CompSci 201, L12: Debugging and Testing (and more Linked List)
Person in CS: Barbara Liskov

- Turing Award Winner in 2008 for contributions to practical and theoretical foundations of programming language and system design, especially related to data abstraction, fault tolerance, and distributed computing.

- “The advice I give people in general is that you should **figure out what you like to do, and what you can do well**—and the two are not all that dissimilar, because you don’t typically like doing something if you don’t do it well. ... So you should instead watch—**be aware of what you’re doing, and what the opportunities are, and step into what seems right**, and see where it takes you.”
Announcements, Coming up

• Wednesday, 2/22 (today)
  • Project 3: DNA (Linked List) available, due March 6
  • APT Quiz deadline extended to tomorrow Th 2/23.
  • Quiz, no extra grace/late period!

• Monday 2/27
  • Nothing due

• Next Wednesday 3/1
  • APT 5 (linked list problems) due
Ed Discussion Guidelines

Going to review the README
Today’s agenda

1. Wrapping up linked list problems
   • Reverse in-place

2. Testing & Debugging, Concepts & Tools

3. More DIYLinkedList
Canonical Linked List Problem

• How do we reverse nodes in a linked list (without creating a new list)?
  • Go from A->B->C to C->B->A
  • Typical interview style question
  • https://leetcode.com/problems/reverse-linked-list/
  • https://www.hackerrank.com/challenges/reverse-a-linked-list
Methodical Development

• Turn list = ['A', 'B', 'C'] into
  • rev = ['C', 'B', 'A']

• Move one node at a time, *no new nodes*!
  • Iterative/loop solution with invariant

```
rev ───> ||||
     list ───> A ───> B ───> C ───> ||||
```
Invariant to help reason about code

- An invariant is some property that is true each loop guard check (top of the loop)
  - May become false part way through loop
  - Always re-established before guard check

- Example: After k iterations, rev points to the reverse of the first k nodes.
  - before loop iterates at all? rev = null
  - Then at the end we just return rev
one node at a time, assume invariant!

- After 1 iteration: rev is reverse of the first 1 elements
  - list has moved to represent [B,C]
    - So rev represents [A]
- How to move B to front?
- Save a temp reference:
  - Don't lose C-node!

Diagram:
- rev
- list
- temp

2/22/23
rev = [A], list = [B,C], change:
[B,A], [C]

• Pictures and code

1. `temp = list.next` (so we don't lose C)
2. `list.next = rev` (make B point to A)
3. `rev = list` (rev now points to B)
4. `list = temp` (list now points to C)
Working code

```java
public ListNode reverse(ListNode front) {
    ListNode rev = null;
    ListNode list = front;
    while (list != null) {
        ListNode temp = list.next;
        list.next = rev;
        rev = list;
        list = temp;
    }
    return rev;
}
```

B
C
A
rev

list

A
B
C
WOTO

Go to duke.is/n9pg5

Not graded for correctness, just participation.

Try to answer *without* looking back at slides and notes.

But do talk to your neighbors!
Consider the reverse method. If front is a ListNode with `front.next == null`, then `reverse(front)` will...

```java
public static ListNode reverse(ListNode front) {
    ListNode rev = null;
    ListNode list = front;
    while (list != null) {
        ListNode temp = list.next;
        list.next = rev;
        rev = list;
        list = temp;
    }
    return rev;
}
```

- Encounter a NPE
- Return null
- Return the same node
- Return a copy of the same node
Same reverse method. We claimed that the following is a loop invariant:
"After k iterations, rev points to the reverse of the first k nodes."

