The version of the Langton Ant that we made has the time complexity of O(1).

The reasoning behind the Ant having a time complexity of O(1) is that we give it a certain amount of iterations that it has to go through. We give it a total amount of 1000 steps that it can go through before it stops, and a board size of 41 x 41. If this was not the case, and the user could decide what the size of the board would be, it would be dependent on the board size. For this case then the complexity would be O(x\*y) in addition to the O(N), x and y would just be the length and width of the board (amount of cells). It would also be O(N) because we would not be limiting the amount of steps that it can take, since there would be no given limit, it would be O(N).

We had switch cases in our program that resulted in a complexity of O(1) and each of our function calls also resulted in a complexity of O(1).

It's important to note that we had function calls inside of function calls, which in this case did not affect our time complexity because functions such as: save(), beginPath(), fill(), etc. all had a time complexity of O(1), so in the end our time complexity as a whole remained O(1).