# MonDRAWsity – 30 points

This was a good showcase for communication and its challenges. Technical writing is about confusing and complex things, and this game was about describing complex and difficult to describe monsters. Words can only communicate so much, and a visual aide can communicate so much more (but less precisely). Metaphors and comparisons alongside concrete numbers are a really good way to communicate complex and foreign things through words.

# Instructional Writing – 30pts

External Readme file that will be attached somehow.

# Prove You’re Human – 20pts

Things are always changing. There’s a specific challenge/change we aren’t making. The way we write! (Wow crazy left field moment!) The tone we send emails does not match the tone of modern business. This is honestly exactly what I don’t like about how we’re taught to “be professional”, it’s the exact same kind of out-of-touch that these email examples are. Clear concise and easy to understand communication is paramount.

# Clean Code Notes – 30pts

Chapter 1:

We must treat our code like professionals. We need to have and uphold standards and best practices. This boy scout tule is a little different than what I grew up with, but I agree “Always leave the campground cleaner than you found it” is an important mantra to have with code. (My troop changed the saying to “Leave no evidence” haha). Anyways this section is essentially a preamble going over the importance of good code, and how ridiculous and illogical it is to let bad code exist.

Chapter 2:

Good names reveal intention, this means that larger scope things will have simpler names, and tiny scope things will have longer more specific names. Using custom classes instead of ambiguous built in data structures can help with displaying intent. Names should try to have different “silhouettes”, similar looking variable names are tempting because they look nice, but they are not easy to understand. Things that are similar need meaningful distinction, in a simple math method it would be easy to do “num1” and “num2” as variable names, but you should use the actual math terms for understandability. Also avoid noise words like “Data” literally everything is data you moron. Names you can say are also a must, people will need to talk about this code someday and that day is either tomorrow or years in the future. Class names are nouns not verbs! Otherwise that clashes grammatically with the concept of OOP!

Chapter 3:

Functions should be tiny! Microscopic even! They should be microbes like Salmon! Functions should not have deep nesting within them, if deep nesting is found then you likely have more functions to extract. This goes with the “FUNCTIONS DO ONE THING” message. A standard thought in response to that is “how do we know if a function is doing one thing?”, and the answer is “You don’t”. To find if a function is doing one thing you have to look at it and ask yourself “can I make part of this another smaller function?”, if so then do so, if not, then your function is doing one thing! Each function should be at a set level of abstraction, and the functions in it should be at a different more precise level of abstraction as you work your way down. This is for skim-ability, you should be able to look at a class and understand at a high level what’s going on, and if you need to know more you should be able to inspect the methods to find more details, and you should be able to follow this pattern down to the bottom of your program. Switch statements are clunky and generally to be avoided if possible (it is not always possible). A long descriptive name is just as good or better than a long descriptive comment. Methods should have as few arguments as possible, and they should never have an output argument. Boolean arguments imply the function does more than one thing, and we don’t want that.

Chapter 4:

Comments are necessary failure! Most comments are substitutions for good method names and are symptoms of a method that isn’t doing just one thing! License comments are an exception to this. Comments that are good are ones that explain intent, or give context, but those comments would be even better if they were just self-documenting code. TODO comments are ok unless you don’t (to)DO them. Warning future programmers of dangerous (time consuming) code is helpful. Examples of bad comments are ones that don’t say much or say what is already obvious. The most evil kind of comment is the misleading or misinformed comment, this isn’t always how the comment starts, but when someone comes along and changes code but not comments this can occur. Comments that fulfill the same purposes as version control are another example of bad comments.

# Create a Troubleshooting Guide – 40pts

External File: Basic Web Troubleshooting Guide.md

# Graph of the Week – 20pts

<https://www.turnersgraphoftheweek.com/uploads/1/1/2/2/112252793/gow_11-1_2021_gen_z_top_concerns__compared_to_other_generations_.pdf>

This is on the mental health of Gen Z. There are 2 graphs, one that explains that Gen Z self-reports their mental health as the worst when compared to other generations. The second graph looks at the reasons why, it seems that Gen Z is a bit more stressed about all the big problems in the world. I think I may be biased because I am Gen Z and I also worry about pretty much only these things. The pdf asks, “What do you foresee happening in the next 10 years?” and I foresee a great uprising due to those in power not fixing the things on this graph.

# Graph of the Week II – 20pts

<https://www.turnersgraphoftheweek.com/uploads/1/1/2/2/112252793/gow_11-8_2021_the_rising_cost_of_college_.pdf>

This graph shows how college prices have been going up like crazy since the early 80s. I didn’t live in the time, but I have a feeling this corresponds with a shift in what kind of work was wanted from people. Nowadays all the high paying jobs are big thinker jobs, so you need a degree, but if everyone needs a degree it has less value, but because everyone needs a degree, colleges can pump up the prices. It’s a pretty great spot to be in as a business, and it’s a pretty awful spot for students. We need a second renaissance. As for my future predictions, I honestly am not sure. AI could bring many things including a shift in the jobs the economy wants, the need for a degree, or even the way we value work altogether. There’s a second graph at the bottom that I almost didn’t see. It’s comparing the costs between undergraduates and postgraduates. The costs are almost identical, but graduates pay less on tuition and fees. The graph has 4 bars, but they are only labeled 2 as graduate and 2 as undergraduate, without any distinction within those categories. I believe that the first bar in either category is for public schools, and the second bar in either category is for private schools.

# Fix a bad ReadMe – 30pts

External Files: dontreadme.md was changed to yesreadme.md

# Test Plan – 20pts

External File: HiLevelTestTempl.doc

# Fix a bad ReadMe II – 30pts

External Files: oldREADME.md was changed to newREADME.md

# The Power of Written Communication in a Technological Age – 30pts

This guy kinda just talks about how writing a letter can have a huge impact on those in the future. I’m not entirely sure what “write or create an infographic with your summary and/or reactions” means, so I am talking about my summary and reactions as well as an “infographic”. The speaker recounts the importance of letters to his father and grandfather and how their legacy was able to ripple across generations through these letters is significant.

Here's the MSPaint infographic, in the form of a letter envelope:

A white board with black text

Description automatically generated

I feel conflicted if I should clean this up with a font instead of handwriting, but I think the intimacy of handwriting is important to what is important to the speakers connection to writing a letter.