Anthony Herco
CMSI 402
January 29, 2018

Homework 1

Problem 1.1
The basic tasks that all software engineering products must handle are: requirements gathering, high-level design, low-level design, development, testing, deployment, maintenance, and wrap-up.

Problem 1.2
- Requirements Gathering: Determine, in clear terms, the customer's desires.
- High-Level Design: Describe the major parts and interactions that make up the application.
- Low-Level Design: Describe the details and intricacies of each piece of the application for developers to implement.
- Development: Write the code of the application as described in the design.
- Testing: Test the application in order to detect and prevent bugs, errors, or flaws before deploying it.
- Maintenance: Additions, enhancements, updates, and bug fixes to the live application.
- Wrap-up: Reflect upon the strengths and weaknesses of the project's development history in order to improve the development of future projects.

Problem 2.4
Link to Google Doc created: https://goo.gl/aTckN8

Problem 2.5
Just Barely Good Enough means that you should not write more code documentation or comments than is necessary.

Problem 3.2
Critical Path :: Expected length == 32 working days
1. G :: Rendering engine
2. D :: Character editor
3. E :: Character animator
4. M :: Character library
5. Q :: Character testing

Problem 3.4
Link to GANTT Chart: https://goo.gl/qU9kjA

Problem 3.6

Deus ex machina problems can be handled by scheduling extra time at the end of each task to account for the time lost.

Problem 3.8

The biggest mistake one can make while tracking tasks is to not act when a task encounters obstacles. The second biggest mistake is to assign more people to a task with the mindset that "more people working on a task equals spending less time on that task". Such an event would increase the time it takes to finish the task because it would take time for new people to learn about and understand the task, unless those people have the specific abilities to finish the task quickly.

Problem 4.1

Five Characteristics of Good Requirements:

1. Clear
2. Unambiguous
3. Consistent
4. Prioritized
5. Verifiable

Problem 4.3

No implementation requirements, unless you need new hardware to run the application.

a. business
b. user, functional
c. user, functional
d. user, functional
e. nonfunctional
f. nonfunctional
g. nonfunctional
h. nonfunctional
i. nonfunctional
j. functional
k. functional
l. user, functional
m. user, functional
n. user, functional
o. user, functional

Problem 4.9

Must:

- Switch from landscape to portrait view. Users should not have to hold their phone sideways to play this game.
- Move the “New Game” button to a main menu screen so users don’t accidently press it while they are playing.
- Take the user back to the aforementioned main menu screen after a game ends.

Should:

- Source the word bank from a dictionary API instead of a hardcoded list.
- Have a score keeping system to encourage the user to beat their previous scores and play more.

Could:

- Switch to a more aesthetically pleasing skeleton graphic.
- Have a button that offers a hint to the player if he or she is stuck.

Won’t:

- Support for languages other than english.
- Allow the user to pick colors for the theme of the app.