Assignment 1 – LRC Report Template

Nickolas Tran

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

Randomness

Describe what makes randomness. Is it possible for anything to be truly random? Why are we using pseudorandom numbers in this assignment?
Randomness is the idea that there is no definitive pattern or predictability. I do not think it is possible for anything to be truly random. As the most non-bias way to generate random numbers is through a mathematical equation. Pseudorandom numbers are used to provide a sense of randomness.

What is an abstraction

When writing code, programmers often use "abstractions". Define an abstraction in non computer science terms (Don't google it!)
An abstraction is an idea that becomes more generalized by the quality.

Why?

The last assignment was focused on debugging. How can abstractions make debugging easier? What other uses are there for abstractions? Hint: Do you have to be the one to write the abstraction?
Abstractions can make debugging easier by differentiating the distinctions and making the ideas more clear. Abstractions can also be used to organize code and maintain it in a clean way.

Functions

When you write this assignment, you can choose to write functions. While functions might make the program longer, they can also make the program simpler to understand and debug. How can we write the code to use 2 functions along with the main? How can we use 8 functions? Contrast these two implementations along with using no functions. Which will be easier for you? When you write the Program design section, think about your response to this section.
8 functions allows for specific functionalities to occur, whilst 2 functions will make it more readable, smaller, and more manageable but less functional. Functions allows for easier readability and organization. Using

no functions creates a scenario with no readability. I think 8 functions might be easier as it could be more direct to what I need to do.

Testing

The last assignment was focused on testing. For this assignment, what sorts of things do you want to test? How can you make your tests comprehensive? Give a few examples of inputs that you will test.
I want to test
- zero inputs
- extreme inputs
- boundary inputs
- input validity

Putting it all together

The questions above included things about randomness, abstractions and testing. How does using a pseudorandom number generator and abstractions make your code easier to test?
Using a pseudorandom generator and abstractions makes the code easier to test through creating a controlled and repeatable tests.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags that your program uses, and what they do.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[1]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[1][2][3].

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

[1] This is my footnote.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have?
- Setting up parameters
- Generating psuedocode
- Calling functions
- Testing

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge.
    - Parameters passed to the code.
- Generate psuedonumbers into the function
- Produce a sequence
- Apply abstraction to specific concepts for better organization
- Run various functionalities using pseudorandom numbers
- Evaluate the code through test scripts.

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.