

---

CSE 331

Software Design & Implementation

Spring 2024

Section 8 – Debugging

---

# Administrivia

---

- HW8 released later today, due Wednesday (5/22) at **11pm**
- Can resubmit as many times as you'd like until the deadline.
  - Use the autograder as a tool if you're not sure if your code/tests have bugs

# Question 1

---

Recall our definition of a binary search tree from HW5:

```
type BST := empty
         | node( $x : \mathbb{Z}$ ,  $S : \text{BST}$ ,  $T : \text{BST}$ ) with conditions A and B
```

Suppose that we wanted to have a way to refer to a specific node in a BST. One way to do so would be to give directions from the root to that node. If we define these types:

```
type Dir := S | T
type Path := List<Dir>
```

then a Path tells you how to get to a particular node where each step along the path (item in the list) would be a direction pointing you to keep going down the left (S) or right (T) branch of the tree.

For example, `cons(S, cons(T, nil))` says to select the “S” (left) child of the parent and then the “T” (right) child of that node, giving us a grand-child of the root node.

- (a) Define a function “`find( $p : \text{Path}$ ,  $T : \text{BST}$ )`” that returns the node (a BST) at the path from the root of  $T$  or undefined if there is no such node.
- (b) Define a function “`remove( $p : \text{Path}$ ,  $T : \text{BST}$ )`” that returns  $T$  except with the node at the given path replaced by empty.

# Question 1a

---

- (a) Define a function “ $\text{find}(p : \text{Path}, T : \text{BST})$ ” that returns the node (a BST) at the path from the root of  $T$  or undefined if there is no such node.

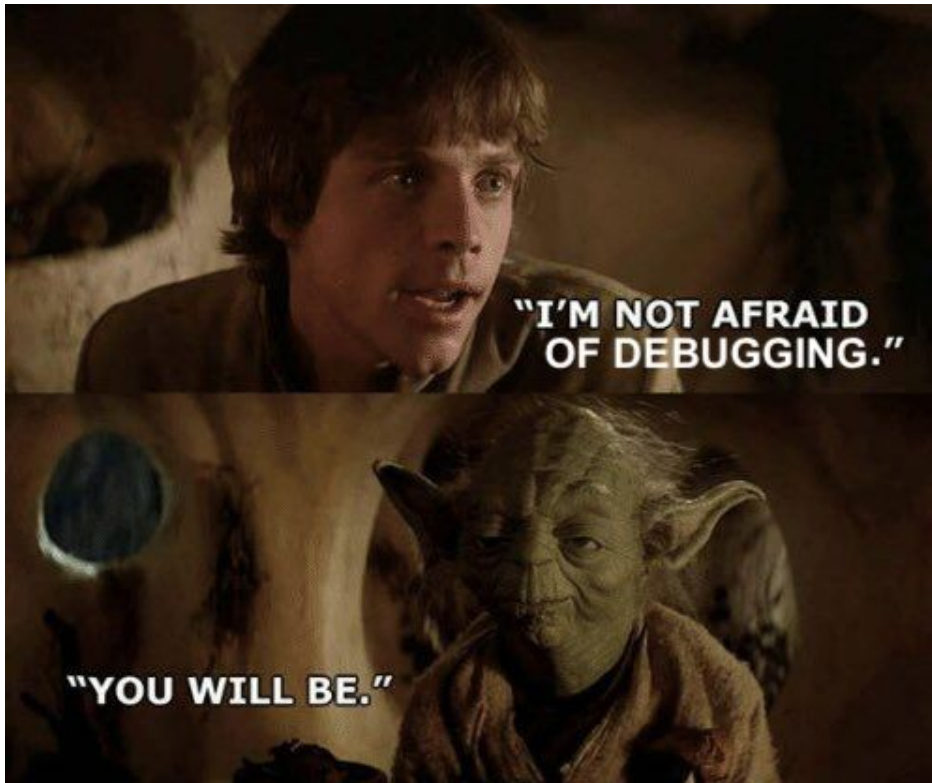
# Question 1b

---

- (b) Define a function “remove( $p$  : Path,  $T$  : BST)” that returns  $T$  except with the node at the given path replaced by empty.

# Debugging!

---



- **Don't stay stuck on the same bug**
  - Continuing isn't helpful
  - Get help!
  - Take a break!
- **Turn bugs into test cases**
  - In case they come back again
- Utilize `console.log()` and the network tab

# Client-Server Communication Debugging Steps

---

- 1. Do you see the request in the Network tab?**
  - the client didn't make the request
- 2. Does the request show a 404 status code?**
  - the URL is wrong (doesn't match any `app.get` / `app.post`)  
**or**  
the query parameters were not encoded properly
- 3. Does the request show a 400 status code?**
  - *your* server rejected the request as invalid
  - look at the body of the response for the error message **or** add `console.log`'s in the server to see what happened
  - the request itself is shown in the Network tab

# Client-Server Communication Debugging Steps

---

- 4. Does the request show a 500 status code?**
  - the server crashed!
  - look in the terminal where you started the server for a stack trace
  
- 5. Does the request say “pending” forever?**
  - your server forgot to call `res.send` to deliver a response
  
- 6. Look for an error message in browser Console**
  - if 1-5 don't apply, then the client got back a response
  - client should print an error message if it doesn't like the response
  - client crashing will show a stack trace



---

# sec-debug coding exercise

debugging practice !!

---