# Creative Project 0: Warm Up and Review

## Specification

# Overview

This assignment is intended to be a review and warm up for CSE 123. It will require you to use the skills and concepts that you should be familiar with from your prior programming experience. It will also serve as an introduction to your first IDE, Visual Studio Code. This is designed to help everyone review and practice the programming skills that will be necessary to succeed in CSE 123. While we don't necessarily expect everyone to find this assignment *easy*, if you find yourself having major difficulties with any of the content, please contact the course staff to get support!

# □□ Learning Objectives

By completing this assignment, students will demonstrate their ability to:

- Use the Visual Studio debugger to go through a program line-by-line
- Predict the behavior and results of executing a Java program that includes complex and/or compound data
- Identify errors in a Java program's state or behavior, and implement fixes for identified errors
- Write functionally correct Java programs that meet a provided specification using compound data types
- Write functionally correct Java classes to represent new, compound data types

# □ Assignment Structure

Unlike most future assignments in CSE 123, this assignment will consist of a series of individual questions and problems. By focusing on a few separate and slightly smaller programming problems, we can help you target your practice on the programming skills that will set you up for success in our course.

To complete this assignment, you should go to each slide and complete the task(s). For quiz slides (indicated by a blue clipboard icon), provide an answer to each question. For coding challenge slides (indicated by a yellow angle brackets icon), upload your code to the workspace. When you have successfully completed each slide, you will see the dot next to the slide title fill in. The assignment is complete when you have a filled-in dot for every slide, including the "Final Submission" slide. The problems can be worked on in any order.

# Feeling Stuck?

While we expect this assignment to be review, it's still OK if you find this assignment a bit challenging! Remember that learning is a challenging process, and you don't have to do it alone!

- You can visit the Introductory Programming Lab (IPL) to talk with a TA about programming concepts or get help on assignments.
- You can stop by instructor office hours to discuss course concepts or get help on assignments or discuss the course in general.
- You can post questions on the discussion board! You can make questions public (anyone can see them) or private (only course staff can see them). This is a great way to asynchronously get help on an assignment or ask questions about the course.

It is OK to get stuck and feel challenged by this assignment. However, note that this is intended to be a warm-up for the type of programming we will be doing for the rest of the quarter, and the tasks we will be solving in future weeks will be more complex than these problems and rely on a solid grasp of the skills practiced in this assignment. If you feel like you cannot do this assignment at all, we recommend reaching out to the course instructor (jrw12@cs.washington.edu) or the CSE undergrad advisors (ugrad-adviser@cs.washington.edu) to discuss more about academic planning and which programming course might be a good fit for your goals.

# Submission

When you are ready to submit, go to the "Final Submission " slide, read the statement and fill in the box, then click "Submit" in the upper-right corner. You may submit as many times as you want until the due date.

You can see your previous submissions by clicking the three dots icon in the upper-right and selecting "Submissions and Grades." By default, we will grade your latest submission from before the deadline. However, if you would like us to grade a different submission, you can select that submission on the left side of the window and click "Set final." Note that we will not grade any submission made after the deadline-- if you mark a submission after the deadline as final, we will grade your most-recent on-time submission instead.

