Solution for: OkCupid Challenge

Issue: OkCupid is an online dating site has an interesting matching algorithm. You're going to implement it. See the OkCupid's description of their algorithm. Write a program in the language of your choice that reads a set of user profiles from stdin (represented in JSON) and writes the top 10 matches for each user profile to stdout (also in JSON), sorted in rank order.

Guidelines:

In order to calculate the “True Match Score” between to profiles, we got to take in count the following details:

- For each profile we have:
    - Answers of questions.
    - Acceptable answers from other users
    - Importance of the question.

- The “importance” level of a question can be “Irrelevant” (0 points), “A little important” (1 point), “Somewhat important” (10 points), “Very important” (50 points), and “Mandatory” (250 points). Then, given a set of questions, we will have, for each profile, an amount of “importance points”.

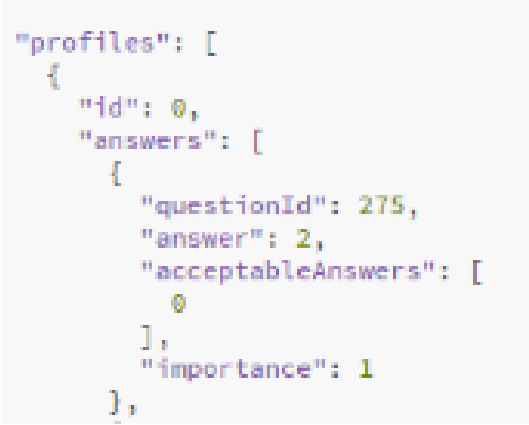
- On the other hand, if a user answers a question in the way we expected (the user gives an acceptable answer), it will have some “earned importance points”. The relation “earned importance points” per “importance points” help us to obtain a “satisfaction percentage”, that is, how satisfactory a user feels according to the answers of another one.

- The Match Percentage is defined by the Square Root of the product between the Satisfaction percentages of two different profiles. Also, in order to normalize the result, there is a Reasonable margin of error, defined as 1 divided by the number of questions. The Match Percentage minus Reasonable margin of error, results in the “True Match Score”.

- If a user states that expects any kind of answer for a specific question, it means that question is “irrelevant” (importance zero).

The plan:

Read the input.json file. The model of the information is:



```
"profiles": [
  {
    "id": 0,
    "answers": [
      {
        "questionId": 275,
        "answer": 2,
        "acceptableAnswers": [
          0
        ],
        "importance": 1
      },
```

Loop 1: Walk through the profiles. This will serve as profile A.

Loop 2: Nested inside of loop 1, walking through the profiles again. This will serve as profile B. This means we are going to loop n * n times (where n is the number of profiles) to compare all the profiles (profile A vs profile B).

Loop 3: Nested inside Loop 2, walking through the array of answers of profile B.

Loop 4: Nested inside Loop 3, walking through the answers of profile A. In order to calculate the Satisfaction percentages, it is needed all the questions that both profile A and profile B answered. Therefore, for every answer of profile B, we need to check if profile A answered the same question (questionId), in order to get the answers to compare. Every time the loop check and find a question that both profiles answered, proceeds to:

1. Collect the "importance points" for that question, for both profile A and Profile B.

2. Collect the amount of questions answered by both profiles (S), needed to calculate the "Reasonable margin of error".

3. Profile A side: Check if the answer from profile B is an acceptable answer according to profile A. If true, check if acceptable answers from profile A contains all possible answers. This means that the question is irrelevant for profile A. In that case, you will have zero "earned importance points" for that question. Otherwise, collect the importance points of the question as "earned importance points" for profile A.

4. The same as above, but from the Profile B side.

5. Close loops 3 and 4. After the loops 3 and 4 have finished their execution, we have all the data needed to calculate the "True Match percentage score" as it follows:

a. Calculate the “satisfaction percentage” from Profile A vs Profile B, that is, (earned importance points) / (importance points). The same for Profile B vs Profile A.

b. Calculate the “Reasonable margin of error”: 1 / (number of questions answered by both profiles).

c. Now, calculate the “True Match score” as the square root of the product between the satisfaction percentages obtained in step 1, minus the reasonable margin of error obtained in step 2.

d. Store the score as a pair <profile Id (profile B) – Score> in an array (ranking).

6. Close loop 2. Sort the ranking array in descending order according the scores, and slice the array to obtain the top 10 matches. Finally, store the pair <profile Id (profile A) – Array of matches> in an array (Results).

7. You can examine the results on the terminal, or check the output.json file.