Quordle Keyboard

Instructions

Similar to Wordle, Quordle requires the user to solve for four five-letter words (answers) in nine attempts or fewer.

After each attempt, letters on the provided keyboard will be greyed out if they are no longer required. This may be the case if the answers containing that letter are solved, or it was not used in any of the answers. Greyed out letters can still be used in future attempts.

There are two parts to the challenge.

1. We would like you to derive a numeric string based on when are the letters greyed out.
2. We would also like to perform a simple checksum-like computation.

Assumptions for the purpose of this challenge:

- Each word is not actually an English word, it is really just a random sequence of five letters (`A` to `Z`, single case, may repeat).
- The four answers are distinct.
- There are at least four attempts (one per answer), and no more than nine attempts; if it helps all answers will be found.

Endpoint

Create an endpoint `/quordleKeyboard` that accepts a JSON payload over `POST` described below.

Input

Example of said JSON payload:

```
{
  "answers": ["ABCDE", "FGHIJ", "KLMNO", "PQRST"],
  "attempts": ["XYZXY", "ABCDE", "FGHIJ", "AAAAA", "PQRST", "KLMNO"],
  "numbers": [125, 441, 968, 137, 417, 554, 978, 666, 145, 137, 343, 26, 898, 54, 22
}
```

For part 1, you can ignore the `"numbers"` attribute; that is only applicable in part two.

Output

The output (i.e. your answer) to return should be in the shape of:

```
{
  "part1": "5555544444111111222222666",
  "part2": "HTKRGUVW"
}
```

Our evaluation is flexible enough to determine the scoring based on given attributes, more will be explained in the scoring section below.

Part One

We would like to see when are the letters greyed out after every attempt.

After the last attempt (which will also be an answer), we will map each *greyed out* letter (in sequence) to the number of attempts it has been greyed out for.

In other words, the resulting string of numbers will not contain a `0`.

Example

- Answers: `[VVIDH, MZLPS, BPCYN, XYGGM]`
- Attempts: `[JKGJB, ZGRUJ, XYGGM, BPCYN, MHXGE, DZENT, ZXWQW, VVIDH, MZLPS]`

```
         ABCDEFGHIJKLMNOPQRSTUVWXYZ
JKGJB    ABCDEFGHI  LMNOPQRSTUVWXYZ    J, K not in remaining 4 answers
ZGRUJ    ABCDEFGHI  LMNOPQ ST VWXYZ    R, U not in remaining 4 answers
XYGGM    ABCDEF HI  LMNOPQ ST VW YZ    G, X not in remaining 3 answers
BPCYN    A  DEF HI  LM OPQ ST VW  Z    B, C, N, Y not in remaining 2 answers
MHXGE    A  D F HI  LM OPQ ST VW  Z    E not in remaining 2 answers
DZENT    A  D F HI  LM OPQ S  VW  Z    T not in remaining 2 answers
ZXWQW    A  D F HI  LM OP  S  V   Z    Q, W not in remaining 2 answers
VVIDH    A    F     LM OP  S      Z    D, H, I, V not in remaining answer
MZLPS    A    F        O               L, M, P, S, Z answered, left with only A, F,
```

We can see `A` has not been greyed out, `B` has been greyed out for `6` times, and likewise for `C`.

If we were to replace the greyed out letters with the numeric value (empty columns) and the remaining letters with ` `, we can see:

```
         ABCDEFGHIJKLMNOPQRSTUVWXYZ
JKGJB             99
ZGRUJ             99      8  8
XYGGM          7  99      8  8  7
BPCYN     66   7  99  6   8  8  76
MHXGE     66 5 7  99  6   8  8  76
DZENT     66 5 7  99  6   8 48  76
ZXWQW     66 5 7  99  6  38 48 376
VVIDH     6625 72299  6  38 482376
MZLPS     6625 72299116 13814823761
```

Therefore, the answer to the first part of the challenge is `"66257229911613814823761"`. Note that it should be treated as a string of numbers, not a numeric-typed value.

Part Two

This is a continuation of part 1's answer and the `"numbers"` attribute on the input.

We will now need to derive a letter sequence based on the following steps:

1. Partition the 25-element number list into continuous lists of 5 elements each.
2. For each partition, check if the number can be found in part 1's answer, mapping this `true|false` value to `1|0`.
3. Convert this five-digit binary representation into a number, representing a letter's position in the alphabet, i.e. `1` to `A`, `26` to `Z`.
4. Finally, join these five letters with *leftover* letters in order, if there are any left.

Example

Continuing from part 1's example:

- Answers: `[VVIDH, MZLPS, BPCYN, XYGGM]`
- Attempts: `[JKGJB, ZGRUJ, XYGGM, BPCYN, MHXGE, DZENT, ZXWQW, VVIDH, MZLPS]`
- Numbers: `[761, 720, 13, 750, 936, 237, 482, 609, 585, 706, 240, 23, 76, 61, 700, 711, 823, 406, 376, 455, 818, 482, 338, 572, 257]`

Part 1's answer for this is `"66257229911613814823761"`, with leftover letters `"AFO"`.

Following the partitioning and the conversion, we get the following letters:

```
[761, 720,  13, 750, 936] -> [1, 0, 1, 0, 0] -> 20 = T    761, 13 are found in part1
[237, 482, 609, 585, 706] -> [1, 1, 0, 0, 0] -> 24 = X    237, 482 are found in part
[240,  23,  76,  61, 700] -> [0, 1, 1, 1, 0] -> 14 = N    23, 76, 61 are found in pa
[711, 823, 406, 376, 455] -> [0, 1, 0, 1, 0] -> 10 = J    823, 376 are found in part
[818, 482, 338, 572, 257] -> [0, 1, 0, 1, 1] -> 11 = K    482, 572, 257 are found in
```

Therefore, the answer to the second part of the challenge is `"TXNJKAFO"`.

Scoring

Refresher on the output:

```
{
  "part1": "5555544444111112222266",
  "part2": "HTKRGUVW"
}
```

If we encounter any error parsing the response, the score is `0`.

Part 1

If `"part1"` is absent, or it does not match the string of digits `1-9` with the expected length, the score is `0`.

Else, starting with a score of `20`, subtract the absolute difference per digit to get part 1's score, with a minimum of `0`.

Examples

```
"111345135153444"
"111345135153444" -> 20

"111345135153444"
"911345135153444" -> 20 - 8 -> 12

"111345135153444"
"977145135153444" -> 20 - 8 - 6 - 6 - 2 -> 0 (minimum)
```

Part 2

If `"part2"` is the expected answer, the score is `30`.

Else, if there are leftover letters and `"part2"` ends with those in order, the score is `15`.

Else, the score is `0`.

Scoring Summary

The score range for part 1 is `[0, 20]` and that for part 2 is `[0, 30]`.

There will be four evaluation attempts, summing to a maximum of `200` before that raw score is halved (rounding down), then multiplied by the challenge weightage.

Example

See here.