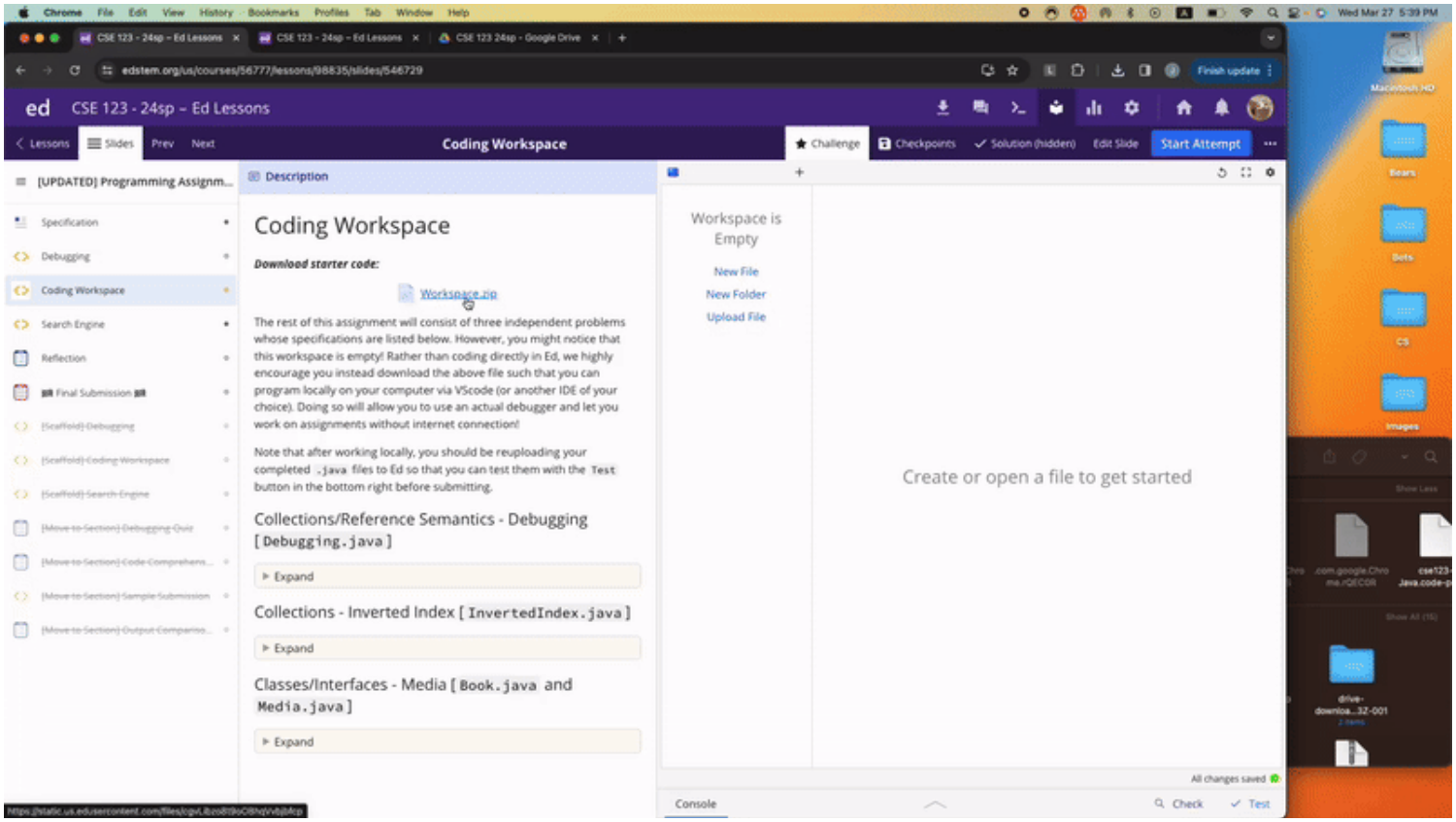
Please make sure you are familiar with the resources and policies outlined in the syllabus and the programming assignments page.

