Lecture 09 - Midterm 1 Review (last updated: 02/13/23)

Note to the reader: This document is what will be used for my lecture on 2/14/23 for COMS W1004. While I intend to publish this afterwards, some of the notes and comments you see may not make sense to you as they are there to serve as general reminders for myself to make sure I cover what I want to cover in the limited time that I have. Best of luck to those who read on their exam, I know you'll do well.

## Exam Overview and other Logistics

Reminder that the Exam is Thursday 2/16, it will be offered online on Courseworks, but an in person setting will be provided - meaning you can bring your laptop to the lecture hall and take your test there - the exam is closed everything, closed book, closed notes, closed internet, closed terminal etc … Any student caught violating these will immediately be reported to SCCS and they will take it from there. Courseworks has built in tools in place that tracks your activity so do not risk your entire academic career on a silly exam.

The exam has 30 multiple choice questions and you will have 75 minutes from the moment you begin the exam to finish the exam. If you are a student who is set up with Disability Services, the proper additions should have been made for your exam.

Homework 3 was due yesterday evening and Homework 4 is due Monday night.

If you wish to ask a question during the lecture today, post on the Lecture Question thread and I will answer those questions periodically as I have time.

## Question 01

This question should be familiar as it is a variation of a problem from HW 1. The goal to solving this problem is setting up the equation properly and then solving for the missing variable:

$n(n-1)/2 + [\text{floor}(\lg N)+1]s = ns$ | Plug in $n = 15$ and solve for $s$

## Question 02

This question should be trivial; linear search runs in linear time and grows linearly as n increases.

Question 03
---
This is just the standard application of selection sort like you did for your homework; Remember Selection Sort selects the smallest element in the collection and swaps it with the current element.

0. 9 2 20 3 6 4 21 | Current element is 9 smallest element is 2, swap
1. 2 9 20 3 6 4 21 | Current element is 9 smallest element is 3, swap
2. 2 3 20 9 6 4 21 | Current element is 20 smallest element is 4, swap
3. 2 3 4 9 6 20 21 | Current element is 9 smallest element is 6, swap
4. 2 3 4 6 9 20 21 | Current element is 9 smallest element is 9, NA
5. 2 3 4 6 9 20 21 | Current element is 20 smallest element is 20, NA
6. 2 3 4 6 9 20 21 | Current element is 21 smallest element is 21, NA
7. 2 3 4 6 9 20 21 | Finished

We can use this to find what the list looks like after the third iteration!

Question 04
---
This is just the standard application of insertion sort like you did for your homework; Remember Insertion Sort inserts the current element into its proper spot in the sorted portion of the array. Use the box method!

0. 9 ] 2 20 3 6 4 21 | Current element is 2, compare with 9 and insert
1. 2 9 ] 20 3 6 4 21 | Current element is 20, compare with 9 and insert
2. 2 9 20 ] 3 6 4 21 | Current element is 3, compare with 20,9,2 and insert
3. 2 3 9 20 ] 6 4 21 | Current element is 6, compare with 20,9,3 and insert
4. 2 3 6 9 20 ] 4 21 | Current element is 4, compare with 20,9,6,3 and insert
5. 2 3 4 6 9 20 ] 21 | Current element is 21, compare with 20 and insert

We can use this to find what the list looks like after the second iteration!

Question 05
---
If you format and write down your run through insertion sort like I did you should be able to easily answer this question!

Question 06
---
Some would be tempted to count the number of comparisons individually, but remember since selection sort takes the same amount of time to run regardless of how the elements of the collection are arranged just use the efficiency function for selection sort and this problem becomes trivial.

Question 07
---
Standard use of Binary search; Take the sorted list from one of the previous questions and perform binary search.

2 3 4 6 9 20 21 | Select the middle element, compare it with the target and update the markers accordingly
2 3 4 6 9 20 21 | Select the middle element, compare it with the target and update the markers accordingly
2 3 4 6 9 20 21 | Select the middle element, compare it with the target and realize you have found your target.

You can now answer the question!

Question 08
---
This is just a standard recall problem, you either know it or you don't. It is easiest to remember that int is allocated 4 bytes of memory and that the biggest is *double* that amount so 2*4 is 8 bytes.

Question 09
---
Simple problem involving methods of the String Class, remember substring takes in two integer values, a and b, and returns a string starting at index a and ending at index b-1. Remember the range is this [a,b) not [a,b] knowing that we can answer this question!

Consider that String indexing begins at 0: That means the String "Columbia" has a valid index range of [0,7] anything beyond this in either direction will yield you a StringIndexOutOfBounds Exception. Knowing all of this, the answer to the question should be easy to derive.

