How the Radiant Quiz Works:

A Release from the Truthwatchers:

When calcuating what order you are, there needs to be a reference for what the "Optimal" Radiant is, I hesitate to call this the "Ideal" Radiant as there can never be a true ideal, so long as you follow the ideals presented by one's order you *are* the ideal. But there *IS* an "Optimal" radiant for the purposes of the quiz. And so I took a peek behind the scenes and extracted what that the "Optimal" answers to each question are for each order. As well as how scores are calculated. I now present these to the Cosmere as a curiosity.

The Quiz

The quiz is a series of Sliders that range from 0 to 100 each starting at 50.
The steps are outlined in code as:

```
var results = calculateResults();
var chosenOrder = 0;
var lowScore = 999999;
for (var k = 0; k < results.length; k++) {
  if (results[k] < lowScore) {
    chosenOrder = k;
    lowScore = results[k];
  }
}
var results = sortWithIndices(results);
var normalizedResults = normalizeResults(results);
```

The first thing done is to Calculate the results of the quiz:

Calculate Results

```
function calculateResults() {
  var results = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
  var sliders = document.getElementsByClassName("slider");
  var promptTrait;
  for (var i = 0; i < quizPrompts.length; i++) {
    var promptSliderValue = sliders[i].value;
    promptTrait = quizPrompts[i].traitRight;
    var tempResults = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
    for (var j = 0; j < orders.length; j++) {
      tempResults[j] = Math.abs(traitData[promptTrait][j] - promptSliderValue) ** 2;
      results[j] += tempResults[j];
    }
    console.log("Trait " + promptTrait + " resulted in " + tempResults);
  }
  results[9] *= 1.3;
  console.log("Order totals: " + results);
  return results;
}
```

The results are stored as an array of 10 numbers which start at Zero. This is effectively your `affinity/results` for each radiant order. We'll call this the `affinityArray/results`.

For Each Question the following happens:
We get the `promptSliderValue` from the slider, this being a number between 0-100,

Next we grab the `promtTrait` for the question, this points to a row in the `traitData` table that is an ordered array of how each Order will answer the question. This is the "Optimal" answer for that question for that Order's members.

It should be noted that in the code there is a `traitLeft` and a `traitRight` but only `traitRight` is used. When I was mulling over why there was left and right while exercising I suspect that the first version of this quiz had a lower-bound and and upper-bound for the answer for each Radiant order but that was phased out and now it's just the "Optimal" answer.

So we look at the `traitData` table, grab the row associated with the question's `traitRight` then for each order we subtract the `promptSliderValue` by the `OptimialAnswer` and take the absolute value of result and raise it to the power of 2.

$$|(YourAnswer - RadiantOptimal)|^2$$

This answer is saved to a temporary affinity list and we move on to adding the next order's values. But is then immediately added into the actual results list, I suspect that this is another holdover from when they had a version of this quiz that used both the lower and upper bounds, calculated one, saved it into the tempResults array, calculated the other, then added the merge of the two into the final results. But for now it does nothing.

The above process is repeated for each order, for each question, summing their `affinity` points together.

At this point there is a singular Order who's values are manipulated. The Bondsmiths get a multiplier of 1.3 to their score. Perhaps the base percentage chance of being a bondsmith is inherently higher? Or perhaps the nature of Bondsmithery is such that most other Orders will overshadow them otherwise. Or maybe they got the "Optimal" answers wrong and boosting the Bondsmith score fixed the issue... We would need to talk to the creator of this Quiz to know for sure why this is as it is.

The next step is to find the order with the lowest number. Since the goal is to find the order that you most closely align with, you will want your deviation from your answers to their answers to be as low as possible. This is why we took the absolute value of the difference earlier, because we don't really care if the difference is positive or negative, mearly the distance from that order matters.

Sorting the Orders

```
function sortWithIndices(toSort) {
  for (var i = 0; i < toSort.length; i++) {
    toSort[i] = [toSort[i], i];
  }
  toSort.sort(function(left, right) {
    return left[0] < right[0] ? -1 : 1;
  });
  toSort.sortIndices = [];
  for (var j = 0; j < toSort.length; j++) {
    toSort.sortIndices.push(toSort[j][1]);
    toSort[j] = toSort[j][0];
  }
  return toSort;
}
```

The sorting algorythm does two things, because generally The orders themselves have fixed location in the array. (eg. index 0 for Windrunners, 9 for Bondsmiths) We need to keep that information during the sort, since the mere act of sorting is inherently destructive to that fixed order. So we save the Order into a 2D array, with the score of that order in the first position and the original index value of that score in the second position (the associated order). The 2D array is then sorted via low to high and the indicies are stored in a second sorted array `sortIndicies`

Normalization

```
function normalizeResults(results) {
  var normalizedResults = [];
  for (var i = 0; i < results.length; i++) {
    normalizedResults[i] = Math.floor((120000 - results[i]) / 1200);
  }
  return normalizedResults;
}
```

From here we get to the final step before displaying the results. Normalization. There are two major issues with our results at this point.

1. The numbers are the sum of all our scores raised to the power squared.
2. The numbers are inverted from our score, we only have the "top" portion of our score (like having 30% representing a score of 70%, they are inverses of eachother)

Before we talk about what happens here we need to talk a bit about scores. Let's say that the optimal answer is 100 and you answered 0. That would mean that your result for that question would be `10,000`. Since:

$$|(0-100)|^2$$

I'll admit this is about the extent of my statistics knowlege leads me. There are two "magic" numbers here. `120,000` and `1,200` I don't know why they were chosen but I suspect that if you have more or less questions you would need to change the numbers. But I can say with fair accuracy that by them being off by a factor of `100`, allows for a score of 0-100 once the scores are inverted and reduced. In the end that's all this function does. Inverts and reduces the number to the correct 0-100 Scale for presentation in the final results.