struct Direction [0=North, 1=East, 2=South, 3=West]

Ant

Ant Location = *Board[X][Y]

void Ant_move (char Ant Location *[X][Y], Direction* direction)

if square white

if Direction* = North

Direction* = East

Ant Location* = "#"

Ant Location = *Board[X+1][Y]

else if Direction = East

Direction = South

Ant Location = "#"

Ant Location = *Board[X][Y-1]

repeat for South + West

if square black same as above except change square to "_"

if [X+1] > arraysize-1

Then change x to [0]

same with y

Also if [x-1] < 0

Then x to [arraysize-1]

same with Y cases

Input: Size of Board (Row + Columns), # of Steps
Starting Loc of Ant

Keep track of Ant Location's square color

Ant (Row, Col, X, Y)

Ant will build 2D array w/ Row + Column

Variables in Ant:
- currentDir = North, South, East, West
- currentSqColor = Black or White
- currentX = X
- currentY = Y

turnRight
- if (currentDir = North)
  - array[currentX][currentY] = "#"
  - currentX += 1
  - if (arr[currentX][Y] = " ")
    - currentSqColor = White
  - else if (arr[currentX][Y] = "#")
    - currentSqColor = Black
  - array[X][Y] = "@"

Repeat for All Directions with correct changes to X or Y

Also for turnLeft

input Validation for integers

Needs size < 80

Size of Board (Rows + Columns), Steps

Input: Starting Location of Ant (X, Y)

- Let user know Ant Starts North

Calc: Ant(Row, col, X, Y)

Direction

enum (North, East, South, West)

Enum (White, Black)

Ant will build a 2D array of Char

Ant Location (X, Y)

for (Steps)

antMove ()

antEnd will de allocate memory

antMove is heavy lifter function

currentLoc = White)

if (

turnRight()

array[X][Y] =

turnRight()

array[X][Y] =

private turnRight

if (antDir = North)

antDir = East

if (antDir = East

antDir = South

private turn left

if antDir = North

antDir = West ...

Andrew Sturevant
CS 162 – Project 1 Test Plan

Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**	**Observed Outcomes**
Input too low (Rows & Columns)	Input < 1	Main() checkIntRange()	Repeat within checkIntRange asking for valid #s	Repeated as expected
Steps at 0	Input = 0	Main() checkPosInt()	Program performs normal, 0 steps taken	As expected
Rows, Columns & steps at extreme low	Row = 1 Col = 1 Step = 0	Main()	Call all relevant functions, and works normal	As expected
Rows, Columns & steps at extreme highs	Row = 80 Col = 80 Steps > 10000	Main()	Call all relevant function and works normal (might take some time to complete)	As expected
Rows & Columns too high	Rox > 80 Col > 80	Main() checkIntRange()	Repeat within checkIntRange asking for valid #s	As expected
Input not an integer	Row = abc Col = abc Steps = abc Location X = abc Location Y = abc	Main checkIntRange() checkPosInt()	Repeat within the input validation until a valid integer if placed	As expected
Menu option outside of bounds	Input != 1 or 2	simpleMenu()	Repeat until a valid option is selected	As expected

Andrew Sturtevant
CS 162 – Project 1 Reflection

For Project 1, I do not feel I did as much as I could have in decomposing my Ant class into its components, however by the time I noticed it, I did not have the time to go back and re-design. In the future, I need to resist the urge to turn pseudo-code into actual code as I am working. I tend to delve into the weeds instead of trying to keep my design process looking over the entire project.

I had also not thought much about my menu, so that was not designed at all in the beginning process, but with some examples by classmates in Piazza, I think I was able to develop a passable and simple menu that I will be able to grow depending on future assignment needs.