## Recommended Development Process

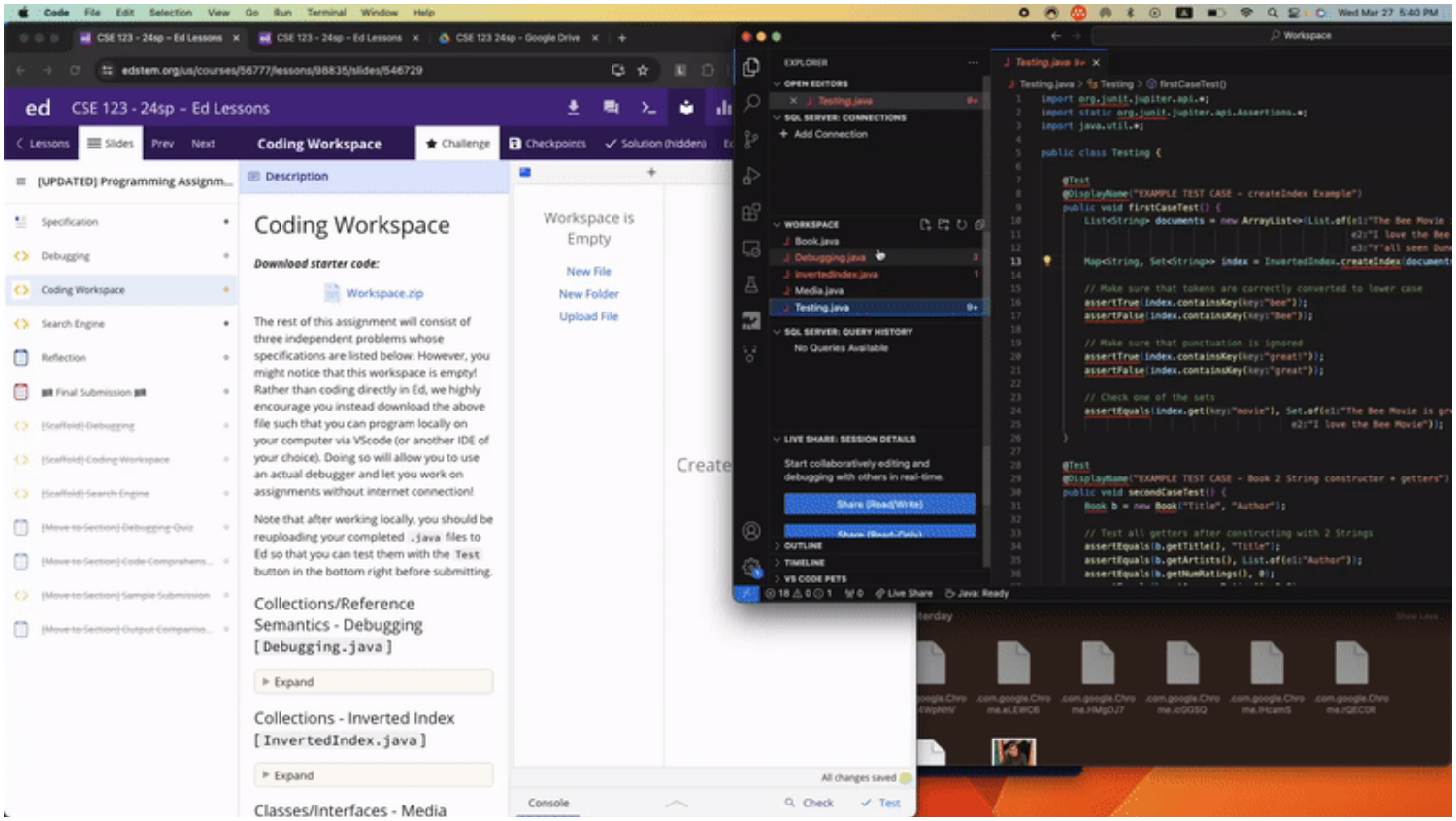
The general process we're expecting you to follow when working on and submitting HW assignments is slightly different from previous courses now that you have an IDE (VSCode)! You should

1. Download the provided `.zip` file
2. Find it in `Finder` / `File Explorer`
3. Unzip it to get the folder with relevant files.
4. In VSCode, click File > Open Folder, and select the recently unzipped folder to open it!

This process is outlined in the following `.gif` (mac shown, but the process would be the same for windows with File Explorer)




Then, once you've finished, you should re-upload your code by copy-pasting or drag-dropping the files back into Ed! At this point you can run tests via the `Test` button.


CSE 123 - 24sp – Ed Lessons
edstem.org/us/courses/56777/lessons/98835/slides/546729
ed CSE 123 - 24sp – Ed Lessons
Lessons
Slides
Prev
Next
Coding Workspace
Challenge
Checkpoints
Solution (hidden)
[UPDATED] Programming Assignm...
Specification
Debugging
Coding Workspace
Search Engine
Reflection
Final Submission
Description
Coding Workspace
Download starter code:
Workspace.zip
The rest of this assignment will consist of three independent problems whose specifications are listed below. However, you might notice that this workspace is empty! Rather than coding directly in Ed, we highly encourage you instead download the above file such that you can program locally on your computer via VScode (or another IDE of your choice). Doing so will allow you to use an actual debugger and let you work on assignments without internet connection!
Note that after working locally, you should be reuploading your completed .java files to Ed so that you can test them with the Test button in the bottom right before submitting.
Collections/Reference Semantics - Debugging [Debugging.java]
Expand
Collections - Inverted Index [InvertedIndex.java]
Expand
Classes/Interfaces - Media
Workspace is Empty
New File
New Folder
Upload File
All changes saved
Console
Check
Test
Testing.java
Book.java
Debugging.java
InvertedIndex.java
Media.java
Testing.java
No Queries Available
Start collaboratively editing and debugging with others in real-time.
Share (Read/Write)
Live Share
Java: Ready

# Debugging

# Background

Before you begin this assignment, you need to download and set up Visual Studio Code with the CSE 123 Profile! If you have not done so already (either in section or by yourself), you can do this here!

Once you have set up the Visual Studio "IDE" (integrated development environment – an application that helps you code), download and open the file below. You will have to determine the correct code needed to defuse the bomb by using the debugger (more info the the following video / on the course website). Time is of the essence, and the fate of the world rests in your hands!

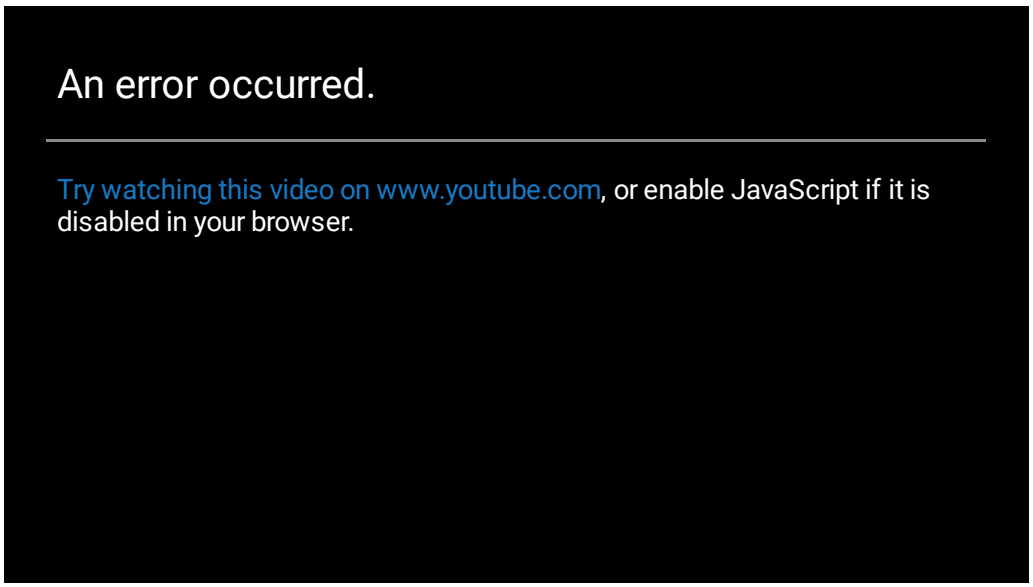
***Download starter code:***

C0_Debugging.zip

**NOTE:** You should only need to update `line 30` of the provided file: `defuse("00000");` to the defusal code you determine using the VSCode debugger!

**WARNING:** We're trying our best to encourage you to use the VSCode debugger here, so you may find that printlns don't actually print anything! This is expected behavior :)

The following video will walk you through some of the useful features of the VSCode debugger. Watch the video, and then defuse the bomb!




