Problem 1.1, Stephens page 13
Gathering requirements, high-level design, low-level design, development, testing, deployment, maintenance, postmortem/wrap-up are the tasks all projects must handle.

Problem 1.2, Stephens page 13
- Gathering requirements: Outlining the customer's wants and needs by documenting what the customer will be getting and what the project members will be building.
- High-level design: Basic architecture details for all the requirement aspects.
- Low-level design: Information about how the high-level design aspects should generally work, sometimes including how different project pieces work together.
- Development: Programmers write, test, and remove bugs in the code to fulfill the low-level design requirements.
- Testing: Remove bugs by having developers test their own code and then have other programmers (who didn't write the code) test it until there is an acceptably low rate of bugs or the bugs do not severely impact the user's experience.
- Deployment: Roll out the software project and any other related components that are needed, such as upgraded hardware, user training, etc.
- Maintenance: As users find bugs or suggest enhancements/new features, improve and implement changes to the software.
- Postmortem/wrap-up: Evaluate the project to understand what went well and how to continue that success, and on the flip side, understand what went poorly and how to prevent issues in the future.
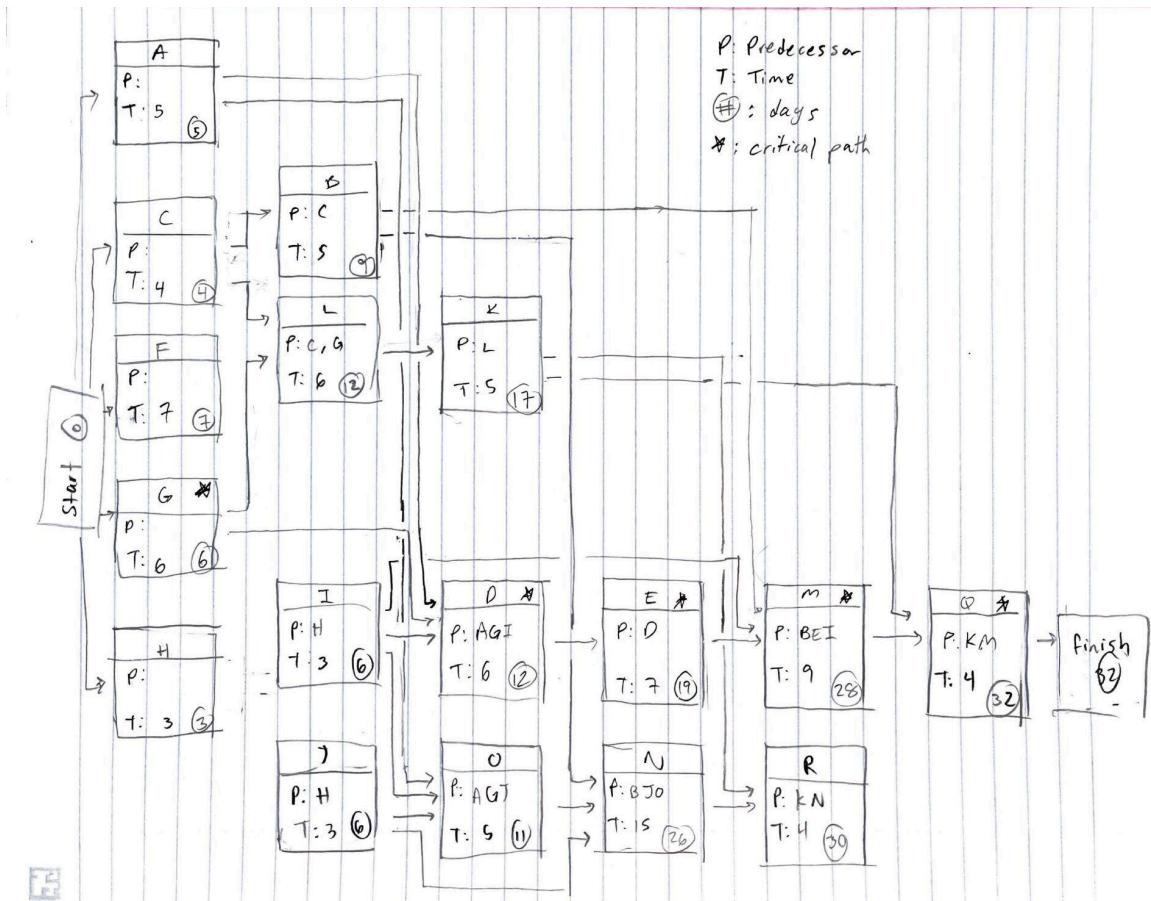
Problem 2.4, Stephens page 27
In the version history, I can click through timestamps of where I edited my file. The text changes (both additions and removals) are a different font color and highlighted to differentiate them between the unchanged text. With GitHub versions, I am able to hide/expand both the changed/unchanged chunks of code. I can also compare versions side by side, rather than only looking at one at a time with Google Docs.
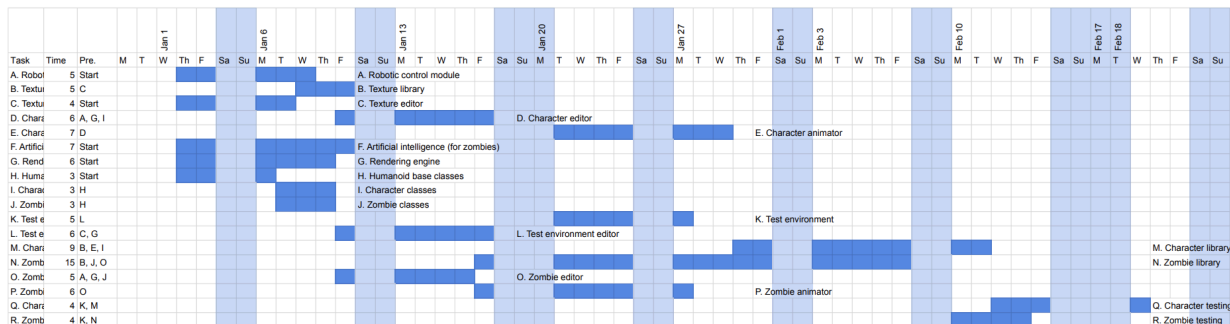
Problem 2.5, Stephens page 27
JGBE stands for "just barely good enough" and means documenting code without being excessive, so just enough so that programmers can understand it. Documentation can take a long time and the code may change anyways, so it is important to not waste development time on excessive comments.

