**Assignment #2: Scrabble Scorer**

To see how far your skills have come, we are going to give you a mission to update our *super* important Scrabble Scorer program. Did you think you were going to work on Mars rover code already?

The current Scrabble Scorer, is a program that will take in a word and return the point value. The program needs to be rewritten for two reasons. First the data structure used to store the letter point values is inefficient. Second the program should have three scoring algorithms and allow the user to pick which algorithm to use. These new features will make the program more efficient and user friendly.

If you are NOT enrolled in a repl.it classroom for this course, fork the starter code at this [repl.it](https://repl.it/@launchcode/scrabble-scorer).

**Note**

The requirements below are what your END assignment will look like. This assignment is broken down so you can complete small pieces as you go. You need to move sequentially starting at “Transform”. Please read the WHOLE assignment page before starting.

**Requirements**

1. Create an **oldScoreKey** object that contains the previous scoring system.
2. Create a **transform** function which will return a **newScoreKey** object.
3. Create an **initialPrompt** function that asks the user which scoring algorithm to use.
4. Create a **scoringAlgorithms** array to hold three scoring objects, which include different scoring functions.
5. Create a **runProgram** function to serve as the starting point for your program.

**Detailed Requirements**

**Transform**

Currently the Scrabble Grader software uses the data structure below for the traditional Scrabble scoring algorithm. Take a few moments to review how **oldScoreKey** relates a letter to a point value.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **const** oldScoreKey = {  **1**: ['A', 'E', 'I', 'O', 'U', 'L', 'N', 'R', 'S', 'T'],  **2**: ['D', 'G'],  **3**: ['B', 'C', 'M', 'P'],  **4**: ['F', 'H', 'V', 'W', 'Y'],  **5**: ['K'],  **8**: ['J', 'X'],  **10**: ['Q', 'Z']  }; |

The object keys of **oldScoreKey** are the Scrabble points, and the values are arrays of letters. All letters in the array have the Scrabble point value equal to the key. For example, **'A'** is worth 1, and **'J'** is worth 8.

With the old format, to find the point value for a letter, the program must iterate over each key in **oldScoreKey** and then check if the letter is inside the array paired with that key. This search within a search is inefficient.

It would be better to create a **newScoreKey** object that has 26 keys, one for each letter. The value of each key will be the Scrabble point value.

Example of new key storage

* **a** is worth **1**
* **j** is worth **8**
* **k** is worth **10**

In **newScoreKey**, the letters themselves are keys, so a single search will identify a point value.

**Example**

Example of **newScoreKey** object usage.

console.log("Scrabble scoring values for");

console.log("letter a: ", newScoreKey.a);

console.log("letter j: ", newScoreKey.j);

console.log("letter z: ", newScoreKey["z"]);

**Console Output**

Scrabble scoring values for

letter a: 1

letter j: 8

letter z: 10

Write a **transform** function that takes the **oldScoreKey** object as a parameter. **transform(oldScoreKey)** will return an object with the letters as keys. The keys in the **newScoreKey** object should be *lowercase* letters.

**Hints**

1. Recall that **for...in** loops iterate over the keys within an object.
2. To access the letter arrays within **oldScoreKey**, use bracket notation (**oldScoreKey['key']**).
3. To access a particular element within a letter array, add a second set of brackets (**oldScoreKey['key'][index]**), or assign the array to a variable.

**Example**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | console.log("Letters with score '4':", oldScoreKey['4']);  console.log("3rd letter within the key '4' array:", oldScoreKey['4'][**2**]);  **let** letters = oldScoreKey['8'];  console.log("Letters with score '8':", letters);  console.log("2nd letter within the key '8' array:", letters[**1**]); |

**Console Output**

Letters with score '4': [ 'F', 'H', 'V', 'W', 'Y' ]

3rd letter within the key '4' array: V

Letters with score '8': [ 'J', 'X' ]

2nd letter within the key '8' array: X

**User Prompts**

The current Scrabble Scorer only uses one scoring algorithm. For the new version we want to let the user pick between three algorithms. Define an **initialPrompt** function that will introduce the program and then ask the user which scoring algorithm they want to use.

