Academic Integrity Response2

Name: UC Choudhary

PSU email: ufc5009@psu.edu

PSU ID: 971854622

May 17, 2024

Contents

1 Notes before I begin:

Before I begin, I would like to extol Dr. Zhu for his indefatigable perseverance and his meticulous provision of corroborative evidence. However,

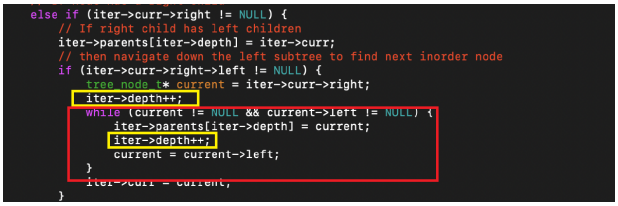
I am willing to solve the tree iterator problem under supervision to prove my innocence.

This is a lose-lose situation for me as I am spending my valuable time fighting for something that will not add any value or serve any purpose. I want to clarify that I am well-versed with programming and **not an average student**. Here is my GitHub. Here is my website. If you view this you can gauge what my level is. I have also made multiple projects in Rust a memory safe modern low-level programming language and a project in Zig the fastest optimizable low-level language.

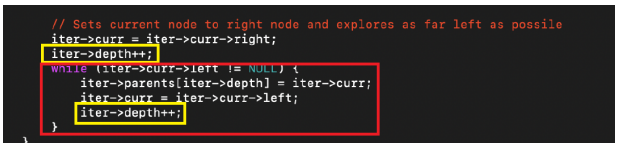
Back to Contents →

2 Analysis of samples provided by Dr. Zhu.

I have reviewed the samples shown by Dr. Zhu and I would call his solutions having ***minor cosmetic differences*** between each other. I liked Dr. Zhu's style of multi-colored boxes so I used it here.
The figure on the **left is Sample1 and the figure on the right is Sample2.**

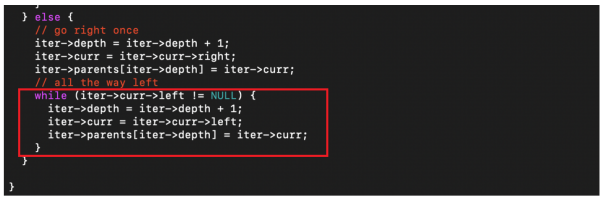


(a) Sample1

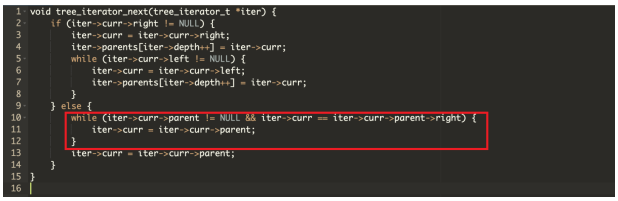


(b) Sample2

Figure 1: Sample1 and Sample2 minor cosmetic differences



(a) Sample3



(b) AI response

Figure 2: Sample3 and AI response minor cosmetic differences

He also uses the complex increment and decrement operators. In fact, Dr. Zhu would have more of a case if I would have used any of the three provided samples.
More facts about the samples are in the next page.

Dr. Zhu's samples incorporate deliberate strategies to write bloated verbose code probably to mimic a student's ignorance and slow down the core function. One could even call Sample 3's `nested ifs` unreadable. Readability of code is often used to criticize junior developers when there is nothing to criticize in the business logic. It is subjective because people read in different ways.

