ECE 30864 (364) Prelab Assignment 01

Intro to Software Engineering

Passing this lab will satisfy course objectives CO1

Instructions

- You must meet all *base requirements* in the syllabus to receive any credit.
- Work in your Prelab01 directory, and copy all files from ~ee364/DataFolder/Prelab01 into your working directory:

  ```
  cp -r ~ee364/DataFolder/Prelab01/* ./
  ```

- Remember to add and commit all **required** files to Git. **We will grade the version of the file that is in Github!**
- Do *not* add any file that is *not* required. **You may lose points if your repository contains more files than required!**
- Make sure you file compiles. **You will not receive any credit if your file does not compile.**
- Name and spell the file, and the functions, exactly as instructed. Your scripts will be graded by an automated process. **You may lose some points, per our discretion, for any function that does not match the expected name.**
- Make sure your output from all functions match the given examples (If applicable).
- Unless otherwise specified, you cannot use any external library, but you can use any module in the **Python Standard Library** to solve this lab, i.e. anything under:

  ```
  https://docs.python.org/3.8/library/index.html
  ```

- Make sure you are using Python 3.8 or above for your Prelab.

Intro to Software Engineering

Required Files `simpleTasks.py`, `index.html`, `index.css`

Create a Python file named `simpleTasks.py`, and do all of your python work in that file. Refer to the **Python File Structure** section below for a reminder on the requirements. For HTML/CSS, use the provided index.html and index.css files

Implementation Requirements

1. **[20 pts]** Write a function called `writePyramids(filePath, baseSize, count, char)` that saves one or more pyramid-shaped sequence of characters in file, separated by a single space at the base. The `filePath` is the full path of the file to save the pyramids to, `baseSize` is an odd integer representing the size of the pyramid's base, `count` is the number of horizontally-concatenated pyramids to create, and `char` is the character used to build the pyramids with. For example, the files `pyramid13.txt` and `pyramid15.txt` have been obtained using the commands:

   ```
   >>> writePyramids('pyramid13.txt', 13, 6, 'X')
   >>> writePyramids('pyramid15.txt', 15, 5, '*')
   ```

   Note: The files you generate must be an *exact* match to the ones provided, if you use the same arguments.

   Hints:

   - You may use a "Diff" tool to compare your result with the given samples. Visual eyeballing comparison is *not* sufficient, and can result in zero credit for missing one or more spaces. Note that VSCode extension "Diff" (By Fabio Spampinato) can be used for file comparison.
   - You can solve this question anyway you want, but a *suggested* procedure is as follows:
     - Generate line n for a single pyramid.
     - Generate a single pyramid, then use it to generate the rest.
     - Combine the pyramids in a list.
     - Write the list to the target file.

2. **[20 pts]** Consider the string `"AAASSSSSSAPPPSSPPBBCCCSSS"` containing a sequence of uppercase letters. Write a function called `getStreaks(sequence, letters)` that takes in a string similar to the example shown, and returns a list of the streaks, in the order of their appearance in the sequence, of the letters provided. For example:

   ```
   >>> sequence = "AAASSSSSSAPPPSSPPBBCCCSSS"
   >>> getStreaks(sequence, "SAQT")
   ['AAA', 'SSSSSS', 'A', 'SS', 'SSS']
   >>> getStreaks(sequence, "PAZ")
   ['AAA', 'A', 'PPP', 'PP']
   ```

   Notes:

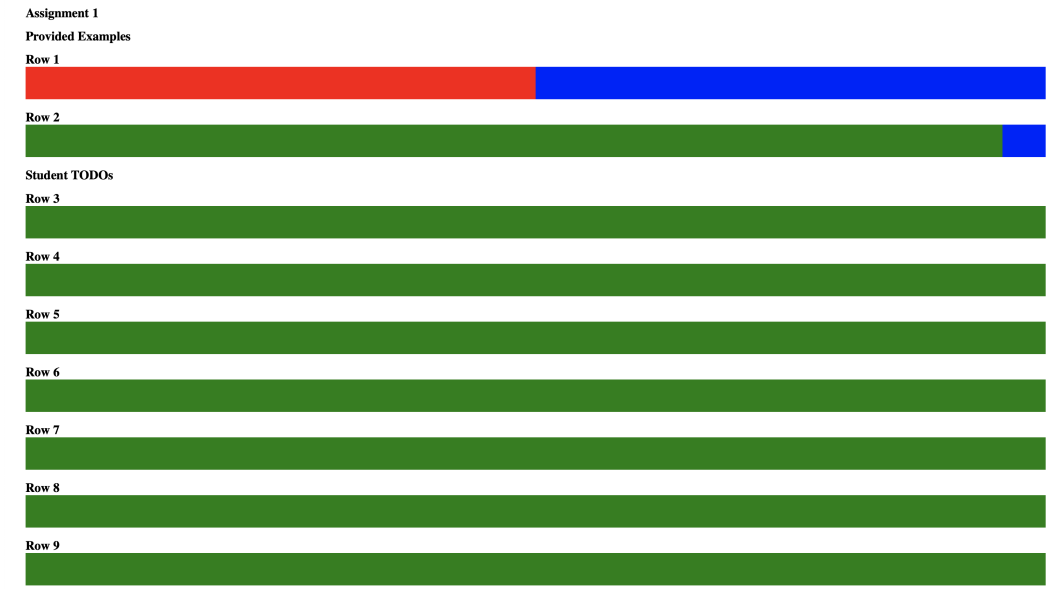
   - Return an empty list if no matches were found of any of the letters passed.
   - As shown in the example, the order of letters in the input argument `letters` is arbitrary, and not all letters may be present.