Question 10
---
This question relies on your understanding of Explicit vs Implicit parameters, remember explicit parameters are those explicitly listed in the method call, so given the statement myAccount.withdraw(amount); the answer should come naturally.

Question 11
---
The logic for this question is the same but reversed when compared to Question 10. Remember implicit parameters are all arguments that are not

explicitly listed in the method call. Implicit parameters can only be present when an Object Instance is calling a method, which in this problem is definitely the case

Question 12
---
This question tests your ability to read and interpret Java Code. Notice first that we are looping through the String s, in a nested loop structure. I.e. for each character in s we will loop through s again. We then compare the static character from the outer loop to the constantly updating character from the inner loop, if they match we increase the counter by 1. At the very end we concatenate the counter to s2 and then start again.

This should let you know that the counter represents how many instances of a character exist in the String. From this we can easily conclude what the resulting String looks like.

Question 13
---
This question is easy if you make the right connections, if c only updates when the expression is true and we know the resulting string, simply add up the values in the String s2 to get your final answer.

Intermission for Questions
---
At this time I will now check the EDStem Thread for any questions before moving on to the additional topics of interest. At this point I hope to be a little bit before the halfway mark of the lecture.

Topic 1: Big-O and Analysis of Algorithms
---
So far in this course you have seen 4 algorithms that you are responsible for knowing the algorithmic complexities of. For some algorithms, mainly those which could be described as non-adaptive, there is an efficiency function T(n) that can be derived to tell us how efficient the algorithms are. You know that for Selection Sort: **T(n) = n(n-1)/2** but for some algorithms in the real world the expressions for T(n) are either too cumbersome to write out or (at least for purposes of 1004) do not exist. So what is a more effective way to compare the complexities of algorithms? Well Computer Scientists developed a notation known as Big-O. Big-O in reality compares the orders of growth of the algorithms but Big-O also provides a convenient way to approximate the efficiency of algorithms to the point that we now have a standardized way to compare algorithms to one

another. At this point in the course here are the Big-Os for all the algorithms you are supposed to know:

1. Linear Search: O(n)
2. Binary Search: $O(\log_2 N)$
3. Selection Sort and Insertion Sort: $O(N^2)$

## Topic 2: Java Syntax, Classes and Objects

Classes are the main structural unit for Java programs, pretty much all the code you write for any program will be contained within a Class (with the exception of import statements and packages). Objects are instances of Classes, you can think of Objects as the final product that are built using a blueprint, therefore you can think of Classes as blueprints for Objects. The Constructor can be thought of as the assembly line, it is where the object is constructed. While the actual attributes can change from Object to Object you cannot add or remove attributes without modifying the blueprint (class). There is much more that can be said on this topic, so to keep it concise I will stop here, please see the Lecture Reviews for more information on this. To wrap up this topic though, I will live code a Java class from scratch, in the way I recommend you build all of your classes.

## Topic 3: Modular Arithmetic and Boolean Expressions

A lot of students often get confused by the modulus - % - from Java and how to conceptually think about it. Consider the following expression: m % n; Now think of a clock with n marks on it, every full revolution you can make around that clock represents an even division between m and n. But there will come a time when m will become 0, at this point when m is 0 whatever mark around our hypothetically clock we are at is what will be returned to the user. m % n can return any value in the interval [0,n).

Boolean expressions are a critical component of programming in Java. You write them whenever you write a conditional or a loop. Let's quickly review booleans. A boolean is a primitive data type in Java where the accepted values are true or false. So in Java syntax here is how we define a boolean:

```
boolean isRainy = false;
```

There are some operations we can perform on booleans too! Take the following statement for example:

```
System.out.println(!isRainy); //prints true to the terminal
```

The ! is the not operator, it takes the inverse of the boolean value provided

If we assume we have another boolean declared and initialized called isCold. We can generate a more complex boolean:

```
boolean isRainyOrCold = isRainy || isCold;
boolean isRainyAndCold = isRainy && isCold;
```

The || is the logical OR operator in Java, the resulting boolean is true if at least one of the components is true. It is only false if both components are false.

The && is the logical && operator in Java, the resulting boolean is true iff both components are true. It is false if at least one of the components is false.

## Final Intermission for Questions

At this time I will now check the EDStem Thread for any questions before moving on to the additional topics of interest. At this point I hope to be a little bit before the end of the lecture.

## Concluding Remarks

Thank you for reading and/or attending this lecture. I will be holding office hours tonight from 6-8pm in the usual location and on Wednesday from 4-6pm for further discussion of topics to help you for your exam. These sessions are for exam questions only; I will not discuss HW4 during either of those sessions and will resume aid for HW4 on Thursday. Best of luck to you this week!