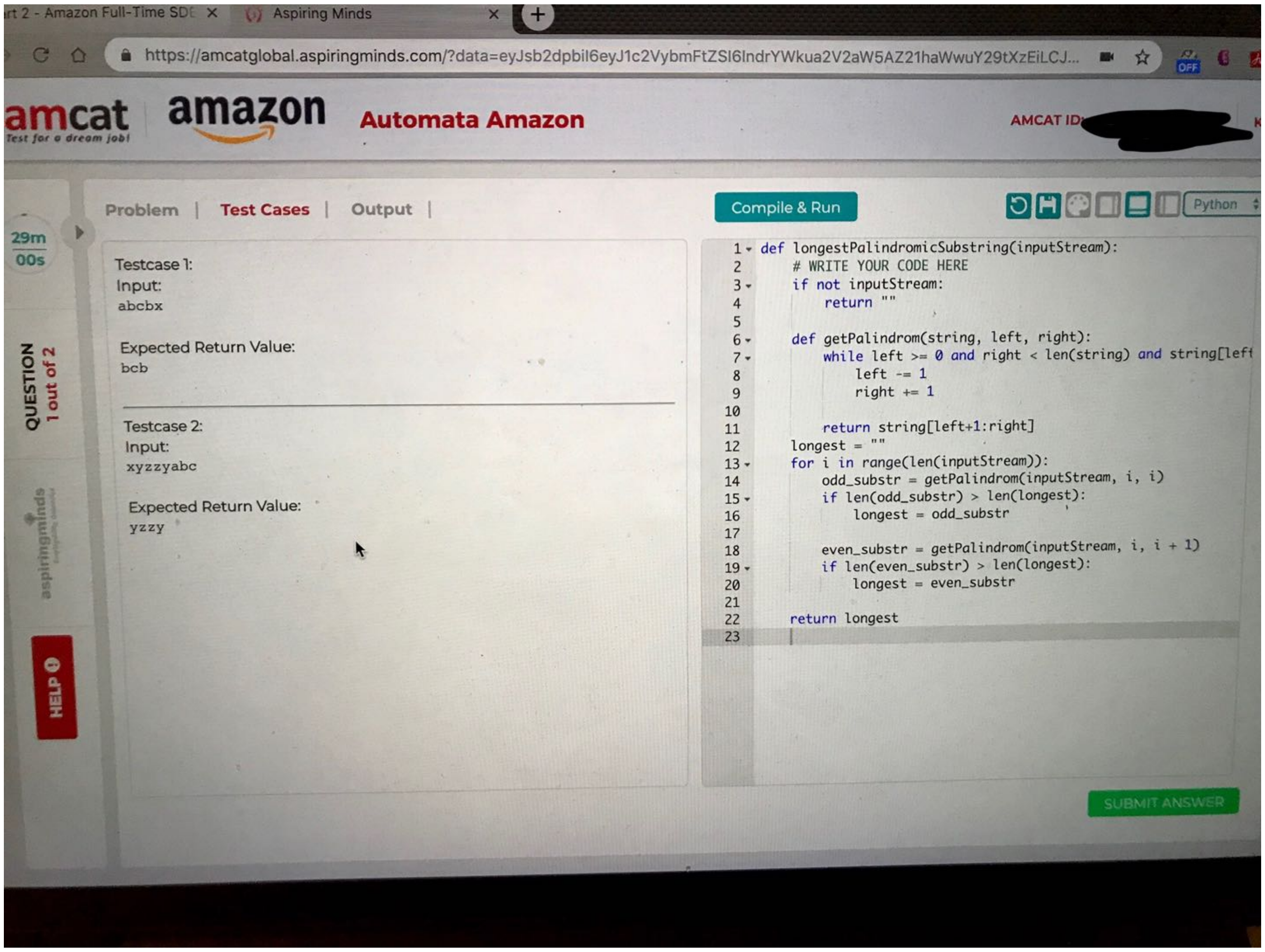
1. Longest Palindrome Substring

input: abcbx return bcb

input: xyzzyabc return yzzy



Testcase 1:
Input:
abcbx

Expected Return Value:
bcb

Testcase 2:
Input:
xyzzyabc

Expected Return Value:
yzzy

```python
def longestPalindromicSubstring(inputStream):
    # WRITE YOUR CODE HERE
    if not inputStream:
        return ""

    def getPalindrom(string, left, right):
        while left >= 0 and right < len(string) and string[left
            left -= 1
            right += 1

        return string[left+1:right]
    longest = ""
    for i in range(len(inputStream)):
        odd_substr = getPalindrom(inputStream, i, i)
        if len(odd_substr) > len(longest):
            longest = odd_substr

        even_substr = getPalindrom(inputStream, i, i + 1)
        if len(even_substr) > len(longest):
            longest = even_substr

    return longest
```

