FANO 有光科技 Labs

INSTRUCTIONS:

You will have four (4) hours to work on the problems below, counting from the minutes you started reading this instruction. Once the time is up, you should package all the source code you have written into a .zip file and send it to your interviewer. We might accept delayed submissions when given some reasonable explanations but please try to be punctual. Please note the following:

1. You could use **any programming language** to complete the coding tests, although we do prefer seeing **Python 3** and **Java 8**.

2. You are free to use any search engine or references during the coding test but you **must write the code yourself**. Copying and pasting others' code or having someone else write the code for you will be considered a plagiarism, and, therefore, permanently disqualify you from working at Fano Labs.

3. **NO third-party libraries** in your code. We want to test how well you know the basics of computer programming and algorithms and, therefore, you are not supposed to use any third-part high-level libraries to do the tasks.

4. We **expect your code to run correctly**.

5. In the worst case, if you already know how to solve a particular problem but doesn't have enough time to make your program work, please write a **README.txt file** that explains how your logic works and include some pseudocode.

6. You should **write your own test cases** and ensure that your code runs. The test cases, together with test data (if necessary), should be included in the .zip file you send us.

7. **Pay special attention to your coding style** and make sure that your code is readable. Messy codes will result in low marks, even though it passes our test cases.

8. In the .zip file you are to submit, please create **one directory/folder to store your codes for each problem**. For instance, you may create a P1 folder for codes related with problem 1, a P2 folder for codes related with problem 2, and so on.

9. This problem set given to you is **strictly private and confidential**. Circulating it without prior approval from Fano Labs is prohibited.

PROBLEM SETS:

1. Implement a LinkedList data structure using one of your preferred language.

    1) Now given a linked list, write a function/method that shuffles the nodes in a way that the all originally odd-numbered nodes are placed before the even-numbered nodes. Note: The number is the original index of the given linked list and has nothing to do with the data stored. For example: a linked list of #1->#2->#3->#4->#5 should be converted into #1->#3->#5->#2->#4.

    2) Write an algorithm that detects whether there are loops inside the linked list. Analyze the algorithm complexity.

    3) Improve the aforementioned algorithm by returning the node in the linked list where the loop starts. Returns None if no loops are detected.

2. Write a program that reads a text file from the same directory that your main program is stored and then perform some operations on it. Concretely:

    1) Your program reads from the console (1) the name of the file ("test.txt" for instance) and (2) an integer *k*.

    2) Your program then reads the file. While reading the file (NOT after the file has been read), you need to count the occurrence of each word. Note that "cat" and "cats" could be considered as two different words. In addition, special characters such as ./*? are all separate words. For instance, "I'm having some headache" has six words ["I", "'", "m", "having", "some", "headache"].

    3) Once the file-reading is completed, your program should output the EXACT (not approximate) the top *k* most frequent words. In case there are ties in terms of word frequencies, return all.

    4) You must ensure that your program runs **as efficiently as possible** because, in reality, your program might be reading a file with billions of words and a very large *k* value. Please provide the algorithmic complexity of your code.

3. The 2048 Game was a popular mini-game that came out in around the year of 2014. The game is simple. You will first be given a four-by-four grid of 16 tiles, each tile would hold an integer. At the beginning of the game, two random tiles on the grid would be filled by two number two's, two number four's, or a number two and a number four. When the game starts, you can move all tiles up, down, left, or right, trying to join two equal numbers. If two tiles of equal numbers are in touch, they will add up and become one tile. After every move, one extra number (either a two or a four) appear in a random empty cell on the grid. The game ends if there are no more free tiles and that you cannot make any moves. You will win the game if the number 2048 appears on the grid. To get a feel of this game, please visit this website: http://2048game.mobi/.

    1) Please write a class, *Game*, to simulate the 2048 game. The *Game* class should contain a four-by-four two-dimensional array, *grid[4][4]*, which represents the status of the grid. Set *grid[i][j] = 0* to represent an empty tile at position *(i, j)*. This class should also contain a method, *move()*, which should take a parameter that specifies which direction related with

this move. Feel free to define and implement any extra methods needed to make this *Game* class complete. Note: Don't forget to define a proper initializer method.

2) Using the *Game* class defined above, write a program and implement a strategy to play the 2048 game as optimally as possible.