Problem 4.2, Stephens page 78

Legend (handwritten):
- P: Predecessor
- T: Time
- ⊞ : days
- ※ : critical path

Nodes (handwritten):
- A — P: , T: 5 (5)
- B — P: C, T: 5 (9)
- C — P: , T: 4 (4)
- L — P: C, G, T: 6 (12)
- K — P: L, T: 5 (17)
- F — P: , T: 7 (7)
- Start (0)
- G ※ — P: , T: 6 (6)
- H — P: , T: 3 (3)
- I — P: H, T: 3 (6)
- D ※ — P: A, G, I, T: 6 (12)
- E ※ — P: D, T: 7 (19)
- M ※ — P: B, E, I, T: 9 (28)
- Q ※ — P: K, M, T: 4 (32)
- finish (32)
- J — P: H, T: 3 (6)
- O — P: A, G, J, T: 5 (11)
- N — P: B, J, O, T: 15 (26)
- R — P: K, N, T: 4 (30)

Critical path has the tasks G → D → E → M → Q and will take 32 days.

Problem 4.4, Stephens page 78

Task	Time	Pre.
A. Robotic control module	5	Start
B. Texture library	5	C
C. Texture editor	4	Start
D. Character editor	6	A, G, I
E. Character animator	7	D
F. Artificial intelligence (for zombies)	7	Start
G. Rendering engine	6	Start
H. Humanoid base classes	3	Start
I. Character classes	3	H
J. Zombie classes	3	H
K. Test environment	5	L
L. Test environment editor	6	C, G
M. Character library	9	B, E, I
N. Zombie library	15	B, J, O
O. Zombie editor	5	A, G, J
P. Zombie animator	6	O
Q. Character testing	4	K, M
R. Zombie testing	4	K, N

Problem 4.6, Stephens page 79

In the project planning stage, I can expand each task's expected time by a certain amount (like 5%) to prepare for unplanned delays. Another strategy is creating tasks specifically for sick time and vacation time.

Problem 4.8, Stephens page 79

The two biggest mistakes in tracking tasks are (1) ignoring any problem that occurs to save it for later and (2) bringing in more developers to work on a task when you are behind.

Problem 5.1, Stephens page 114
Clear, unambiguous, consistent, prioritized, and verifiable are five good requirement characteristics.

Problem 5.3, Stephens page 114
    A. Business, Functional
    B. User, Functional
    C. User, Functional
    D. User, Functional
    E. User, Functional
    F. Non-functional
    G. Non-functional
    H. Non-functional
    I. Non-functional
    J. Functional
    K. Functional
    L. User, Functional
    M. User, Functional
    N. User, Non-functional
    O. User, Functional
    P. User, Functional

Problem 5.9, Stephens page 115
M: Must
- Colors must adhere to accessibility guidelines
- Include an instruction button that when pressed, displays instructions on how to play the game
- Have a restart button once a game is finished

S: Should
- Have a light mode and dark mode
- Rotate based on landscape/portrait mode on phone
- Have a scoring system

C: Could
- Have multiple hangman character illustrations
- Play background music that can be turned on/off
- Allow user to choose different difficulty levels

W: Won't (optional)
- Multiplayer mode: initiate multiplayer races by having a join code for each game
- Expand words/keyboard to different languages
- Share results to other apps/social media