**Scoring Algorithms**

Create a **scoringAlgorithms** array that contains three scorer objects. Each object should contain three keys: **name**, **description**, and **scoreFunction**.

The **scoreFunction** for each object should be a function that takes in one parameter named **word** and returns a point value based on the logic listed below. The **scoreFunction** functions can named or anonymous.

| **Name** | **Description** | **Score Function** |
| --- | --- | --- |
| Scrabble | The traditional scoring algorithm. | A function with a **word** parameter that returns a score. Uses the **newScoreKey**object to determine that score. |
| Simple Score | Each letter is worth 1 point. | A function with a **word** parameter that returns a score. |
| Bonus Vowels | Vowels are 3 pts, consonants are 1pt. | A function with **word** parameter that returns a score. |

**Example**

*// Scrabble scoring*

console.log("algorithm name: ", scoringAlgorithms[**0**].name);

console.log("scoreFunction result: ", scoringAlgorithms[**0**].scoreFunction("JavaScript"));

*// Simple scoring*

console.log("algorithm name: ", scoringAlgorithms[**1**].name);

console.log("scoreFunction result: ", scoringAlgorithms[**1**].scoreFunction("JavaScript"));

*// Bonus Vowel scoring*

console.log("algorithm name: ", scoringAlgorithms[**2**].name);

console.log("scoreFunction result: ", scoringAlgorithms[**2**].scoreFunction("JavaScript"));

Console Output

algorithm name: Scrabble

scoreFunction result: 24

algorithm name: Simple Score

scoreFunction result: 10

algorithm name: Bonus Vowels

scoreFunction result: 16

**Note**

All three scoring algorithms are case *insensitive*, meaning that they should ignore case.

**Tie it All Together**

Define a **runProgram** function that will:

1. Accept the **scoringAlgorithms** array as an argument.
2. Use **initialPrompt** to pick the algorithm.
3. Prompt the user for a word to score.
4. Use the selected algorithm to determine the score for the word:
   1. If the user entered **0** or an invalid option, use the Scrabble **scoreFunction**.
   2. If the user entered **1**, use the Simple Score **scoreFunction**.
   3. If the user entered **2**, use the Bonus Vowels **scoreFunction**.
5. Display the score for the word.
6. Repeat steps 3 to 5 until the program is stopped.

**Test Words**

Here are some words you can use to test your code:

1. **JavaScript** = 24 points using Scrabble, 10 using Simple Score, and 16 using Bonus Vowels.
2. **Scrabble** = 14 points using Scrabble, 8 using Simple Score, and 12 using Bonus Vowels.
3. **Zox** = 19 points using Scrabble, 3 using Simple Score, and 5 using Bonus Vowels.

**Example Output**

Welcome to the Scrabble score calculator. Enter 'Stop' to quit.

Which scoring algorithm would you like to use?

0 - Scrabble: The traditional scoring algorithm.

1 - Simple Score: Each letter is worth 1 point.

2 - Bonus Vowels: Vowels are 3 pts, consonants are 1pt

Enter 0,1,2: 0

Using algorithm: Scrabble

Enter a word to be scored: LaunchCode

Score for 'LaunchCode': 18

Enter a word to be scored: Rocket

Score for 'Rocket': 12

Enter a word to be scored:

**Bonus Mission**

Score words spelled with blank tiles by adding **' '** to the **newScoreKey** object. The point value for a blank tile is **0** points.

**Submitting Your Work**

1. From the address bar at the top of the browser window, copy the URL of the repl.it that contains your solution.

**Example**

repl.it classroom URL: **https://repl.it/student/submissions/9999999**

1. Go to the Graded Assignment #2 page in Canvas and click *Submit Assignment*.
2. Paste the URL into the Website URL input.
3. Click *Submit Assignment* again.
4. Notify your TA that your assignment is ready to be graded.