Leetcode 5. Longest Palindromic Substring

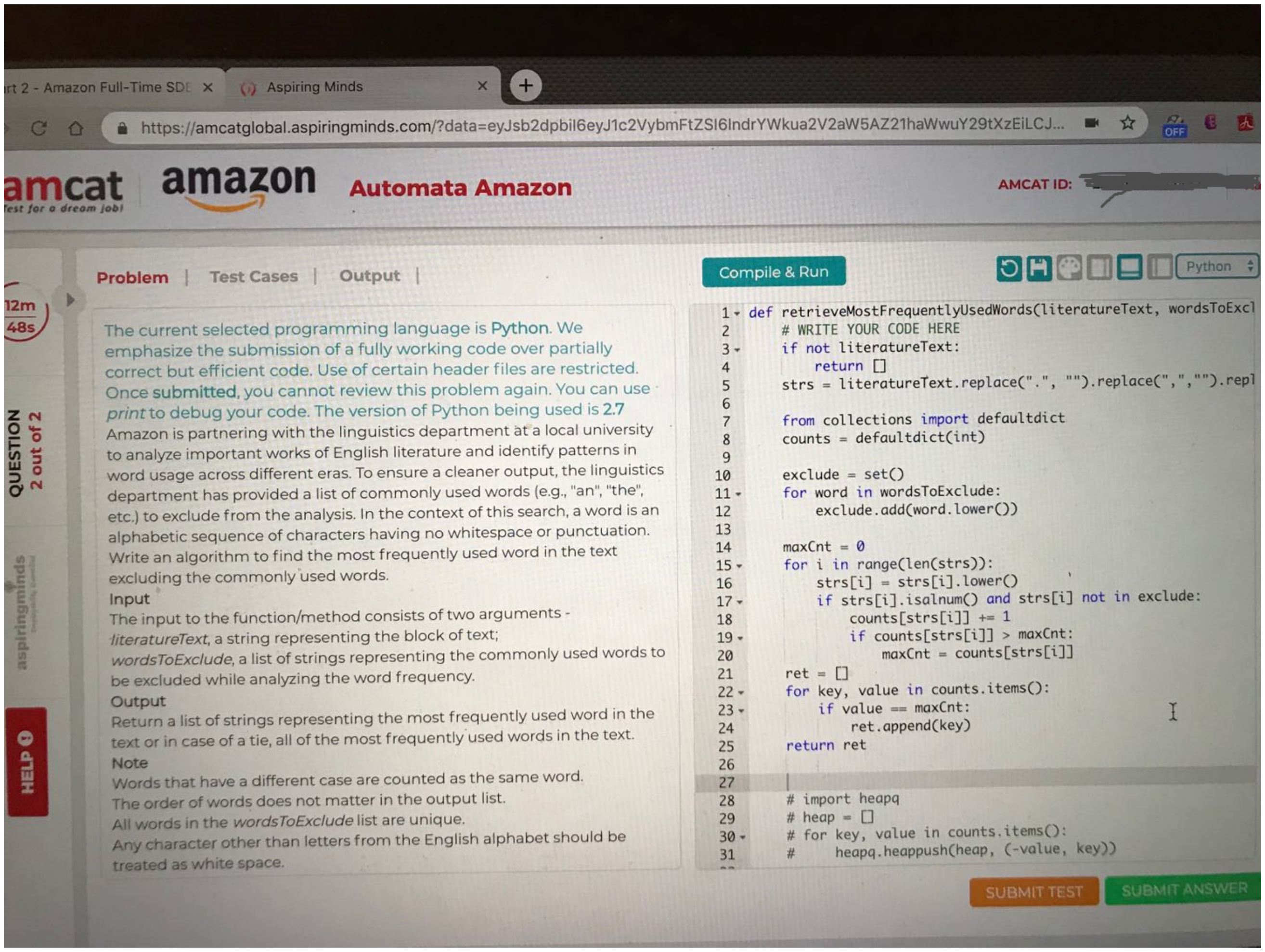
2. Most Frequent Used Words

You are given a sentence string and a list with words to exclude. Find the most frequent used words in the sentence but not in the words-to-exclude list.

input: "Jack and Jill went to the market to buy bread and cheese. Cheese is Jack's and Jill's favority food."

["and", "he", "the", "to", "is", "Jack", "Jill"].

return: ["cheese", "s"]



The current selected programming language is Python. We emphasize the submission of a fully working code over partially correct but efficient code. Use of certain header files are restricted. Once **submitted**, you cannot review this problem again. You can use *print* to debug your code. The version of Python being used is 2.7

Amazon is partnering with the linguistics department at a local university to analyze important works of English literature and identify patterns in word usage across different eras. To ensure a cleaner output, the linguistics department has provided a list of commonly used words (e.g., "an", "the", etc.) to exclude from the analysis. In the context of this search, a word is an alphabetic sequence of characters having no whitespace or punctuation. Write an algorithm to find the most frequently used word in the text excluding the commonly used words.

Input

The input to the function/method consists of two arguments -
literatureText, a string representing the block of text;
wordsToExclude, a list of strings representing the commonly used words to be excluded while analyzing the word frequency.

Output

Return a list of strings representing the most frequently used word in the text or in case of a tie, all of the most frequently used words in the text.

Note

Words that have a different case are counted as the same word.
The order of words does not matter in the output list.
All words in the *wordsToExclude* list are unique.
Any character other than letters from the English alphabet should be treated as white space.

```
def retrieveMostFrequentlyUsedWords(literatureText, wordsToExcl
    # WRITE YOUR CODE HERE
    if not literatureText:
        return []
    strs = literatureText.replace(".", "").replace(",","").repl

    from collections import defaultdict
    counts = defaultdict(int)

    exclude = set()
    for word in wordsToExclude:
        exclude.add(word.lower())

    maxCnt = 0
    for i in range(len(strs)):
        strs[i] = strs[i].lower()
        if strs[i].isalnum() and strs[i] not in exclude:
            counts[strs[i]] += 1
            if counts[strs[i]] > maxCnt:
                maxCnt = counts[strs[i]]
    ret = []
    for key, value in counts.items():
        if value == maxCnt:
            ret.append(key)
    return ret


    # import heapq
    # heap = []
    # for key, value in counts.items():
    #     heapq.heappush(heap, (-value, key))
```

Leetcode 819. Most Common Word

3. Count number of substrings with exactly k distinct characters

https://www.geeksforgeeks.org/count-number-of-substrings-with-exactly-k-distinct-characters/

similar to leetcode340. Longest Substring with At Most K Distinct Characters

4. MaximumMinimumPath

You are gonna climb mountains represented by a matrix. Each integer in the matrix represents the altitude at that point. You are supposed to climb from the top-left corner to the bottom-right corner and only move rightward or downward in each step.

You can have many paths to do so. Each path has a minimum altitude. Find the maximum among all the minimum altitudes of all paths.

e.g.