Figure 3: A solution provided by Dr. Zhu

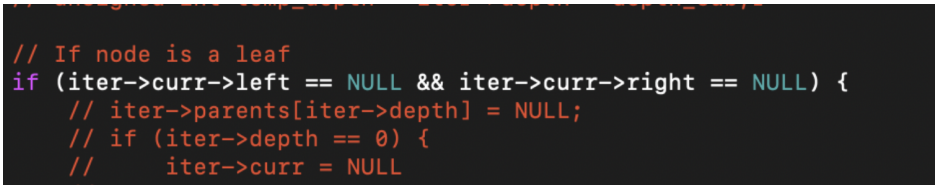


Figure 4: Another solution provided by Dr. Zhu

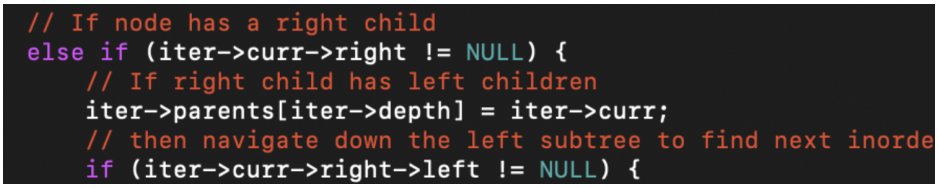


Figure 5: Another solution provided by Dr. Zhu

If these three are not three different ways to write the same thing, I don't know what is.

Also, Comments make the code more readable period.

Back to Contents →

In case there is any confusion about why I used Arrow operators which supposedly are complex structures, Dr. Zhu went over it in the first class(Jan 9)
Here is an excerpt of what he went over:

```
void insertNode(struct Node** head, int newData) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    // Dynamically allocate memory for a new node
    newNode->data = newData;
    // Set the data for the new node
    newNode->next = *head;
    // Point the next of new node to the current head
    *head = newNode;
    // Update the head to point to the new node
}

void displayList(struct Node* head) {
    struct Node* current = head;
    // Start from the head of the list
    while (current != NULL) {
    // Traverse the list until the end is reached
        printf("\%d ", current->data);
        // Print the data of the current node
        current = current->next; // Move to the next node
    }
    printf("\n");
    // Print a newline at the end of the list
}
```

This code has clear indication of using Arrow operators in both the `insertNode` and the `displayList` functions.
Back to Contents →

3 Allegation1: The use of post-increment and pre-decrement operators is cited as evidence, suggesting it's a style generated by ChatGPT and unfamiliar to most students.

I am genuinely confused why Dr. Zhu thinks that increment and decrement operators are complex. In the first lecture of CMPSC 311, Dr. Abutalib Aghayev told us that Java and C have near similar syntax. Anyone who has implemented a for loop in Java or C++ knows what ++ and −− does. It is a standard unary operator which is neither esoteric nor complex.

Here is a lesson of for loop in Java from w3 schools. Please scroll down.

```java
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

We have taken a course called CMPSC 311 which made us familiar with the most intricate and complex C syntax.
Here is an assignment I did for CMPSC 311 which uses similar syntax. In CMPSC 311, Dr. Abutalib Aghayev also told us to add comments after each line to describe what it does calling it a *minor cosmetic difference* is frankly, not only contradictory but also insulting. Dr. Zhu says and I quote:

> "Most experienced programmers avoid this style since it can be confusing and makes the code harder to read."

This statement is contradictory because in the supporting document:

- Dr. Zhu calls students experienced programmers with good judgement and coding style.
- Later compares student skills with a TA who is his 'benchmark' for time standards.

I disagree, ++ and −− are way easier to read as they stand out more than +=1 or +1

Back to Contents →

4 Allegation2: Commonality of Similar `While` Loop Conditions in Programming

It is not uncommon for code to have similar `While` loop conditions, especially when they are designed to achieve the same logical output. *For example if we want to print the numbers from 1 to 100 the code should use a for loop that begins with 1 and ends with 100 or any other interval with 100 iterations.* In programming, particularly in C, certain patterns and constructs are used frequently because they are proven to be effective and efficient. The presence of identical or nearly identical `While` loop conditions across different codebases does not inherently indicate any wrongdoing.
In this particular case, the `While` loop condition in question is both complex and specific, but it is not unique. The structure of the condition is a standard approach to ensure that multiple criteria are met before the loop terminates:

```
while (iter->depth > 0 && iter->curr == iter->parents[iter->depth - 1]->right) {
    iter->curr = iter->parents[--iter->depth];
}
```

This loop condition ensures that the iterator moves up the tree until it finds the next in-order node. The minor difference in the order of the conditions (the equality check) does not alter the functionality of the loop.

Rationale for Unified Loop Conditions

The decision to keep the loop condition unified rather than splitting it into multiple lines or using an additional `if` statement is driven by both logic and style preferences. A unified condition enhances readability by presenting all necessary checks at once, avoiding unnecessary fragmentation of logic. **Splitting the condition into separate statements or lines can sometimes obscure the overall intent of the loop, making it harder to understand at a glance.** This is particularly true for complex conditions where the

relationship between different checks is crucial for understanding the loop's operation.
There was no guideline provided by the instructor on how to structure code. Students cannot anticipate the insrtuctor's exact expectations without clear guidelines. Back to Contents →

5 Allegation3: The ternary notation is yet another uncommon code style that students aren't expected to know or use.

I am genuinely confused why Dr. Zhu thinks that ternary operators are complex. Just like unary operators ternary operators are neither esoteric nor complex. It is a superior and easier way to write if-else code, when there is a single condition. It is a standard in JavaScript and TypeScript which I learned in CMPSC 297 Spring 2023.
Here is a project I did back then which implements ternary operators.
Please check the commit history and find that this was last updated in Summer 2023 well before Spring 2024.Here is the relevant excerpt from the file:

```
const Navbar  = () => {
    const [click, setclick] = useState(false);
    const handleClick = () => setclick(!click);
    ...
        return (
            <div className={color?("header header-bg"):("header")}>
                <Link to ="/">
                    <h1>Portfolio</h1>
                </Link>
                <ul className= {click ? "nav-menu active" : "nav-menu"}>
                    <li>
                        <Link to = "/">Home</Link>
                    </li>
                    <li>
                        <Link to = "/About">About</Link>
                    </li>
                    <li>
                        <Link to = "/Contact">Contact</Link>
                    </li>
                    <li>
                        <Link to = "/Projects">Projects</Link>
                    </li>
                </ul>
                <div className="hamburger" onClick = {handleClick}>
                    {click ? ( <FaTimes size={20} style={{color:"#fff"  }} />) : (  <FaBars size
    ={20} style={{color:"#fff"  }} />)}

                </div>
            </div>
          )