3. **[40 pts]** In this section you will tinker with basic HTML and CSS to style the given webpage. Use flexboxes to align HTML elements. Flexboxes are widely used in modern web development (BootstrapV4 is built with flexboxes).

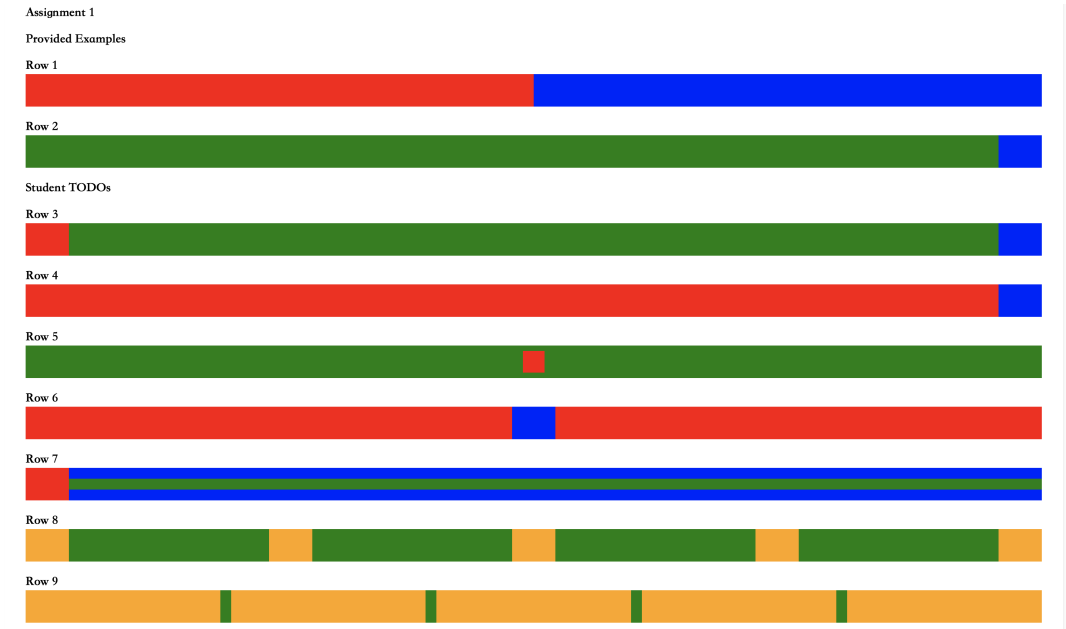
   References:

   - https://cs.brown.edu/courses/csci1320/assignments/assignment1/assignment1.html
   - Flexbox usage: https://css-tricks.com/snippets/css/a-guide-to-flexbox

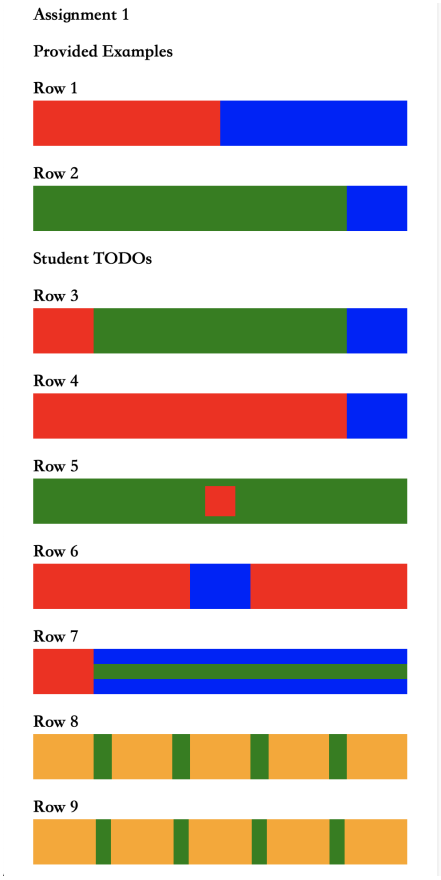
   When you open the provided HTML file part1/index.html, it should look like this:



   The styling of the first two rectangles is already done for your reference. Apply styling (with CSS) and add div elements (HTML) inside the 7 green rectangular blocks to create a webpage as below:



These rectangular blocks should be responsive (should automatically resize when browser window size changes). Eg:



Try to style the boxes as close to the solution image as possible not worrying about exact dimensions or rgb values. The color of the boxes we used are background-color: red, blue, and orange

Final Check!

You are given a file that checks for the exact spelling of the required functions. Make sure you run this file before your final commit. Remember that **you will lose some points for any function that does not match the expected name.**

Python File Structure

The following is the expected file structure that you need to conform to:

```
#########################################################
#   Author:     <Your Full Name>
#   email:      <Your Email>
#   ID:         <Your course ID, e.g. ee364j20>
#   Date:       <Start Date>
#########################################################

import os     # List of module import statements
import sys    # Each one on a line


#########################################################
# No Module-Level Variables or Statements!
# ONLY FUNCTIONS BEYOND THIS POINT!
#########################################################

def functionName1(a: float, b: float) -> float:
    return 0.


def functionName2(c: str, d: str) -> int:
    return 1


# This block is optional and can be used for testing.
# We will NOT look into its content.
#########################################################
if __name__ == "__main__":
    # Write anything here to test your code.
    z = functionName1(1, 2)
    print(z)
```