5 4 5 1 3 6

Three paths: 5 1 3 6, 5 4 3 6, 5 4 5 6. Minimums are 1, 3, 4 respectively. Return the maximum in them which is 4.

another example:

```
[8, 4, 7]
[6, 5, 9]
3 paths:
8-4-7-9 min: 4
8-4-5-9 min: 4
8-6-5-9 min: 5
return: 5
```

You should use DP. Similar to Leetcode 174. Dungeon Game

DP code:

```
int helper(int[][] matrix){
    int[] result = new int[matrix[0].length];
    result[0] = matrix[0][0];
    for(int i=1; i<matrix[0].length; i++){
        result[i] = Math.min(result[i-1], matrix[0][i]);
    }
    for(int i=1; i<matrix.length; i++){
        result[0] = Math.min(matrix[i][0], result[0]);
        for(int j=1; j<matrix[0].length; j++){
            result[j] = (result[j]<matrix[i][j] && result[j-1]<matrix[i][j])?
Math.max(result[j-1], result[j]):matrix[i][j];
        }
    }
    return result[result.length-1];
}
```

DFS code:

```
 public class MaximumMinimumPath {
    private int min, max, row, col;
    public int maxMinPath(int[][] matrix) {
        row = matrix.length;
        col = matrix[0].length;
        min = Integer.MAX_VALUE;
        max = Integer.MIN_VALUE;
        dfsHelper(matrix, min, 0, 0);
     return max;
    }
    public void dfsHelper(int[][] matrix, int min, int i, int j ){

        if (i >= row || j >= col) return;
        if (i == row - 1 && j == col - 1) {
            min = Math.min(min, matrix[i][j]);
            max = Math.max(max, min);
            return;
        }
        min = Math.min(min, matrix[i][j]);
        dfsHelper(matrix, min, i, j + 1);
        dfsHelper(matrix, min, i + 1, j);
    }
}
```

5. Subtree: Maximum average node

Description Given a binary tree, find the subtree with maximum average. Return the root of the subtree.

Example Given a binary tree:

1 / \ -5 11 / \ / \ 1 2 4 -2

return the node 11.

Your problem can be different in ways like -- it may not be a binary tree or the average doesn't include the root of the substree.

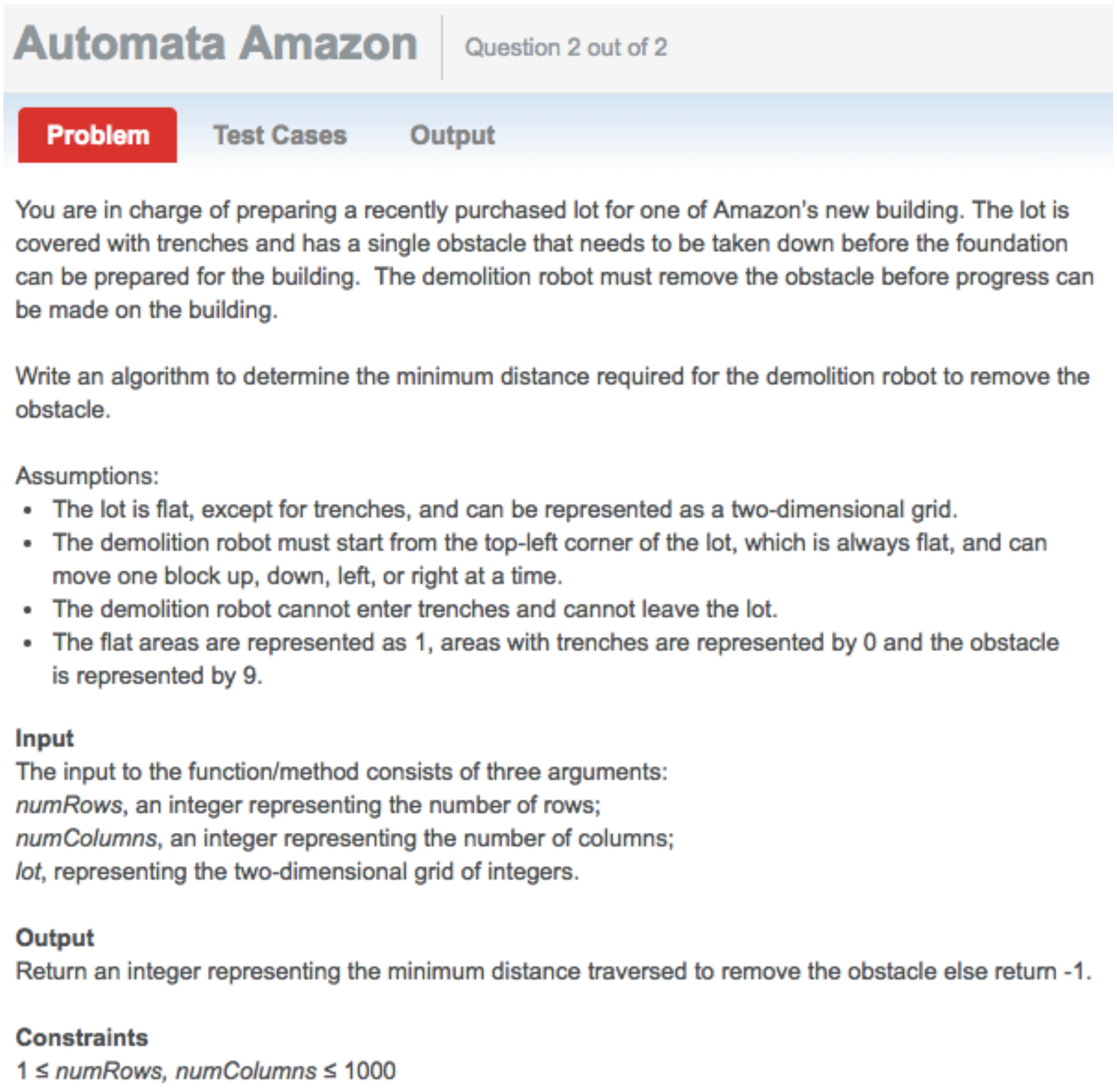
reference code:

Code

```
/**
 * Definition of TreeNode:
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left, right;
 *     public TreeNode(int val) {
 *         this.val = val;
 *         this.left = this.right = null;
 *     }
 * }
 */
public class Solution {
    /**
     * @param root the root of binary tree
     * @return the root of the maximum average of subtree
     */

    class ResultType {
        TreeNode node;
        int sum;
        int size;
        public ResultType(TreeNode node, int sum, int size) {
            this.node = node;
            this.sum = sum;
            this.size = size;
        }
    }

    private ResultType result = null;

    public TreeNode findSubtree2(TreeNode root) {
        // Write your code here
        if (root == null) {
            return null;
        }

        ResultType rootResult = helper(root);
        return result.node;
    }

    public ResultType helper(TreeNode root) {
        if (root == null) {
            return new ResultType(null, 0, 0);
        }

        ResultType leftResult = helper(root.left);
        ResultType rightResult = helper(root.right);

        ResultType currResult = new ResultType(
                    root,
                    leftResult.sum + rightResult.sum + root.val,
                    leftResult.size + rightResult.size + 1);

        if (result == null
            || currResult.sum * result.size > result.sum * currResult.size) {
            result = currResult;
        }

        return currResult;
    }
```

```
  }
```

6. K Minimum Distances



You are in charge of preparing a recently purchased lot for one of Amazon's new building. The lot is covered with trenches and has a single obstacle that needs to be taken down before the foundation can be prepared for the building. The demolition robot must remove the obstacle before progress can be made on the building.