For an input list with N elements, which of the following is also a loop invariant for this method? "After k iterations, list..." *

```java
10    public static ListNode reverse(ListNode
11         ListNode rev = null;
12         ListNode list = front;
13     while (list != null) {
14         ListNode temp = list.next;
15         list.next = rev;
16         rev = list;
17         list = temp;
18     }
19    return rev;
20 }
```

- Points to the first k elements of the input list
- Points to the first N-k elements of the input list
- Points to the last k elements of the input list
- Points to the last N-k elements of the input list
Same reverse method. Suppose we have a linked list as follows:

list --> 1 --> 2 --> 3 --> null

What will be returned by reverse(list.next)? *

```java
public static ListNode reverse(ListNode list)
{
    ListNode rev = null;
    ListNode list = front;
    while (list != null) {
        ListNode temp = list.next;
        list.next = rev;
        rev = list;
        list = temp;
    }
    return rev;
}
```

- 3 --> null
- 3 --> 2 --> null
- 3 --> 2 --> 1 --> null
- Nothing, null pointer exception occurs
Same reverse method. Suppose we have a linked list as follows:

list --> 1 --> 2 --> 3 --> null

In the main program. What will list (also in the main program) be after running reverse(list.next)? *

```java
public static ListNode reverse(ListNode list) {
    ListNode rev = null;
    ListNode list = front;
    while (list != null) {
        ListNode temp = list.next;
        list.next = rev;
        rev = list;
        list = temp;
    }
    return rev;
}
```

- null
- 1 --> null
- 1 --> 2 --> null
- 1 --> 2 --> 3 --> null
- Nothing, null pointer exception occurs
Testing and Debugging
An Algorithmic Problem-Solving Process: UPIC

- **Understand**: Understand the problem you're trying to solve. Read carefully, do examples.
- **Plan**: Generalize insights, develop an algorithm.
- **Implement**: Translate the algorithm into code. Only doing a lot of actual programming in this step!
- **Correctness**: Test and debug to verify and fix the code.
Not really a linear process

So, something is not correct. Could be...

1. The plan (algorithm) did not match the understanding.
2. The implementation does not match the plan.
3. The understanding was not correct.
First approach to correctness

- Natural temptation to rely on reading source code to verify correctness.
- Like editing an essay for a class, read and check that it makes sense, look for typos.
- But…
Code is complex and interrelated

Miss something in your essay? The rest of the essay may still make sense?

One thing wrong in the code? Could prevent the whole program from functioning. And code gets complicated!

Working C code from 1998 contest, see wikipedia
A tale of two programmers...

Too confident
“I’m amazing at programming, I don’t need to test my code because I know it’s correct.”

Low confidence
“My code doesn’t work, that must be because I’m personally bad at this. There is no way I could figure this out myself.”

The beginning of a security vulnerability, broken app, ...

Mistaken expectations, Feeling helpless, not sure what to do
What is testing?

Verifying that an implementation functions as expected.

• What is functionality is expected?
• Given an input, what output is expected?

Can test at multiple levels: single method (unit), class (integration), whole project (integration/functionality), ...

*Black box testing* (can run program, can’t see source code) and *white box testing* (access to source code).
SandwichBar APT Example

Given:

• String[] available, a list of ingredients the sandwich bar can use, and
• String[] orders, the types of sandwiches I like, in order of preference (most preferred first)

return the 0-based index of the sandwich I will buy. If the bar can make no sandwiches I like, return -1.

Example:

• available: { "ham", "cheese", "mustard" }  
• orders: { "ham cheese" }  
• Should return: 0
The first test: the compiler

• Compiler performs static analysis; check for errors detectable in the source code before running.
  • Often type errors (e.g., trying to assign a String to an int, trying to treat an Array as a list, ...)

```java
public int whichOrder(String[] available, String[] orders) {
    for (int i = 0; i < orders.length; i++) {
        if (canMake(available, orders[i])) {
            return orders[i];
        }
    }
}
```

Type mismatch: cannot convert from String to int
Manual test

• Given an input, what is the expected output?
• Run program with expected input. What do you get?

```java
25     public static void main(String[] args) {
26         String[] testAvailable = { "ham", "cheese", "mustard" };
27         String[] testOrders = { "ham cheese" };
28         SandwichBar testInstance = new SandwichBar();
29         int testResult = testInstance.whichOrder(testAvailable, testOrders);
30         System.out.println(testResult);
31     }
```

I expect the code to return 0, example from before. And it does! My solution must work!
How many tests are enough?