(https://youtu.be/_UjyHwBZT6k)

## Coding Workspace

***Download starter code:***

C0_Workspace.zip

The next part of this assignment will consist of two independent problems whose specifications are listed below. However, you might notice that this workspace is empty! Rather than coding directly in Ed, we highly encourage you to instead download the above file such that you can program locally on your computer via VScode (or another IDE of your choice). Doing so will allow you to use an actual debugger and will let you work on assignments without internet connection!

Note that after working locally, you should be reuploading your completed `.java` files to Ed so that you can test them with the `Test` button in the bottom right before submitting.

## Classes/Interfaces - Media [`Book.java` and `Media.java`]

▼ Expand

Write a Java class called `Book` that implements the provided `Media` interface and represents a book. For books, the artists are considered to be the author(s).

Your class should have three constructors:

```
public Book(String title, String author)
```

- Creates a book with the provided title and single author.

```
public Book(String title, List<String> authors)
```

- Creates a book with the provided title and multiple authors.

```
public Book(String title, String author, Scanner sc)
```

- Creates a book with the provided title, author, and scanner connected to a file containing all words within the book

**NOTE:** Any book objects constructed not using the `Scanner` constructor should return an empty list if `.getWords()` is called on them

The title and author(s) should *not* be able to be modified by a client after creation.

**NOTE:** When writing your class, be sure to follow all guidelines in the **Code Quality Guide** and **Commenting Guide**.

Any additional helper methods created, but not specified in the spec, should be declared ***private***.

## Collections - Inverted Index [ `InvertedIndex.java` ]

▼ Expand

Write a method called `createIndex` that creates an inverted index for a list of documents. Your method should take one argument, a list of `Media` objects.

**NOTE:** You can call `.getWords()` on a `Media` object as described within the interface to get a list of all words within the document!

Your method should return a map where the keys are individual words that appear within each of the `Media` objects and the values are sets of `Media` objects in which those words appear.

For a more concrete example, suppose the following documents list:

```
docs = [One by [a1], Two by [a2], Three by [a3]]
```

contained the following words

```
docs.get(0).getWords() = [Raiders, of, the, Lost, Ark]
docs.get(1).getWords() = [The, Temple, of, Doom]
docs.get(2).getWords() = [The, Last, Crusade]
```

In this case, an inverted index would return the following map:

```
{ark=[One by [a1]], crusade=[Three by [a3]], doom=[Two by [a2]], last=[Three by [a3]],
 lost=[One by [a1]], of=[Two by [a2], One by [a1]], raiders=[One by [a1]],
 temple=[Two by [a2]], the=[Two by [a2], Three by [a3], One by [a1]]}
```

The keys of the returned map should be case-insensitive (i.e. treat "The" and "the" as the same word). The keys of the returned map should be in sorted order, while the sets in the values should prefer fast lookup speed.

You may assume that the parameter passed in is non-null, that each element of the parameter is non-null, and that word lists are non-empty.

**NOTE:** When writing your class, be sure to follow all guidelines in the **Code Quality Guide** and **Commenting Guide**.
Any additional helper methods created, but not specified in the spec, should be declared ***private***.

**WARNING:** Your implementation should only iterate over the provided documents list one time. We consider any alternative implementations inefficient.

## Testing [`Testing.java`]

▼ Expand

We have provided an incomplete `Testing.java` file that you should update lines according to the guiding comments within. You should only have to change 9 lines of code within this file such that it compiles and accurately tests your implementations.

> ⚠ **WARNING:** We've provided you a test that checks if your `Testing.java` file compiles and no tests fail. It does not check that the appropriate updates were made according to the comments within the file. It is your responsibility to make sure that you're updating the file correctly.

We recommend reading over this file to better understand how to write JUnit tests. As the quarter progresses, we will be providing you less testing guidance, so if you have any questions or confusions, it's best to ask them now!

