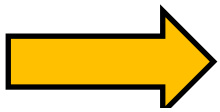
CSE 464: Software QA and Testing
Project Part #1

1. Create a github account and a **private repo** for the course project of CSE 464. You should name the repo using CSE 464 2025 and your ASU email (without@asu.edu).
   - How to create a github account: https://docs.github.com/en/get-started/signing-up-for-github/signing-up-for-a-new-github-account
   - How to create a repo in github: https://docs.github.com/en/get-started/quickstart/create-a-repo. Do remember to set it as a private repo!
   - **Share your repo with the TA/graders**: https://docs.github.com/en/account- and-profile/setting-up-and-managing-your-personal-account-on-github/managing-access-to-your- personal-repositories/inviting-collaborators-to-a-personal-repository
     - TA Liangyi Huang: liangh999
     - Grader Kavya Chandrika Vempalli: kavyachandrikavempalli
     - Grader Saideep Manjunath: Saideep-Manjuath
   - **Provide the link of your github repo at** https://docs.google.com/spreadsheets/d/1YIXeYnzskgnpDyh9BlB5qit1tzw1nVGJMmu2J1EbXPk/edit?usp=sharing
     - **Not doing this step will cause 5 points to be deducted.**
2. Install Java JDK 11 or above and Intellij community edition
   - Java JDK 11+: https://docs.oracle.com/en/java/javase/22/install/overview-jdk-installation.html
   - Intellij: https://www.jetbrains.com/idea/download
     - Download the community edition (free version)
3. Install Graphviz and use it to convert graphs in DOT format to graphics (e.g., jpg, png, and pdf)
   - Download Graphviz: https://graphviz.org/download/
   - Documentation for DOT language and format: https://graphviz.org/doc/info/lang.html
   - Use command line to convert a DOT graph to a png file
     - Command line doc: https://graphviz.org/doc/info/command.html
     - Example: dot -Tpng input.dot -o output.png

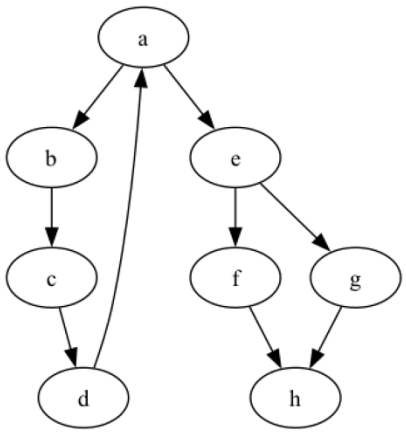
```
xxiao8@EN4217604-SCAI Course Project % dot -Tpng input.dot -o input.png
xxiao8@EN4217604-SCAI Course Project % ls -l input*
-rw-rw-r--@ 1 xxiao8  staff    119 Feb 18  2023 input.dot
-rw-r--r--  1 xxiao8  staff  25906 Sep 17 10:52 input.png
```

```
digraph {
    a -> b;
    b -> c;
    c -> d;
    d -> a;
    a -> e;
    e -> f;
    e -> g;
    f -> h;
    g -> h;
}
```

input.dot



dot -Tpng



input.png

4. Write a Java program that can parse graphs in DOT format, manipulate graphs (e.g., add nodes, add edges), and output graphs in DOT format and graphics (e.g., jpg and png)
    - Example Java libraries for parsing DOT graphs
        - JGraphT: https://jgrapht.org/
        - graphviz-java: https://github.com/nidi3/graphviz-java
        - You may find your own libraries
    - You need to implement 4 features and check in the code to your created repo in multiple commits. Each feature must be checked in using a separate commit, i.e., you need **at least 4 commits. Please use the suggested API names as follows.**
    - **Feature 1:** Parse a DOT graph file to create a graph (20 points)
        - Accept a DOT graph file and create a directed graph object (define your own graph class or use the graph class in your chosen libraries)
            - API: parseGraph(String filepath)
        - Output the number of nodes, the label of the nodes, the number of edges, the nodes and the edge direction of edges (e.g., a -> b)
            - API for printing a graph: toString()
            - API for output to file: outputGraph(String filepath)
    - **Feature 2:** Adding nodes from the imported graph (10 points)
        - Provide APIs to add a node
            - Add a node and check of duplicate labels: addNode(String label)
        - Provide APIs to add a list of nodes
            - Add a list of nodes: addNodes(String[] label)
    - **Feature 3:** Adding edges from the imported graph (10 points)
        - Provide APIs to add an edge
            - Add an edge and check of duplicate edges: addEdge(String srcLabel, String dstLabel)
    - **Feature 4:** Output the imported graph into a DOT file or graphics (20 points)
        - Provide API to output the imported graph into a DOT file
            - API: outputDOTGraph(String path)
        - Provide API to output the imported graph into a graphics
            - API: outputGraphics(String path, String format)
            - Supported format: at least for png (no need for jpg)
    - Export your Intellij project to a zip file (Link)

5. Write unit tests for each feature (20 points)
    - The tests should clearly provide test inputs (may be a string or an input file), the test method, and the expected outputs.
    - For comparing graph output in the text format, you can create another text file like "expected.txt" and read the file to compare with your code's output:
        - ```
          String expected =
          Files.readString(Paths.get("expected.txt"));
          ```
        - ```
          Assert.assertEquals(expected, output);
          ```

6. **Please add maven support for your project hosted in the github repo**: https://maven.apache.org/guides/getting-started/index.html. The maven config file (pom.xml) should include a name (same as your repo) for your built package, your java version, and all the dependencies to build and test your code. Once it is setup, the TA/Graders should be able to run "mvn package" for building and testing your code (20 points).
   - Below is an example pom.xml:

```xml
<groupId>asu</groupId>
<artifactId>CSE464-GraphManager</artifactId>
<version>1.0-SNAPSHOT</version>

<name>CSE 464 Course Project</name>

<properties>
    <maven.compiler.source>18</maven.compiler.source>
    <maven.compiler.target>18</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
    <dependency>
        <groupId>guru.nidi</groupId>
        <artifactId>graphviz-java</artifactId>
        <version>0.18.1</version>
    </dependency>
```

Write a readme (in PDF) that provides the instructions and example code to run your code. TA and graders will run your code using your provided example inputs, run your tests, and run your code using our own inputs. You should submit your code (including your used libraries) and the readme in a zip file. The submission must be accepted via Canvas.

The readme PDF must meet the following requirements, or up to 20 points will be deducted:

- The github link must be provided in the google sheet: ~~https://docs.google.com/spreadsheets/d/1YIXeYnzskgnpDyh9BlB5qit1tzw1nVGJMmu2J1EbXPk/edit?usp=sharing~~ https://docs.google.com/spreadsheets/d/1YIXeYnzskgnpDyh9BlB5qit1tzw1nVGJMmu2J1EbXPk
- The readme PDF must be checked into the repo, or attach in the comments of your canvas submission.
- Your code must be able to be compiled by typing "mvn package", which will go through validate, compile, test, and package phases (https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html).
- You should provide screenshots of the expected output for each feature in the readme PDF, so that the TA/graders can easily check your results after they run your code.
- You must provide links to the github commits for each feature. For example, you must provide a link for showing your continuous integration work properly in github, and you should provide links for your branches and successful merges.
- You don't have to check in your built binaries and other built artifacts.