Write an algorithm to determine the minimum distance required for the demolition robot to remove the obstacle.

Assumptions:

- The lot is flat, except for trenches, and can be represented as a two-dimensional grid.
- The demolition robot must start from the top-left corner of the lot, which is always flat, and can move one block up, down, left, or right at a time.
- The demolition robot cannot enter trenches and cannot leave the lot.
- The flat areas are represented as 1, areas with trenches are represented by 0 and the obstacle is represented by 9.

Input
The input to the function/method consists of three arguments:
numRows, an integer representing the number of rows;
numColumns, an integer representing the number of columns;
lot, representing the two-dimensional grid of integers.

Output
Return an integer representing the minimum distance traversed to remove the obstacle else return -1.

Constraints
1 ≤ *numRows, numColumns* ≤ 1000

Input
The input to the function/method consists of three arguments:
numRows, an integer representing the number of rows;
numColumns, an integer representing the number of columns;
lot, representing the two-dimensional grid of integers.

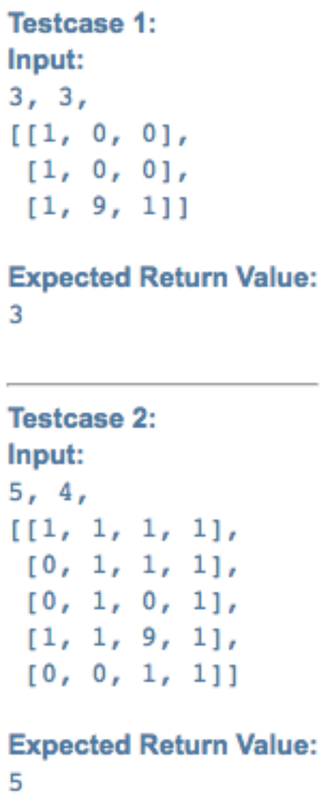
Output
Return an integer representing the minimum distance traversed to remove the obstacle else return -1.

Constraints
1 ≤ *numRows, numColumns* ≤ 1000

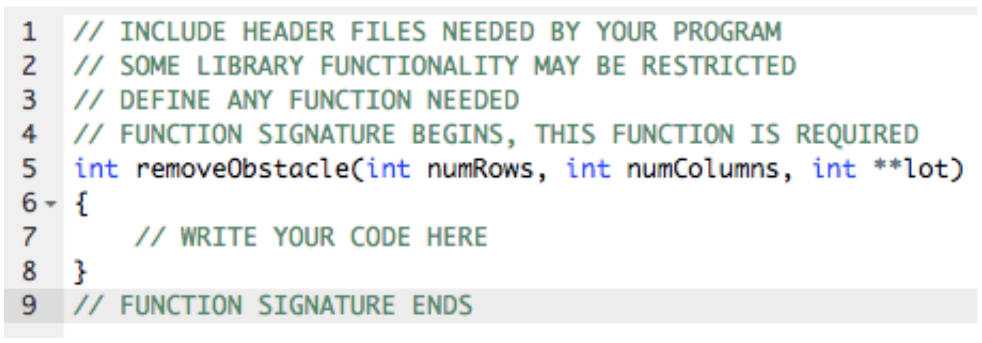
Example
Input:
numRows = 3
numColumns = 3
lot =
[[1, 0, 0],
[1, 0, 0],
[1, 9, 1]]

Output:
3

Explanation:
Starting from the top-left corner, the demolition robot traversed the cells (0,0) -> (1,0) -> (2,0) -> (2,1).
The robot traversed the total distance 3 to remove the obstacle.
So, the output is 3.


Testcase 1:
Input:
```
3, 3,
[[1, 0, 0],
 [1, 0, 0],
 [1, 9, 1]]
```
Expected Return Value:
```
3
```

Testcase 2:
Input:
```
5, 4,
[[1, 1, 1, 1],
 [0, 1, 1, 1],
 [0, 1, 0, 1],
 [1, 1, 9, 1],
 [0, 0, 1, 1]]
```
Expected Return Value:
```
5
```