- Can never have enough tests to guarantee correctness, but...
- More and more diverse tests can help increase confidence.

That’s not right, I can’t make a ham and cheese sandwich without ham...
Automated testing?

For when you want to run many tests without doing it manually one at a time...automate it!

Ways you use automated testing in 201:

• **JUnit tests** – JUnit is a popular external library, no built-in standard library unit testing in Java.

• Gradescope autograder

• APT server

In professional software development? You also write the tests!
Test early, test small, test often

- **Unit testing**: Term for tests conducted on the smallest *units* of code that take inputs and produce outputs.
  - In Java, typically methods, preferably short ones (10-20 lines). Test as soon as you write, don’t wait!
  - Method getting too complex? Helper method!

```java
public void testSize() {
    for (String s : strs) {
        final IDnaStrand strand = assertTimeout(Duration.ofMillis(10000), ()->{
            IDnaStrand str = getNewStrand(s);
            return str;
        });
        assertEquals(s.length(), strand.size(), "This test checks if .size() returns the correct value" + " for basic cases. Your code did not return the correct .size() for strand " + s);
    }
}
```

- Expected output
- What your size() method returns
- Message if condition fails
Debugging loop:

1. Detect unexpected behavior through testing.
2. Isolate *cause* of unexpected behavior.
3. Change implementation.
4. Test again.
How to isolate the cause of unexpected behavior

• Want to identify the first point of divergence from expected behavior.
  • May have started long before your test result!

• Try to answer the question:
  • What is the first line of code in which method of which class that first did something different than I expected?
  • Never fixate on line 30 if you’re not sure lines 1-29 are working.
Debugging Methods

• Three common methods:
  • Examine code and small examples by hand
  • Add print statements to code
  • Use a debugger tool

• We have already seen the basic debugger tool built into an extension on your visual studio code. Will review in detail today.
Debugger tool

- Instead of run? Choose debug!
- Walk through execution of program line by line.
- See current state of all variables line by line.

To customize Run and Debug create a launch.json file.

Show all automatic debug configurations.
Set a breakpoint

• Start by setting a *breakpoint* in your code.
• Says “run the program until the first time this line executes, then pause to step line by line.”
• If you want to go line by line from the beginning? Set to first line in main.
Debug options

Will see a menu like this:

• **Continue**: Go to next breakpoint

• **Step over**: Execute line, go to next. Run whole methods.

• **Step into**: Same as over *unless method call*. Steps into methods, jumping to first line of method code.

• **Step out**: Break out of method back to where called

• **Restart**: Start over again at first breakpoint

• **Stop**: Stop debugging session
State of program

```java
5 public class SandwichBar {
6     public int whichOrder(String[] available, String[] orders) {
7         for (int i = 0; i < orders.length; i++) {
8             if (canMake(available, orders[i])) {

VARIABLES

Local

available: String[3]@15
  0: "cheese"
  1: "mustard"
  2: "lettuce"

orders: String[4]@16
  0: "cheese ham"
  1: "cheese mustard lettuce"
  2: "ketchup"
  3: "beer"

- Can see all values of all local variables while executing at highlighted line.
- Can step through to determine first time values diverge from expectations.
```
Debugging linked list?

- Appears as a nested “list” of object references.
- Expand one node at a time.

```java
public static void main(String[] args) {
    args = String[0]@8
    int[] myNums = {2, 0, 1}; myNums = int[3]@10
    ListNode myNumsList = listFromArray(myNums); myNumsList =
    printList(myNumsList); myNumsList = ListNode@12
    System.out.println(getVal(myNumsList, 1));
}
```
Want something more visual? [pythontutor.com/java.html]

Can use if you need to visualize stepping through some pointer code.
Debugging reflection

Goal is to become a more *active* and *empowered* tester and debugger.

- Build confidence *as you develop*.
- Take *active* steps to isolate the problem
- Test, use the debugger, gather data, reason about it
- Less time staring at the code, feeling frustrated
More DIYLinkedList

Live Coding

• Writing unit tests?
• Add to arbitrary index?
• Efficient iterator?