## Search Engine

**NOTE:** We are only grading work related to the three steps outlined under Creative Extension on this slide.

***Download starter code:***

C0_SearchEngine.zip

Once you've completed all aspects of the assignment, we can put all these pieces together into an application! Most of the work has been done for you here, but you'll have to integrate your implementations from the workspace slide. This will involve 2 main steps:

1. Paste your implementation of `Book` within `Book.java`
2. Paste your implementation of `createIndex` @ `line 37`

**WARNING:** We've noticed that sometimes VSCode will automatically include an import for `javax.print.attribute.standard.Media;`. Please delete this if you notice it present within your `SearchClient.java` file!

**NOTE:** You don't need to worry about reuploading the books directory as it's already included in the scaffold!

Once you've done these 2 things, hit the run button to see your hard work! You'll have a working search engine over all the books within the `books` directory!

# Creative Extension

It turns out that creating an inverted index is only half of the battle when creating a search engine. If we want our version to be on-par with the likes of Google, we also need to create a ranking system that sorts more relevant documents before less relevant ones. To do so, we'll employ the `Comparable` interface we learned about in class!

The only tricky part is that for our ranking algorithm to be useful, our `compareTo` method should know what query the user entered (such that it can determine if `this` or `other` is more relevant) when `compareTo` can only take in one `other` parameter. To get around this, we'll store the current query within each `Book` object via a field / `setQuery` method we call before sorting.

**NOTE:** This workaround is not the best way of sorting using a value not related to specific instances (the query in this case). Something like a Comparator would likely be a better choice, but doing so is out of the scope of the course and disallowed, so we'll use our workaround instead.

This will involve 3 concrete steps:

1. Add `public void setQuery(String query);` to `Media.java` and write an implementation for `setQuery` in `Book.java` that stores the current query in a field.
2. Make `Book.java` also implement the `Comparable` interface and write a corresponding `compareTo` method
    1. This is where you can get creative! How do you want to rank books? **Our only requirement is that you use the query somehow**, but remember that you have other information (ratings, title, author[s]) at your disposal!
3. Uncomment `line 52` within `SearchClient.java` and change `line 99` to a `TreeSet` such that your ordering is used
    1. Note these line numbers might change after pasting in your `createIndex` implementation, but all are marked with a "TODO" comment you can search for within the file

**WARNING:** We require that your ranking algorithm uses the current query somehow. A ranking algorithm that only uses ratings would not be eligible for full points on this assignment

Once you've finished these steps, you have a working search engine that can compete with the likes of Google! Try using it on the provided files. Does the ordering of the results make sense?

# Reflection

The following questions will ask that you practice **metacognition** to reflect on the topics covered on this assignment and your experience completing it. For each question, focus on your plan and/or process for working through the assignment along with the CS concepts. Think about things like how you organized your working time, what sorts of things tended to go wrong, and how you dealt with those errors or mistakes.

Please answer all questions.

**Question 1**

Describe your process using the VSCode debugger in the Debugging problem (`Bomb.java`). What skills have you learned or practiced to help you when debugging code? How will you use or adapt this process for future assignments?