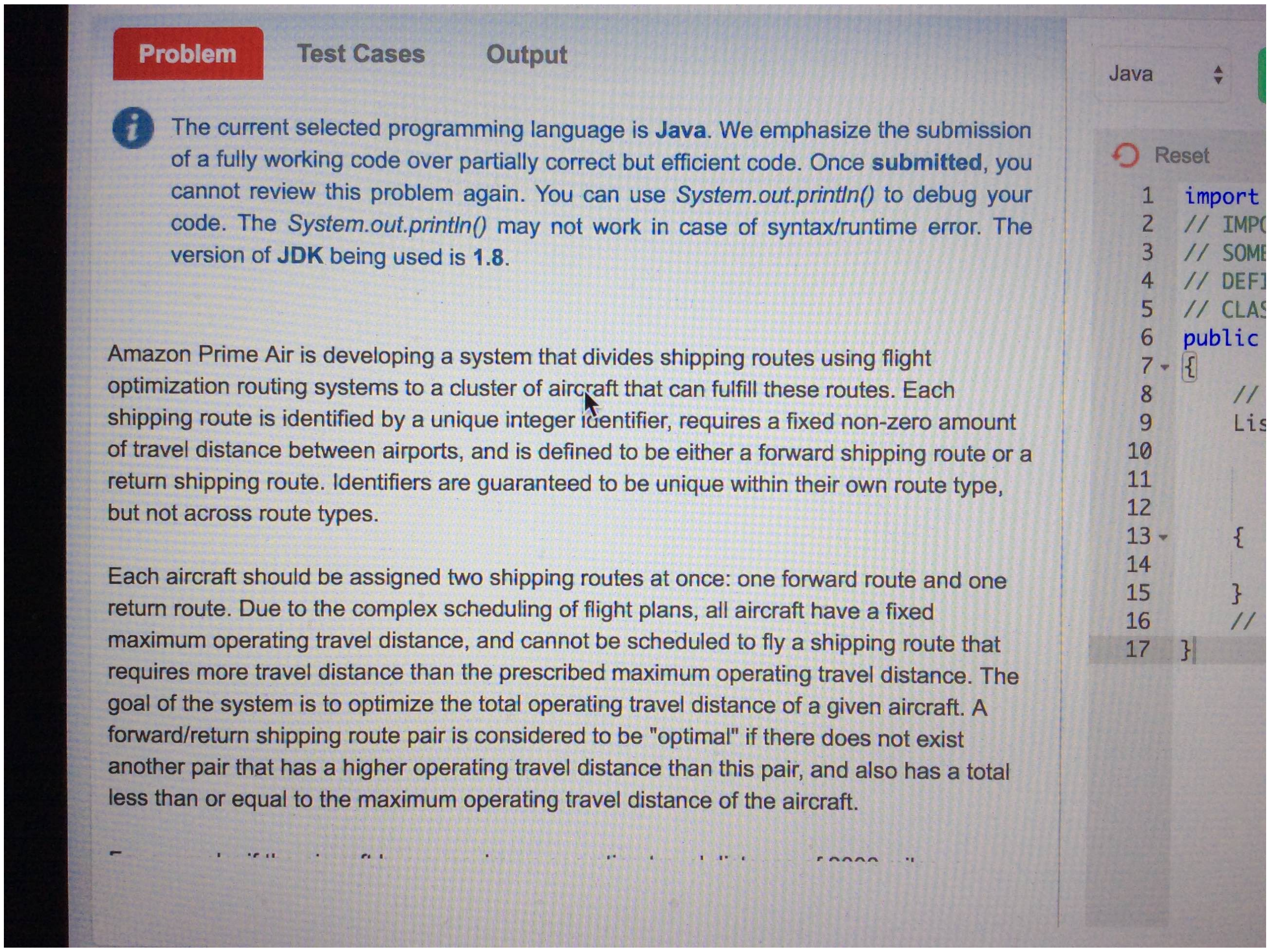

```c
// INCLUDE HEADER FILES NEEDED BY YOUR PROGRAM
// SOME LIBRARY FUNCTIONALITY MAY BE RESTRICTED
// DEFINE ANY FUNCTION NEEDED
// FUNCTION SIGNATURE BEGINS, THIS FUNCTION IS REQUIRED
int removeObstacle(int numRows, int numColumns, int **lot)
{
    // WRITE YOUR CODE HERE
}
// FUNCTION SIGNATURE ENDS
```

7. K Nearest Points

The background is to load goods into a truck. The truck is at the origin and you are given the coordinates of all goods. Find the k nearest goods.

Use PriorityQueue. Write your own Comparator.

8. Flight



Problem Test Cases Output

The current selected programming language is **Java**. We emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use *System.out.println()* to debug your code. The *System.out.println()* may not work in case of syntax/runtime error. The version of **JDK** being used is **1.8**.

Amazon Prime Air is developing a system that divides shipping routes using flight optimization routing systems to a cluster of aircraft that can fulfill these routes. Each shipping route is identified by a unique integer identifier, requires a fixed non-zero amount of travel distance between airports, and is defined to be either a forward shipping route or a return shipping route. Identifiers are guaranteed to be unique within their own route type, but not across route types.

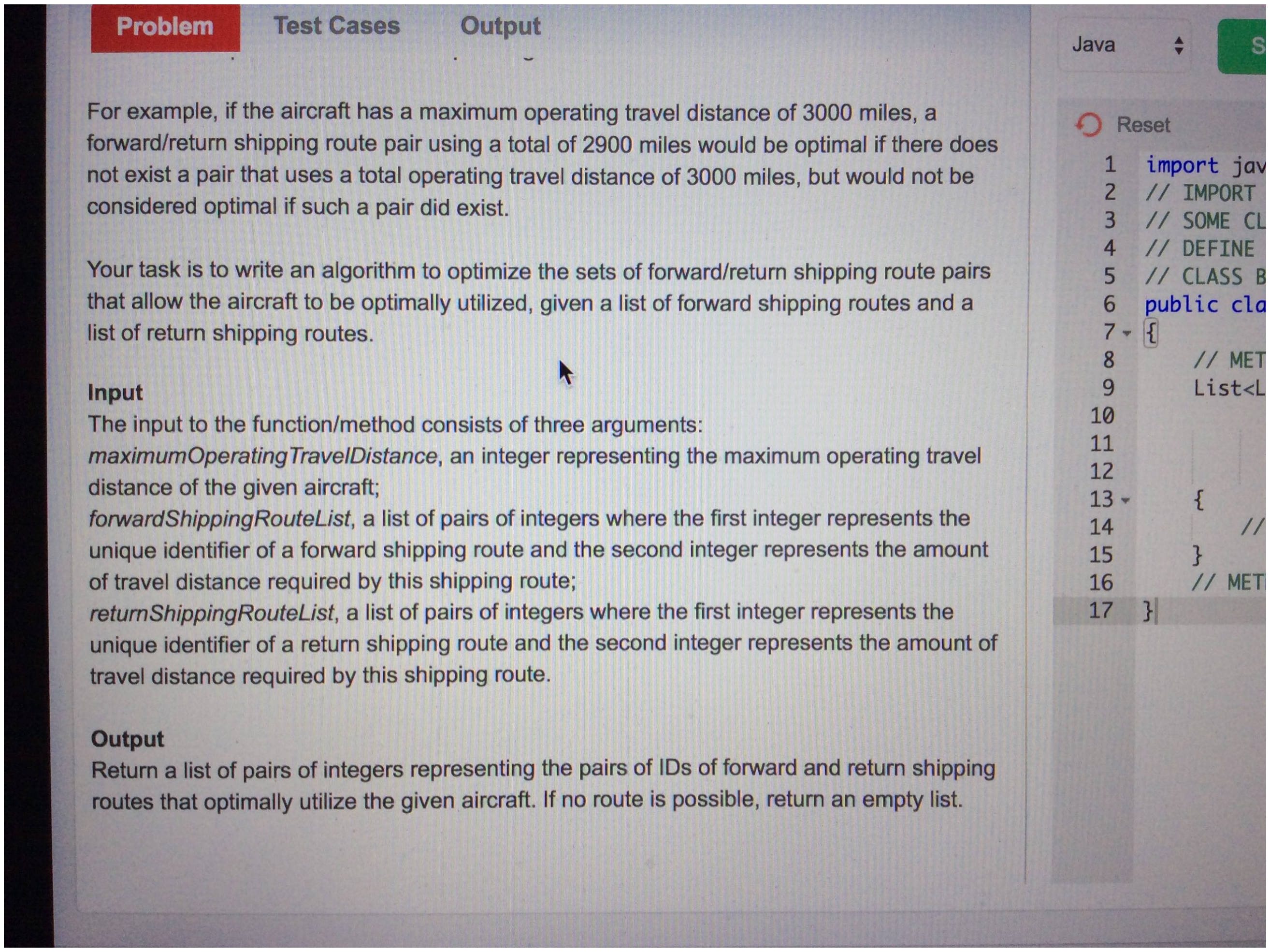
Each aircraft should be assigned two shipping routes at once: one forward route and one return route. Due to the complex scheduling of flight plans, all aircraft have a fixed maximum operating travel distance, and cannot be scheduled to fly a shipping route that requires more travel distance than the prescribed maximum operating travel distance. The goal of the system is to optimize the total operating travel distance of a given aircraft. A forward/return shipping route pair is considered to be "optimal" if there does not exist another pair that has a higher operating travel distance than this pair, and also has a total less than or equal to the maximum operating travel distance of the aircraft.

