CS162 Project 1: Program Design and Testing

Implementing Langton's Ant

Goals

- Review of programming with arrays
- Convert requirements (i.e. the rules) to a software design
- Think about the testing process, before and after writing the program.

You will design, implement, and test a program that implements a simulation of Langton's Ant. For a brief explanation consider Wikipedia: https://en.wikipedia.org/wiki/Langton%27s_ant. Note that it can be considered as a cellular automaton. This means that you have an array or matrix of cells. Each turn or step, the value of each cell may change based upon a simple rule.

For the Ant there are two simple rules. The rules are:

1. In a white square, turn right 90^o and change the square to black.
2. In a black square, turn left 90^o and change the square to white.

You can use a blank character for a white space. You can use the number sign ("#") for black. And for the Ant? An asterisk ("*"). ☺ Left ? Right? Those are relative directions! How are you going to keep track of the direction the Ant is facing? How will you "remember" what color the current cell occupied by the Ant is or was? You will create an Ant class to help organize, hold, and manipulate this information.

Your program will prompt the user to enter the number of rows and columns for the 2D array. You should also prompt the user for the number of steps. Suggest values to the user. Your program should display each step. Use functional decomposition to develop functions to validate the input. What parameters and criteria do these functions need? You should start by asking the user for the starting location of the Ant. You should add the option for the user to have a random starting location. For testing purposes, always start the Ant at the same location. Then try different locations. You should start saving utility functions for future programs you write.

You will create your design document BEFORE you start coding. Simply stated, program design is identifying the problem to be solved, identify the inputs, specify the desired output, and then develop the algorithm(s) to convert the input into the output. In your reflections document, you can explain how you had to change the design because your first idea just didn't work. Just explain what you learned. The TA will grade, in part, on how well your implementation matched your design.

Note: There are a lot of details that are not determined in this requirement. They are left for you to decide, as long as your program meets the required rules in the grading part. For more questions, ask your grading TA for clarification or post it on Piazza.

What to submit:

- Your program files.
- The reflections document (pdf).
- A makefile. If you do not provide a makefile it will not be graded.
- All the files must be submitted in a zip archive.

GRADING

- Programming style and documentation (including memory leak check): **10%**
- Build the array: **10%**
- Input validation function(s): **5%**
- Create the source and header file for the Ant class: **10%**
- Displays each step correctly, including running for the correct number of steps: **10%**
- The display allows the user to see the change(s) in the shape(s): **10%**
- Allow the user to specify the starting location of the Ant: **10%**
- Allow the user to choose a random location to start: **10%**
- Implement a menu function that can be reused in later programs: **10%**
- Reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment: **15%**

IMPORTANT POINT:

Keep in mind that this is a programming assignment. You are not writing a program to share with others for their enjoyment. Too often students forget this and the attempt to make it entertaining which also makes it more difficult, and has little to do with the grading. You are graded on developing the design, writing the code, and testing your program. You will use text-based output. It may not be pretty, but it will demonstrate your design and implementation works. If you have other programming experience and want to use something else, check with your grader before doing it.

Use the Internet to ONLY research how the 'game' is structured and designed. Design and write your own program. **Do NOT use any code that you did not write for yourself**. Much of what is available is **more** complicated or advanced than what is required here. It is harder to use it than to write your own. Submitting any code you did not write is a violation of the university's academic dishonesty policy. **Keep in mind, if you can find it using Google, the TA's can, too**.

Hints:

1. Use the grading breakdown to plan your program. Use incremental development. Get one part working. Test it. Save a copy and continue working on the next step in your plan. This ensures you will have something to submit!

2. Be very careful about borrowing any code, or ideas you see in someone else's code. As always, any code submitted must be yours and yours alone.

Suggested plan to design and code this project:

First, get a pencil and some paper. Seriously. Low tech but it works. Develop an algorithm to apply the rules. Answer the questions about the Ant; knowing the direction it is moving, and saving the color of the cell it is in.

Now that you have your algorithms developed, convert them to pseudocode or flowcharts, whatever system you chose to represent program logic. Walk through your pseudocode. Did you miss any details? It's much easier to find and fix logic errors now.

Now it is time to dig out the keyboard and start entering code. ☺