

Assignment 7

In this assignment you will develop a simple React Native App based on functional components developed in Kotlin for Assignment 1 and Swift for Assignment 4 to gain a preliminary understanding of:

- JavaScript
- React and JSX
- Cross Platform development with React Native
- Automated testing using Jest and React Native Testing Library

This assignment is worth 10% of your final grade.

Late submissions will not be graded.

Configuration

In Android Studio, create an Android Virtual Device (AVD) with the following specs:

Name:	Pixel 5 API 33
Phone:	Pixel 5
System Image:	S 32 (you may have to download this)
RAM:	4096 MB
VM Heap:	256 MB
Internal Storage:	8192 MB

On Windows:

- Install Node.js LTS if you don't already have it: <https://nodejs.org/en/download/>
- Set the following environment variables:

```
ANDROID_HOME=%LOCALAPPDATA%\Android\Sdk
JAVA_HOME="C:\Program Files\Android\Android Studio\jre"
PATH=%PATH%;%ANDROID_HOME%\platform-tools;%JAVA_HOME%\bin
```

On macOS:

- Install Homebrew if you don't already have it: <https://brew.sh/>
- Install Node.js if you don't already have it, and Watchman

```
$ brew install node
$ brew install watchman
```

- Install Java if you don't already have it:

```
$ brew tap homebrew/cask-versions
$ brew install --cask zulu11
```

- Set the following environment variables:

```
ANDROID_SDK_ROOT=$HOME/Library/Android/sdk
PATH=$PATH:$ANDROID_SDK_ROOT/emulator:$ANDROID_SDK_ROOT/platform-tools
```

Setup

Download the starter code archive from Canvas and expand into an empty folder. I recommend creating a folder for the class and individual folders beneath that for each assignment.

- In Android Studio, start the Pixel 5 API 32 AVD
- In a terminal (command prompt on Windows) navigate to the installation folder and run the following command which will take a while to complete:

```
$ npm install
```

- Then run the following command to start the react-native development server:

```
$ npm run start
```

- Now open another terminal (command prompt on Windows) navigate to the installation folder again and enter the following command:

```
$ npm run android
```

The app should appear in the Android Simulator, and you can start editing the JSX.

- To run the tests and check code coverage:

```
$ npm test
```

- To run just some of the tests:

```
$ npm test <filter>      where <filter> is Logic, UI, Basic, or Advanced
```

- To run check code quality:

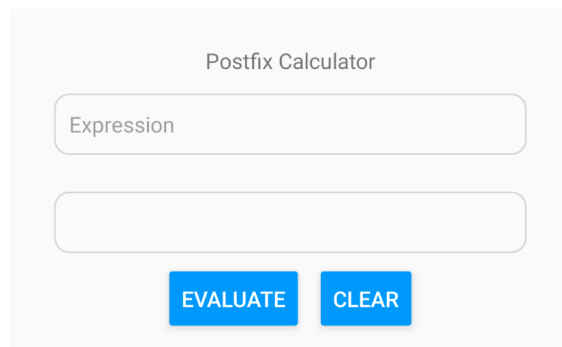
```
$ npm run lint
```

Requirements

Basic:

First, implement a JavaScript version of the Postfix Calculator from Assignments 1 and 4 such that it passes the tests in `__test__/Logic/BasicTests.js`

Next, implement a React Native version of the Postfix Calculator UI from Assignment 4 such that it passes the tests in `__test__/UI/BasicTests.js` and looks something like this:

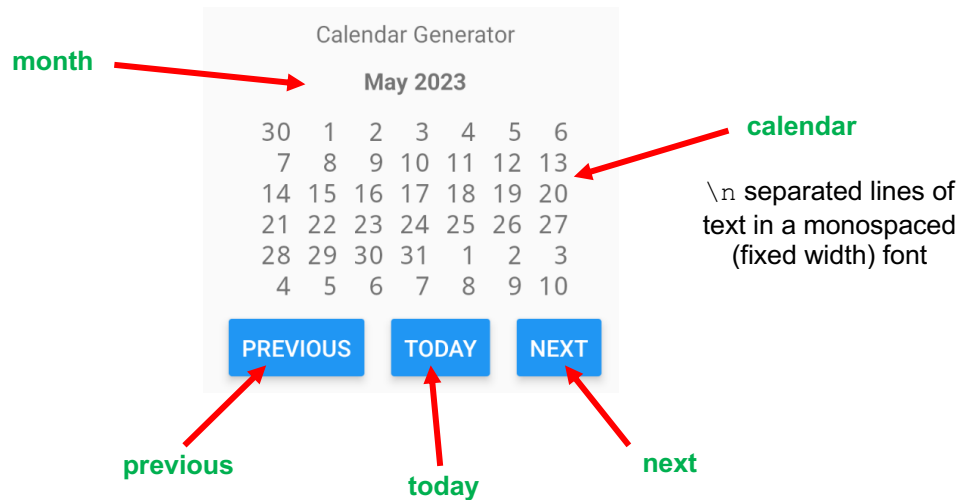


Advanced:

First, implement and write tests for JavaScript version of the Calendar Generator from Assignments 1 and 4.

Next, implement and write tests for a React Native version of the Calendar Generator UI from Assignment 4.

In React Native the JSX components you use are irrelevant, but case sensitive accessibility labels as shown in green are required.



Note you must write tests for the Calendar Generator in `__tests__/Logic/AdvancedTests.js`, and tests for the related UI in `__tests__/UI/AdvancedTests.js`. If you add additional test files your solution may fail to compile in the grading system.

For the new tests, certainly consult <https://jestjs.io/> and <https://callstack.github.io/react-native-testing-library/> but primarily study the supplied Basic tests.

Stretch:

Your App should exhibit no linter errors and 100% class, method, line, and branch coverage when the provided Basic tests and your Advanced tests (if any) are executed.

What steps should you take to tackle this?

Start with the Postfix Calculator, then move on to the Calendar Generator and its tests. The logic is identical to the classes you developed in Assignments 1 and 4; the only difference is the language being used.

On the other hand, the UI code for the two components is very different so will take longer to develop.

Note that completing the Basic requirement with 100% code coverage is worth 60% on this assignment. Take care when implementing Advanced functionality to pay close attention to your coverage.

How much code will you need to write?

A model solution that satisfies all requirements has approximately 300 lines of code, including tests for the Advanced requirement.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

- | | |
|-------------|---------|
| a. Basic | (40%) |
| b. Advanced | (40%) |
| c. Stretch | (20%) |

2. (-100%) **Did you give credit where credit is due?**

- a. Your submission is found to contain code segments copied from on-line resources or created by code generation tools and you failed to give clear and unambiguous credit to the original author(s) in your source code. You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy. (-100%).
- b. Your submission is determined to be a copy of a past or present student's submission. (-100%)
- c. Your submission is found to contain code segments copied from on-line resources that you did give a clear and unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:
 - < 25% copied code No deduction
 - 25% to 50% copied code (-50%)
 - > 50% copied code (-100%)

What to submit

On the console (PowerShell on Windows), navigate to the folder you extracted the starter code into and run the appropriate command to create the submission archive:

```
$ npm run zip
```

**** UPLOAD Assignment7.Submission.zip TO THE CANVAS ASSIGNMENT AND SUBMIT ****