CS106 W20 - Assignment 02

Due: Friday, January 17, 11:59 PM

Revision Note: Assignment 02 was formerly the Trading Canadians hockey-card exchange. Trading Canadians is now Assignment 03. Judgy Typewriter was formerly part of Lab 02. Judgy Typewriter is now Assignment 02.

This assignment is an application of containers, as discussed in the second week of CS106.

In this lighthearted mashup of Scrabble scoring and the function of an electronic typewriter, you will practice using arrays containing single letters, longer strings, and records. You will do simple conversions among these formats.

Code a game that lays out a Scrabble tile when a letter is typed. When enter is pressed, take the letters that are laid out and turn them into a printed word. Show the points earned per word and in total. The video at https://youtu.be/J7BYFf01kiE shows the functionality.

Specifically, follow these steps.

1. Inspect the starter code. Note that it provides the point values for each letter, in array-of-records format. Note that it offers a `drawTile` function that draws a tile. You should not change this function. Note that the starting program shows a few tiles of total nonsense. In the coming steps, you will change this behavior and guard against such nonsense getting into your gameplay.
2. Modify the way that those few fixed tiles get drawn.
    a. Add code to `setup` that fills a few forged tile records into `tilesInRack`. It should have this shape:
    ```
    let fakeTile1 = {};
    // ... now fill in its properties
    let fakeTile2 = {};
    // ... now fill in its properties
    // ... now put them in tilesInRack
    ```
    As a check, you should be able to add a last line of `setup` that goes:
    ```
    print(tilesInRack.length + ' ' +
      tilesInRack[0].letter + ' ' +
      tilesInRack[0].points);
    ```
    and get the message “2 hi 999” to appear in the log.
    b. Modify the code of `draw` so that, instead of drawing fixed fake tiles, it loops through your `tilesInRack` and prints those. This drawn result could accompany the example from step (a).

   c. At this point, note that you have moved the fake data out of draw and into setup. Your draw implementation is now the real deal.

3. Modify the fixed-tile production in `setup` so that a few real tiles from `tileDefinitions` are used, instead of forgeries. Index into `tileDefinitions` using hard-coded numbers for this part. With your logging from (a) still in place, you should be able to get the message "3 a 1" to appear in the log along with this drawn result.

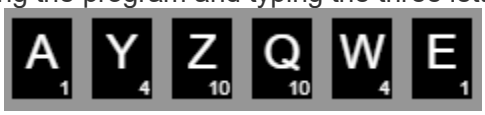
4. Modify your `draw` code so that you see capital letters in the tiles. Do not modify `tileDefinitions` or `drawTile`. It should look like this.

5. Respond to keyboard typing by adding more tiles to the rack.
   a. First, have any typing at all add an "A." So, launching the program and typing the three letters "qwe" should show:

   b. Next, use the character code of the typed character to give the lookup index into the `tileDefinitions` array. Use these values

```
let lowerCharCodeA = 'a'.charCodeAt(0);
let lowerCharCodeZ = 'z'.charCodeAt(0);
```

   along with similar analysis of the typed-in key, and the `map` function. So now, launching the program and typing the three letters "qwe" should show:



   c. As needed, modify your key-typed lookup logic so that it supports typing upper- or lower-case letters, and so that it ignores non-letters. So now, launching the program and typing the five-character sequence "2QooL" should show:

   d. Get rid of the AYZ from `setup`, along with its log message. So now, launching the program and typing the five-character sequence "2QooL" should show:

6. Respond to pressing Enter by logging the letters.
   a. Define a function acceptRack, which your key handler must call on newline or carriage-return characters. Verify that it's getting called in the right cases by using a print statement, and that it's not interfering with other typing.

    b. In acceptRack, write a loop according to the item transformation idiom that converts from `tilesInRack` (which contains tile records), to a local variable, `letters` (which contains an equal number of single-character strings). After the loop, have:
       ```
       print("Accepting " + letters);
       ```
       So now, launching the program and typing the six-step sequence "2QooL*[enter]*" should mean you see "Accepting q,o,o,l" in the log.
    c. Build a string of these letters by doing an appropriate `letters.join` call and logging that.
7. Respond to pressing Enter by saving the word that the tiles spell.
    a. In `acceptRack`, modify your loop to follow the distillation idiom as well. Calculate the total points of all the tiles in the rack.
    b. Create a structure with elements
       `.word`, with the string computed in step 6
       `.points`, with the score computed in step 7a
    c. Add this structure to `wordsPrinted`.
    d. Reset `tilesInRack` to an empty array.
    e. Increment `grandTotalPoints` by the new word's total.
    f. Log the length of `wordsPrinted`.
    g. Verify that typing multiple word-enter rounds shows the on-screen grand total increasing by the right amount and the logged `wordsPrinted` length increasing by one each time.
8. Modify `draw` to implement the typewriting.
    a. Add a loop that draws each `wordsPrinted[i].word` with a horizontal layout in size-20 Times New Roman. Leave the width of one space (`' '`) between them. Because this is not a fixed-width font, you need to use `textWidth` to calculate the x-coordinate jumps.
    b. Aligned with the rightmost edge of each word, draw the associated `wordsPrinted[i].points` just below it. Use size-10 Arial font.
    c. Test that game-play proceeds as in the video.

General Correctness

- One mark will be deducted for files or directories named incorrectly (the zip file, etc.)
- One or more marks will be deducted if the program crashes (depending on the severity).

Assignments that do not run may receive a grade of 0. Even if you don't complete the entire assignment, don't leave it in a broken state. Make sure it runs so we can find ways to give you part marks.

General Style and Efficiency

- One or more marks will be deducted if your code does not adhere to the style and efficiency requirements given in Assignment 1.
- You cannot get part marks if the TA cannot understand what you are trying to do.

Restrictions

In general, you may not use any functions, libraries, or statements not covered in lecture or labs unless not specifically exempted below or in a post by a TA or instructor on this assignment discussion board. For example:

- NO translate(), rotate(), or scale() functions
- No classes
- You MAY use bezier, arc, and other standard drawing functions

If in doubt, make a post to ask about using a specific statement or function.
Functionality marks will be deducted for using forbidden functions/statements.

Submitting

Zip your assignment sketch folder (A2_all) into one zip file CS106_A02. Submit it to the correct assignment dropbox. Consult "How to Submit" on Learn for more information on how to create a ZIP.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will receive a mark of 0.