[illegible]

Java

Reset

```
import
// IMPO
// SOME
// DEFI
// CLAS
public
{
    //
    Lis


    {
    }
    //
}
```



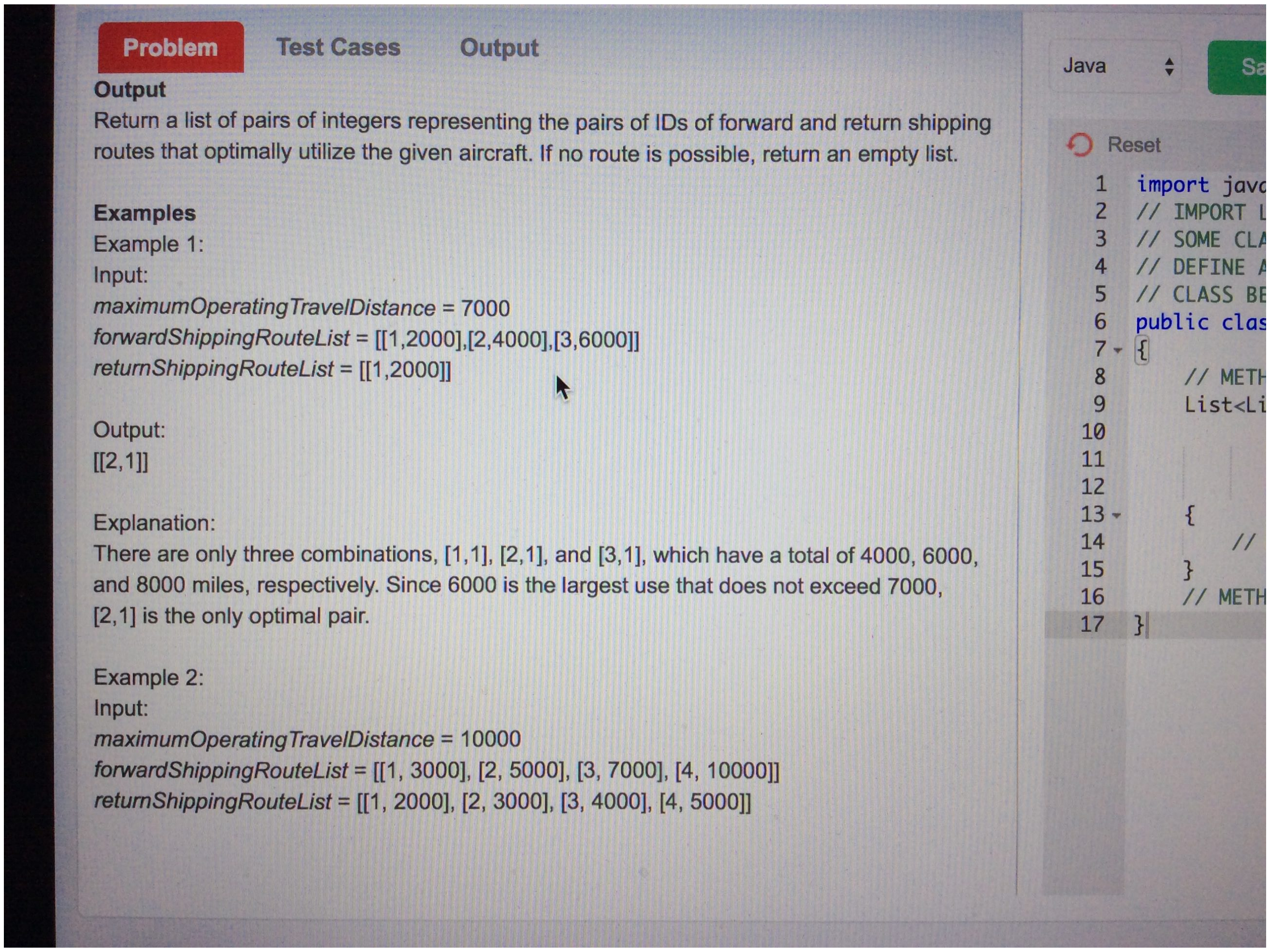
For example, if the aircraft has a maximum operating travel distance of 3000 miles, a forward/return shipping route pair using a total of 2900 miles would be optimal if there does not exist a pair that uses a total operating travel distance of 3000 miles, but would not be considered optimal if such a pair did exist.

Your task is to write an algorithm to optimize the sets of forward/return shipping route pairs that allow the aircraft to be optimally utilized, given a list of forward shipping routes and a list of return shipping routes.

Input
The input to the function/method consists of three arguments:
maximumOperatingTravelDistance, an integer representing the maximum operating travel distance of the given aircraft;
forwardShippingRouteList, a list of pairs of integers where the first integer represents the unique identifier of a forward shipping route and the second integer represents the amount of travel distance required by this shipping route;
returnShippingRouteList, a list of pairs of integers where the first integer represents the unique identifier of a return shipping route and the second integer represents the amount of travel distance required by this shipping route.

