Welcome!

This is a practical quiz of your software engineering and data science skills. It covers a broad range of topics, and and we don't expect you to get every question 100% right. Feel free to use external resources like Google or a calculator.
This quiz will be timed so try to set aside 60 - 90 minutes to take this quiz. If it takes you longer, that's ok. We'd rather you finish with great answers in two hours than submit incomplete answers in 90 minutes.

1. **Data Processing**
Write a program in a language of your choice to read in two CSVs with the same columns as those below, inner join them on userid, and return the joined rows. Instead of two separate min_score columns, combine them into a single min_score column filled with the min of the min_scores from each input csv.

Example:
input1.csv:
```
userid,age,min_score
1,23,6
6,54,300
2,39,40
```

input2.csv:
```
userid,state,min_score
1,CA,12
6,AK,44
2,WA,100
```

output.csv:
```
userid,age,state,min_score
1,23,CA,6
6,54,AK,44
2,39,WA,40
```

Answer:

2. SQL
You have an SQL table (in your favorite SQL database) that looks like this:
```
userid      date        city
1           2015-03-01  Seattle
1           2015-04-06  Portland
```

```
22          2015-02-23  New York
…
```

Write a query to count the number of rows per month for Seattle.
Answer:

Write a query to identify the month with the most rows for Seattle.
Answer:

What flavor of SQL did you write your above answers in?
Answer:

3. Broken Outlier Detector

The code below is intended to take a list of numbers, leave the original list unchanged, and return a new list with any values that are more than 3 standard deviations from the median removed. It should raise an exception if more than 10% of the numbers were removed.

The Code:

```
def strip_outliers(values):
    values.sort()
    median = values[len(values)/2]

    variance = 0
    for i in range(len(values) - 1):
        variance += values[i] - median ** 2 / n
    std_dev = variance ** 2

    min_ok,max_ok = median-3*std_dev, med+3*std_dev
    result = filter(values, lambda v: v <= min_ok or v >= max_ok)
    if len(result) < 0.1 * len(values):
        raise Exception("strip_outliers: too many outliers removed")
    return result
```

Identify all the correctness issues.
Answer:

Identify ways you could improve its performance.
Answer:

4. Languages

What is your favorite programming language? Why? What is your least favorite language? Why?
Answer:

5. Tic-Tac-Toe Design
Think about how to design a tic-tac-toe game that runs on a computer. It should allow any mix of players and AIs to play. It should detect when the game is over and communicate the winner. You're welcome to use a functional or object oriented approach. Don't implement it. Instead, write down the public signatures (names, names of arguments, and argument + return types) for the core functions and/or classes in pseudocode. Feel free to add comments to explain, but keep it high level. Aim for less than 15 minutes on this question.

Answer:

6. Something is Wrong
Imagine you just logged into a Linux server to check on how your data pipeline is running. You notice that a resource intensive job that is usually CPU bound and takes an hour to run has been running for 8 hours.

What could have happened? List your top several hypotheses.
Answer:

For each hypothesis, exactly how do you test it? What tools do you use?
Answer:

7. Storage
Sort these by how long it takes to read one random byte of data: SSD, CPU L2 cache, S3 (accessed from your laptop), redis*, RAM, HDD.

*Assume that the redis server is running on a separate machine in the same building as your client, and that the server and client have a wired ethernet connection between them.
Answer:

8. Performance Issues
Imagine we have a system that's responsible for counting the total number of pages viewed on a website. It's a single process on a single machine, and it has a RESTful HTTP API with one call. It looks like this:
`GET /increment?n=N`

Whenever this endpoint is hit, the system will increment its internal count by N, open its state file on disk, overwrite the state file with the new value of N, and close the file.

There are many webserver processes on many machines, all calling into this system. Every time a webserver sees a web request, it calls out to the counter system like this:
`GET /increment?n=1`

Unsurprisingly, the system is having trouble keeping up. What steps would you take to improve the throughput of the system?
Answer:

Are there any tradeoffs you consider as you take those steps?
Answer: