CS 1332 Recitation Worksheet – Week 0

Scavenger Hunt, Debugger Intro, Data Structures + Big O Review

This worksheet covers material from this week's recitation. It is meant to be additional exercise for your benefit and will not be graded. Feel free to collaborate with other students and ask TAs for help.

Distribution of this worksheet is not permitted.

Scavenger Hunt 🧭

Navigate through your CS 1332 Canvas page and answer the following questions. If you do not see the course on your Canvas page, you may team up with a buddy next to you!

1. Where are CS 1332 office hours located, and when do they start?
   You can find your favorite TAs at the **College of Computing (CoC), in Room 210**! Office hours are scheduled to start on **Tuesday, August 19th**.

2. How many exams will you have in this course (excluding the final)?
   There are **three exams** scheduled throughout the semester for this course, excluding the final.

3. When is the first homework assignment due?
   Homework 1 is due on **August 27th** at **11:59:00 PM**!

   a. Which data structure does the first homework assignment cover? The first homework covers the **ArrayList** data structure.

4. What percentage of your total grade is each exam (excluding the final)?
   Each exam counts for **20%** of your total grade.

5. Where can you find videos about all topics covered in the course? You can find helpful instructional videos under the **"Modules"** tab on Canvas.

6. Where can you find worksheets, practice exams, and live coding files from recitation? Recitation materials will be posted under **Files → Resources → Recitation Materials** on Canvas.

7. On which platform can you ask TAs questions asynchronously (i.e. outside of office hours)? You can always ask questions on **Piazza**, and TAs will help you out!

    a. **True or False:** When asking public questions about the homework on this platform, I am allowed to add Java code to my post. **False**. Do NOT post any code from your assignments when making **public posts** on Piazza. If you *really* need help with debugging, please make a **private post** with your code.

    b. Name 3 TA-recommended ways to be successful in CS 1332.
        i. **Start early.** You get at least an entire week for homeworks. Do not wait until the last minute.
        ii. **Learn to use/write JUnits as early as possible.** We give out a few tests, but it won't be enough. *(take a look at the JUnits activity later in this worksheet!)*
        iii. **Efficiency is key.** You can lose up to 15 points on each homework just for not being efficient (even if it's correct). So, learn how to analyze the efficiency of your code.

8. What are some reasons why you may get points taken off on your homework?
    a. **Checkstyle:** If thousands of people are using your code, you want your code to be easy to read for them, so we have a checkstyle to help keep things consistent.
    b. **Exception messages:** You don't want to run into an IndexOutOfBoundsException that just says "Index is out of bounds." So be descriptive in your own error messages. Help the user. Say something like "You tried to input " + input" + ", but the bounds are between 0 and " + size.
    c. **Generics:** We will talk about generics, but when you don't use generics EVERYWHERE, you get a warning. That means that anytime anyone uses your code, they will get a warning. You don't want them to get a warning just by using your data structure.
    d. **Print statements:** Should be obvious. You don't want everyone using your code to see your debug messages.

9. What is the official visualization tool for the course? **https://csvistool.com/** – you should check it out! It's an amazing resource 😀

10. **True or False:** Completing the Syllabus Quiz is a way of earning extra credit in the course.

    True! It's really easy, and this Scavenger Hunt should help – you should do it!

11. Which IDE do TAs recommend using for CS 1332? We recommend using **IntelliJ** – most TAs are familiar with this IDE and can help you out more effectively during office hours!

12. Which Java version will you be using to write Java programs? You will need to use **JDK 17** to complete programming assignments.

Activity: JUnits and the IntelliJ Debugger

It is good practice to write tests so you can catch and debug errors in your code. This is where JUnits and the IntelliJ debugger become really useful! Once you've downloaded IntelliJ, create a project using the provided recitation files: ArrayWrapper.java and ArrayWrapperTests.java.

Make sure to *run* the JUnits by clicking the  icon!

1. Complete the `replaceFront()` test in ArrayWrapperTests.java. You can use the provided code for `testReplaceMiddle()` as an example!
2. Complete the `testReplaceLargeIndex()` test in ArrayWrapperTests.java. *Hint: a `try-catch` block may be helpful here!*
3. Oh no! Our `testMax()` and `testRange()` JUnits are not passing! Use the IntelliJ debugger to investigate where our code is wrong and fix the error(s). *Hint: read the code for `testMax()`, setting breakpoints to step through the code.*
4. There is a **hidden inefficiency** with one of the methods in ArrayWrapper.java. What is it, and how do we fix it?

Check out **ArrayWrapper_IMPL.java** and **ArrayWrapperTests_IMPL.java** for solutions to this activity!

Data Structures + Big O Review

1. What is the difference between a data structure and an ADT?
    - An **ADT** represents a set of features and properties that define some useful objects (e.g. Lists, Stacks, Queues) It is analogous to a Java Interface.
    - A **Data Structure** is a concrete implementation that typically *implements* an ADT (e.g. ArrayList, Singly LinkedList, Doubly LinkedList).
    - One analogy we can think of is dogs!
        - Think of a "dog" as being an **ADT** – when we think of a dog, we know that dogs bark, ask for treats, and like to play fetch.
        - Certain *types* of dogs – like golden retrievers or poodles – would be the actual **data structures**, or implementations of our "dog" ADT.

2. What do the terms “time complexity” and “space complexity” mean?
    - Time complexity depicts how the **running time** of an algorithm/operation on a data structure grows with the size of the input.
        - Usually the **size** of the data structure, e.g. the number of entries in an ArrayList, enables the measurement of time complexity.
    - Similarly, space complexity is how much **extra space** an algorithm/operation on a data structure uses as input grows.
        - You’ll see more of this when we study algorithms!

3. What is the time complexity of the following code snippet?

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.println(i + " " + j);
    }
}
```

Solution: ***O(n²)***

4. What is the time complexity of the following code snippet?

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < 1000; j++) {
        System.out.println(i + " " + j);
    }
}
```

Solution: ***O(1000n)*** = ***O(n)***

5. Discuss the difference between the two code snippets above. Why do they have different time complexities?

   The first snippet’s inner loop scales with ***n***, resulting in a quadratic overall runtime complexity. The second snippet’s inner loop has a fixed number of iterations, so the total complexity remains linear.

6. Explain the concept of amortized time.

   Amortized time is used when the worst-case runtime is pessimistic, represents an expensive operation, and occurs very rarely.
    - It is usually along the lines of: “Usually this will be ***O(1)***, but every so often it will run in to allow more operations.”
    - One example is the **resize case** when adding to an ArrayList, which will run in ***O(1)**** (amortized) time!
        - This is because we will only rarely need to resize our entire list when adding a new element.