Output
Return a list of pairs of integers representing the pairs of IDs of forward and return shipping routes that optimally utilize the given aircraft. If no route is possible, return an empty list.

```
import jav
// IMPORT
// SOME CL
// DEFINE
// CLASS B
public cla
{
    // MET
    List<L



    {
        //
    }
    // MET
}
```



Output
Return a list of pairs of integers representing the pairs of IDs of forward and return shipping routes that optimally utilize the given aircraft. If no route is possible, return an empty list.

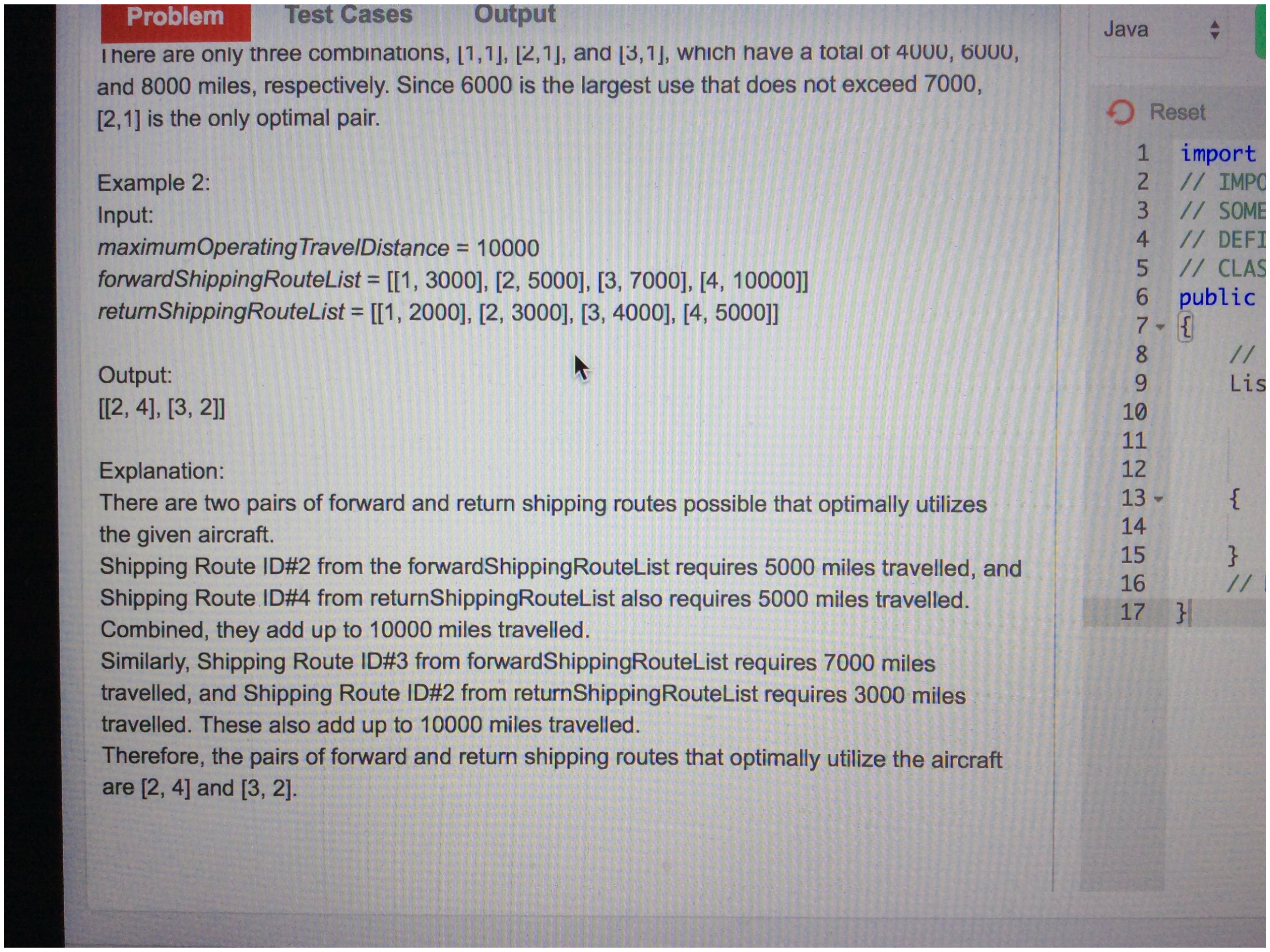
Examples
Example 1:
Input:
maximumOperatingTravelDistance = 7000
forwardShippingRouteList = [[1,2000],[2,4000],[3,6000]]
returnShippingRouteList = [[1,2000]]

Output:
[[2,1]]

Explanation:
There are only three combinations, [1,1], [2,1], and [3,1], which have a total of 4000, 6000, and 8000 miles, respectively. Since 6000 is the largest use that does not exceed 7000, [2,1] is the only optimal pair.

Example 2:
Input:
maximumOperatingTravelDistance = 10000
forwardShippingRouteList = [[1, 3000], [2, 5000], [3, 7000], [4, 10000]]
returnShippingRouteList = [[1, 2000], [2, 3000], [3, 4000], [4, 5000]]

```
import java
// IMPORT L
// SOME CLA
// DEFINE A
// CLASS BE
public clas
{
    // METH
    List<Li



    {
        //
    }
    // METH
}
```



There are only three combinations, [1,1], [2,1], and [3,1], which have a total of 4000, 6000, and 8000 miles, respectively. Since 6000 is the largest use that does not exceed 7000, [2,1] is the only optimal pair.