*No response*

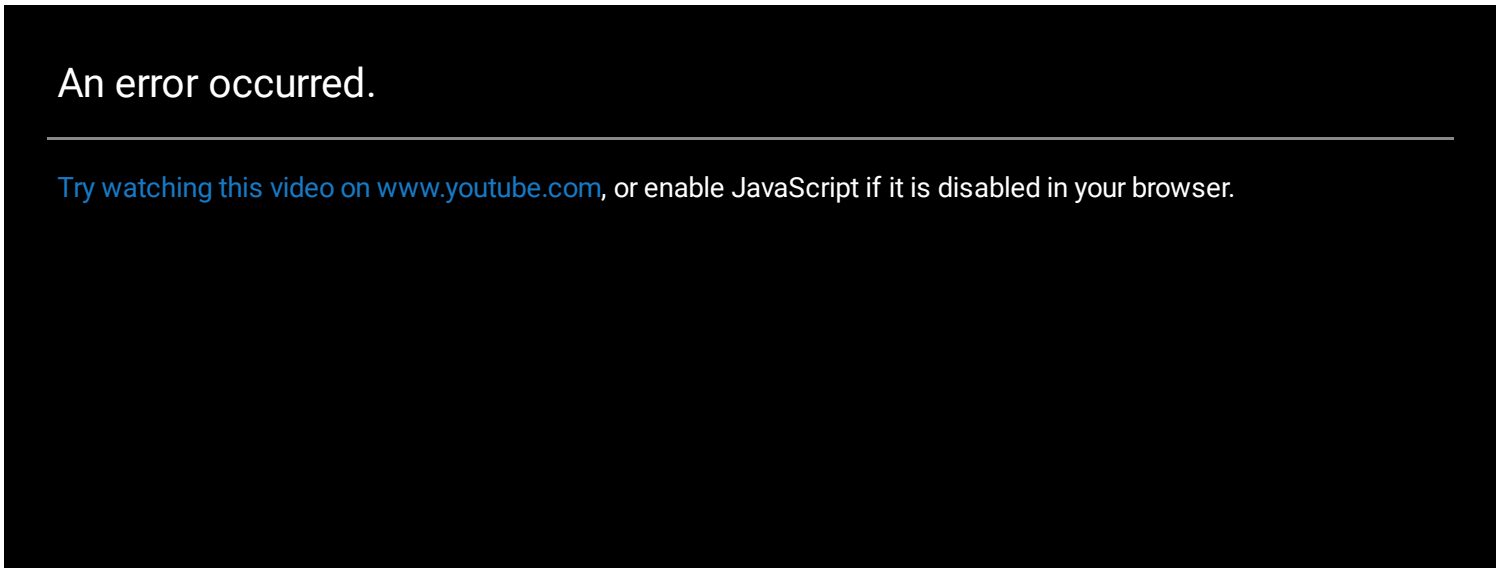
**Question 2**

Choose either the Inverted Index or Media problem: describe how you went about testing that the code you wrote for that problem was correct and met the requirements. What specific test cases did you consider? Why were those cases important?

*No response*

**Question 3**

The next 3 questions will require you reflect on your experience implementing a search engine and respond to the following video (9m 19s):



(https://youtu.be/_vBggxCNNno)

At Google, who's responsibility do you think it is to come up with moral rules and judgements surrounding search engine ranking results as described in the video? (Executives, managers, software engineers, other)

Do you think that they should have that power / responsibility? Why?

*No response*

**Question 4**

Do you think search engine providers (Google, Bing, etc.) have an obligation to remind their users that "unbiased, clean search results" can't truly exist as mentioned within the video? Why?

If you do think so, are they currently acting on that obligation? Why might that be the case?

*No response*

**Question 5**

How do you feel about the video's claim that software reflects the biases of the programmer? Do you agree / disagree? Why?

Have you ever encountered biased programs / applications in your day-to-day life?

*No response*

**Question 6**

What skills did you learn and/or practice with working on this assignment?

*No response*

**Question 7**

What did you struggle with most on this assignment?

*No response*

**Question 8**

What questions do you still have about the concepts and skills you used in this assignment?

*No response*

**Question 9**

About how long (in hours) did you spend on this assignment? (Feel free to estimate, but try to be close.)

*No response*

**Question 10**

Was any part of the specification or requirements unclear? If so, which part(s), how was it unclear, and how could it have been made more clear?

*No response*

**Question 11**

[OPTIONAL] Do you have any other feedback, questions, or comments about this assignment?

(Note that we may not be able to respond to questions here, so please post on the message board if you would like a response!)

*No response*

# □ Final Submission □

# □ Final Submission□

Fill out the box below and click "Submit" in the upper-right corner of the window to submit your work.

**Question**

I attest that the work I am about to submit is my own and was completed according to the course Academic Honesty and Collaboration policy. If I collaborated with any other students or utilized any outside resources, they are allowed and have been properly cited. If I have any concerns about this policy, I will reach out to the course staff to discuss *before* submitting.

(Type "yes" as your response.)

*No response*