretty much no time to write, and was pretty straight forward (just getting and returning variables). What was most time consuming was making sure the ant movement was correct for all 8 maneuvers. I chose to make a long and somewhat straightforward program so as to get the logic correct. I’m sure it can be shortened and repeated code eliminated, but given it was a longer program, I needed to be able to think through all the steps in order to fix my errors. Error testing took forever with this code though, because of that. I had many many small errors (a bracket instead of a parenthese, for example), so it’s bulkiness made the debugging process way more time consuming. I’d definitely think of a sleeker way to approach this problem next time.