Example 2:
Input:
maximumOperatingTravelDistance = 10000
forwardShippingRouteList = [[1, 3000], [2, 5000], [3, 7000], [4, 10000]]
returnShippingRouteList = [[1, 2000], [2, 3000], [3, 4000], [4, 5000]]

Output:
[[2, 4], [3, 2]]

Explanation:
There are two pairs of forward and return shipping routes possible that optimally utilizes the given aircraft.
Shipping Route ID#2 from the forwardShippingRouteList requires 5000 miles travelled, and Shipping Route ID#4 from returnShippingRouteList also requires 5000 miles travelled. Combined, they add up to 10000 miles travelled.
Similarly, Shipping Route ID#3 from forwardShippingRouteList requires 7000 miles travelled, and Shipping Route ID#2 from returnShippingRouteList requires 3000 miles travelled. These also add up to 10000 miles travelled.
Therefore, the pairs of forward and return shipping routes that optimally utilize the aircraft are [2, 4] and [3, 2].

```
import
// IMPO
// SOME
// DEFI
// CLAS
public
{
    //
    Lis



    {

    }
    //
}
```

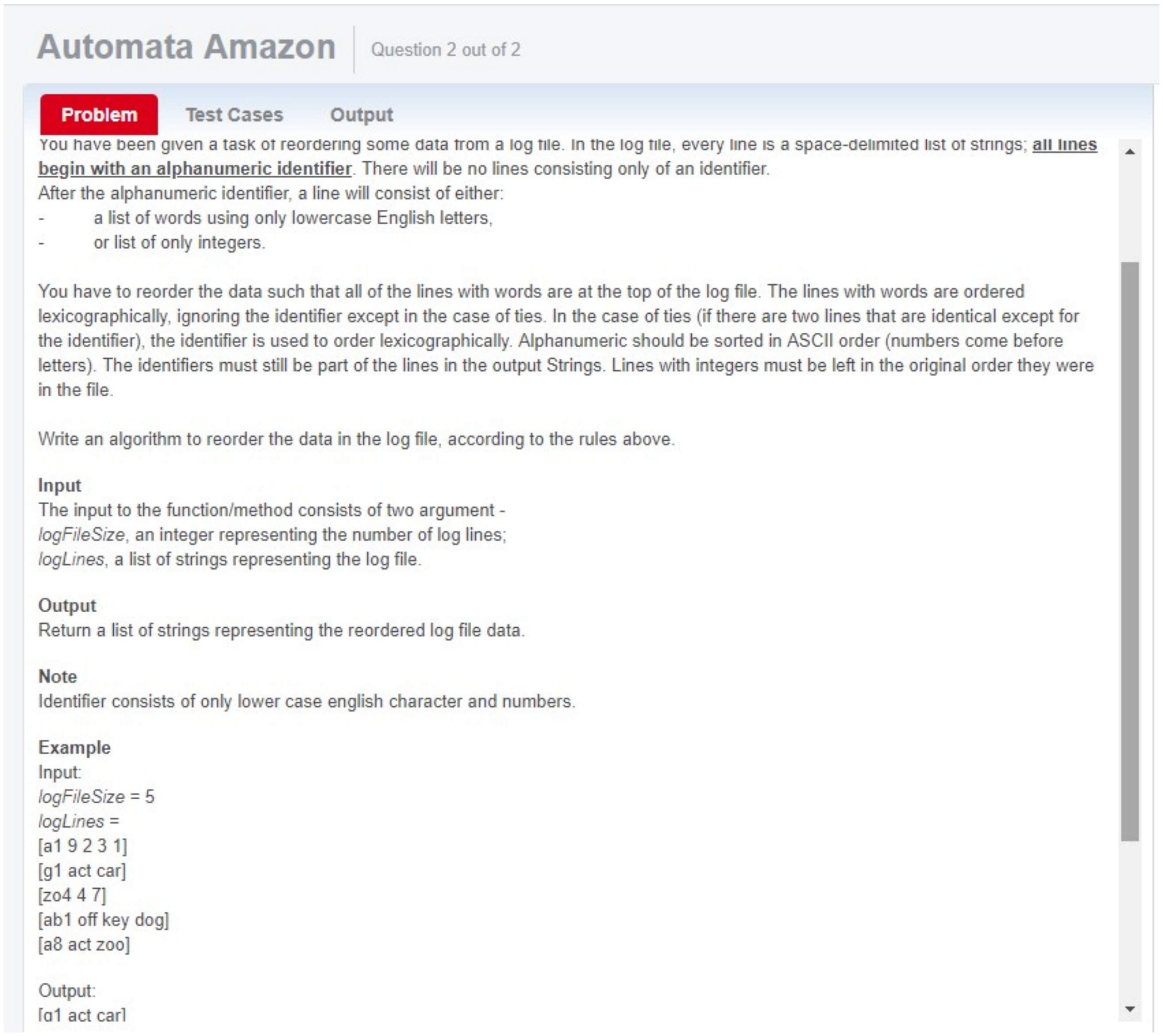
9. Robot

similar to Leetcode 505. The Maze II

10. Movies in flight

You are taking a flight and you wanna watch two movies. You are given int[] dur which includes all the movie durations. You are also given the duration of the flight which is d in minutes. Now, you need to pick two movies and the total duration of the two movies is less than or equal to (d - 30min). Find the pair of movies with the most total duration. If multiple found, return the pair with the longest movie.

11. Reorder Log Files


Automata Amazon | Question 2 out of 2

You have been given a task of reordering some data from a log file. In the log file, every line is a space-delimited list of strings; **all lines begin with an alphanumeric identifier**. There will be no lines consisting only of an identifier.
After the alphanumeric identifier, a line will consist of either:
- a list of words using only lowercase English letters,
- or list of only integers.

You have to reorder the data such that all of the lines with words are at the top of the log file. The lines with words are ordered lexicographically, ignoring the identifier except in the case of ties. In the case of ties (if there are two lines that are identical except for the identifier), the identifier is used to order lexicographically. Alphanumeric should be sorted in ASCII order (numbers come before letters). The identifiers must still be part of the lines in the output Strings. Lines with integers must be left in the original order they were in the file.

Write an algorithm to reorder the data in the log file, according to the rules above.

Input
The input to the function/method consists of two argument -
logFileSize, an integer representing the number of log lines;
logLines, a list of strings representing the log file.

Output
Return a list of strings representing the reordered log file data.

Note
Identifier consists of only lower case english character and numbers.

Example
Input:
logFileSize = 5
logLines =
[a1 9 2 3 1]
[g1 act car]
[zo4 4 7]
[ab1 off key dog]
[a8 act zoo]

Output:
[g1 act car]

12. Closet two sum

Given two arrays, find one number from each whose sum is <= the given limit. Find such two numbers with the biggest sum.

13. Given an array of letters and a window size k, return subarrays of size k with no duplicates.

e.g. [a, d, f, g, k, g] and window size k = 4

return [[a, d, f, g], [d, f, g, k]]

14. MST

Connect all cities with least cost.

Google minimum spanning tree for solutions like union find