}

export default Navbar
```

Please note the two ternary operators used.
Back to Contents →

6 Allegation4: Implementation in 16 minutes

6.1 I have already established that I am a competent programmer

A TA is not a valid benchmark for programming speed; the instructor has little idea how the student compares with the TA. But, that is irrelevant which my next point explains.

6.2 I would like to remind Dr. Zhu that just because I commited 2 different version within 16 minutes of each other does not mean that I solved them in 16 minutes

I absolutely agree with Dr. Zhu that 16 minutes is unrealistically fast to implement the hardest problem which is the tree iterator function. ***It was possible because I was testing my code in a separate copy of the assignment which I was adding to a private repository***

- to bolster my already considerable GitHub portfolio.
- Although I like the `TDD` test driven development programming architecture, I like to write my own tests instead of reading through a TA written or professor written test so that I can find the exact point of failure.

I shall not make that repository public as I am using version control for this LaTex document, but here is a screenshot.

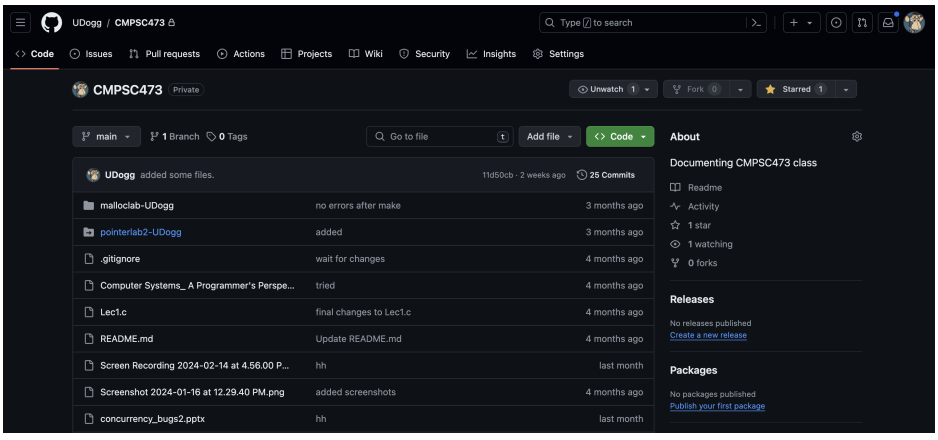


Figure 6: My private repository **notice pointerlab and malloclab directories**

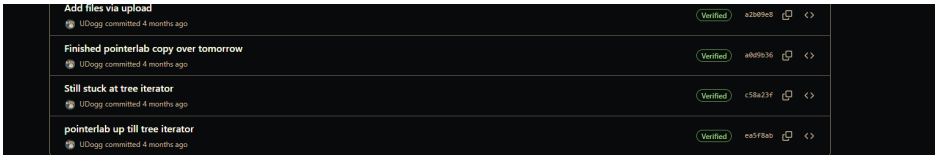


Figure 7: My private repository **notice 3 pointerlab commits**

In fact: I had spent a sizeable 3+ days just trying to implement the tree iterator function. One day for writing the code and 2+ days for debugging. I had written extensive notes on how to solve the function a part of which is the commit message in Dr. Zhu's evidence. I commited early Thursday morning - the morning of the submission date because I did not want to submit under pressure just before the CMPSC473 class.
I normally use Kali Linux for C programming so,

6.3 I spent the 16 minutes doing the following:

- Typing the solution that took me 3+ days to write.
- Testing it using the `make command`
- Committing it.
- Pushing it.

Full Disclosure: This was not my final commit. I was yet to test it in the barebones Vagrant VM recommended by the instructor, and add comments for readability.
Back to Contents →

7 Here are a few points that you should consider while reviewing this case:

In Summary:

- All names and `structs` used were defined by the professor. Just because the names are similar does not mean that I have copied from AI.
- There is consistent style and no unnecessary complications in my code.
- I knew how to iterate through a tree because I had taken CMPSC 360 and CMPSC 465 before. The above classes extensively go through these data structures and ***CODE*** to iterate over these data structures.
- The pointerlab assignment was a tester assignment which was designed to test our knowledge in C.
- The professor said and I quote verbatim:

  ”This assignment is made to test your skills in C, pointers etc”

- I had learned these concepts in CMPSC 311 and was able to get a perfect score in the assignment.
- This Investigation serves no purpose as Spring 2024 is in the past, and Ms. Lori Pifer informed me that I shall not be added to the course due to the timeline of events.
- It is insulting and belittling that the professor does not expect students to write good code and accuses students of cheating when he finds someone doing well.
- Dr. Zhu has recommended these repurcussions for an offense in:

  - A class that I dropped months before the allegation, a class of which the grade has been posted in LionPath.
  - An assignment that **DID NOT COVER ANY TOPICS (apart from arrow operators) DISCUSSED IN CLASS.**
  - The first assignment of the semester with a 2 week timeline.
- I have the utmost respect for Dr. Zhu and his teaching style, but I cannot let this slide.
  - He is more apprehensive about AI than any other professor in any department. See screenshot from RateMyProfessor
  - He would preface each class with a short speech on academic integrity and how he would catch us if we used any AI tool. See screenshot from RateMyProfessor

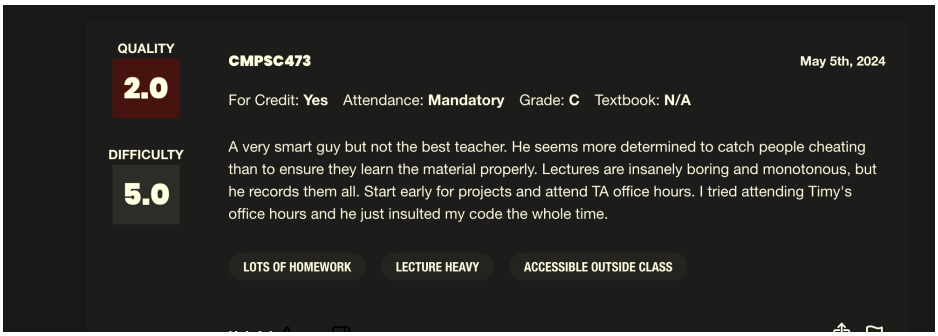


Figure 8: Feedback from other students in RateMyProfessor.

8 Notes for AI Commitee

Ending notes: I am sure I have provided ample proof of my innocence, and demonstrated my own capability. I do not require assistance from AI. I trust the commitee will be wise enough to discern the truth.

Back to Contents →