**Discussion Topics: Pragmatic Programming**

In this module's discussion board assignment, answer the following questions:

1. Using the ***Pragmatic Programmer*** reading assignment, select one (1) topic and complete the following:
   * Why did you select this topic?
   * Summarize the main points (in your own words) of that topic in three or four sentences.
   * Find at least one additional resource (video, book, article, website, etc.) that supports your summary. Include a link to that resource.

After reading the topics for this week’s module, I decided to focus on Topic 24: Dead Programs Tell No Lies. We all make mistakes, especially when coding. It can be tempting to ignore the problem, but this will snowball into more complex issues. But by being proactive, we can overcome them.

I chose this topic since I often feel trapped in the headspace, thinking that everything needs to be perfect, and if a program crashes, especially early, it can feel discouraging. I can relate to feeling like I am too close to the problem. I like to check my perceptions with others, especially when feeling stuck. Sometimes, even just by going through a problem with someone and explaining it, I can detect the error.

When experiencing an error, being diligent and realistic about where you might have gone wrong is vital. Reading the error message is important since it provides helpful insight that can help diagnose the problem and help remedy it. Crashing early can be considered a good thing since it helps identify the mistake early on in the program instead of building on top of it (Thomas & Hunt, 2024/2020, p. 91). We also need to be pragmatic by writing code that is less likely to be outdated. A way to do this is by incorporating code that is automatically propagated (Thomas & Hunt, 2024/2020, p. 91). If there is an error in the program, every result that pops up must be questioned and tested further (Thomas & Hunt, 2024/2020, p. 92). According to Matt Klein (2019), when software crashes, it can be a “valid error handling method,” which can actually help improve the code if utilized correctly.

**References**

Klein, M. (2019, April 7). *Crash early and crash often for more reliable software*. Medium. https://medium.com/@mattklein123/crash-early-and-crash-often-for-more-reliable-software-597738dd21c5

Thomas, D., & Hunt, A. (2020). *The Pragmatic Programmer: your journey to mastery*. Addison-Wesley. (Original work published 2024)

***Before you submit your thread, put your name in the subject line.***

**Assignment Requirements and Grading:**

1. An initial post of approximately 250 words is due by **Thursday, 11:59 p.m. CST**.
2. For the initial post to be considered substantive, it should be at least 250 words in length and fully cover the topics being presented. Single-sentence definitions or responses will not be awarded points.
3. Submit your post by clicking on the **Assignment Link** above, then **Create Thread**. You must create a thread in order to view your peers' posts. Tip: Create your post in a Word document and then copy and paste your work into the thread.
4. A minimum of three (3) responses **to the original threads of other students** of 100-200 words each are due by **Sunday, 11:59 p.m. CST**.
5. To view the rubric grading criteria, click the following link: [Discussion Board Grading Rubric](https://content.bellevue.edu/cst/csd/rubricdbv3.pdf).

**(50 points)**

Nardos, I really enjoyed reading your post! I understand what it is like to always be jotting down work-related or school-related information. I have always considered keeping a journal but always come up with a reason why it is not practical or worth the energy spent on it. When I write notes for work, I tend to throw them away afterward, but there have been many occurrences where I would dig through my recycling bin to retrieve a note because I needed to reference it again. Keeping a daybook, in general, instead of relying on sticky notes and other methods, could be incredibly beneficial.

Joe, I think you did an excellent job on your post! I also chose to focus on the same topic for my post. You are correct; it feels like we have only just begun to scratch the surface of error handling. The idea of thinking that an error could not possibly be erroring even when the message pops up is something I have faced. It seems like a frustrating reality when, after multiple glances, the code seems like it should be functioning properly. I like how you included the summary series on YouTube! I took a look and found it helpful! It seems like the perfect companion piece for our textbook.

Jessica, you did a fantastic job recapping topic 24 from our text. I also chose the same topic, so seeing the resource you chose to include from the Medium website is intriguing. I also included an article from the same site, so reading another interruption of handling program crashes was very helpful. It is reassuring knowing that I should not fully feel defeated when my program fails early and remember that it is better than finding out there is a hidden error from an undisclosed location of code. Reading the syntax errors is beyond helpful in addressing the problem that occurs.