**Version Control System**

This program will be using a version control system. What this means is, when you make a change, you have to commit it to a repository. You can also update your local version to the repository version. This allows concurrent development, go back to previous versions, and allow others to view and copy the code for their own use. Shorter commits are better. Also make sure to clearly define the changes, to help developers fix bugs later.

The code will be under a CC0 license, meaning anyone is allowed to use, modify, and distribute the code without acknowledging us, in personal and commercial applications.

**libGDX**

libGDX is a library for java. Libraries allow a program to use more functions than the JDK has built in. In order to use them, you must tell your IDE where the jar files are. If you are using IntelliJ Idea, the process is as follows:

1) Click "Project Structure" (ctrl+alt+shift+s)

2) Go to the "Libraries" tab

3) Click the green "+" button, and select "Java"

4) Select the "libgdx/libs" folder

5) Repeat step 3

6) Select the "libgdx/sources" folder

Also do this for the reflections library. (Reflections doesn't have and sources, just the library)

And that's it! Your IDE should now auto complete for the new functions.

To get started with libGDX, you create a class that implements ApplicationListener. You then add your main function, which defines aLWJGLApplicationConfiguration, and creates a LWJGLApplication, with an instantiation of the ApplicationListener you created, plus the configuration.

**The Main Menu**

Now lets go to the ApplicationListener. This will be what manages all of the different parts of the program. libGDX works with a hierarchy of screens, stages, tables, and actors. First we will create a screen, the "MenuScreen." In the create() function of the application listener, set the screen to a new MenuScreen. (setScreen(new MenuScreen());)

This MenuScreen will create a stage with a window inside. This window will have buttons to get to each of the different parts of the program. The buttons will be flowchart styled, with arrows going from the basic panels, to the advanced ones.

**The Panels**

Each panel will essentially a slide, showing either more panels, or text, or both. The panel class will be an abstract class implementing Screen and Comparable<Panel>. The panel will have a variable for its return screen (Screen), its title (String), its index(int), and its content (Table). Those will all get set by its implementations. It will also define a stage that it renders. The stage adds the title, content, and a button in the top left that creates a new instance of its return scene. The compare method will look at the indexes of the panels. This is for ordering the top-level panels. The most basic will be at index 0, the next panel 1, etc. Panels that are not top level will have an index of -1.

The Main Menu will use Reflections to detect all classes that implement the Panel class, and put them into an ArrayList of classes. It will then sort them, and make buttons for each of the top-level panels, with (square root of the total number of top-level panels, rounded down) buttons in each row. What all of this does is allow you to develop the panels concurrently without worrying about conflicts with other commits, and add panels without needing to change the main menu. If a panel is not a top-level panel, it also has a button to go to the main menu (defined in Panel class).

**Data Logging**

When the program runs, it will create a file called "log.txt". This file will be appended to whenever someone changes panels, and how long Scratch is open. Each event will add a new line, what happened, and a time stamp. Also, use libGDX's [file handling](https://github.com/libgdx/libgdx/wiki/File-handling) to write to the log.

**The Panels**

This list of panels is subject to change dependent on how long they take (meaning more panels will be added it time allows). Volunteers will be assigned certain panels. You can look at tempTopLevelPanel and tempPanel for an example of how to make a panel.

* What is Computer Science?
* Logic Gates
  + Redstone
  + Scratch Logic Gates program
* What is Scratch? (include QR code to scratch website)
  + Subpanels will show how to use it, broken down into each category of blocks, plus one explaining how the rest of Scratch works
* Scratch (launches Scratch)
* How to contribute to this project
  + Java
  + Version Control Systems
  + Github
* Bibliography (also include helpful links such as code.org)