Assignment: Portfolio

I. **For the problem A answer the below questions**

   Think of an approach to solve the problem, let us say approach 1 (Brute force is a valid approach to think of)

   1. Write the pseudocode / Description of approach 1
   2. Implement your solution
   3. What is the time complexity of the approach 1

   Think of a different approach to solve the problem, let us call this approach 2

   4. Write the pseudocode / Description of approach 2
   5. Implement your solution
   6. What is the time complexity of approach 2
   7. (Bonus) Does your approach 2 have improved time complexity compared to approach 1, if yes compare the time complexities. The bonus points will be awarded only if your approach 2 is an improvement over approach 1.
   8. (Bonus) How did you use the techniques learnt in this course to solve this problem? Name the algorithm techniques/strategies that you have used. (Brute force is not counted as an answer)

II. **For the problems B and C answer the below questions**

   1. Write a pseudocode / Description to solve the problem
   2. Implement your solution
   3. What is the time complexity of your solution?
   4. (Bonus) How did you use the techniques learnt in this course to solve this problem? Name the algorithm techniques/strategies that you have used. (Brute force and Recursion are not counted as an answer)

Problem A:

Name your function for approch1: **checkPalindrome_1(string, k)**

Name your function for approch2: **checkPalindrome_2(string, k)**

File name: **Palindrome.py**

You might be aware of palindrome strings. A word is a palindrome if it reads the same when it is reversed. Example: ‘madam’, ‘refer’

Given a string, find out if the string is a k-palindrome or not. A k-palindrome string becomes a palindrome string when k characters are removed from the original string. Return True if the string is k-palindrome, else return False. Return type is Boolean.

Assume that characters in the string will be of uniform case.

Examples:

1. Input: string: “abcdcba”; k =1

   Output: True

   Explanation: If we remove 1 character d from string “abcdcba”, the string becomes a 1-palindrome string – “abccba”

2. Input: string: “abcdeba”; k =2

Output: True

Explanation: By removing any characters ‘de’ from the string it does not become a 2-palindrome string.

Problem B:

Name your function: **pattermatch(string, p)**

File name: **Patternmatch.py**

Have you heard of pattern matching? Before we dive into the problem, let’s see an example. It will be helpful to know some terminology.

- ‘?’ can match any single character
- ‘*’ can match any sequence of characters (including empty sequence)

Let us see an example of pattern matching. Your function takes string and pattern (p). If the pattern matches the string return True otherwise return False. Return type is Boolean.

Example 1: String: “abcde”, pattern: “*”.

Output: True

The pattern matches the string as * can match any sequence of characters.

Example 2: String: “abcde”, pattern: “*a?c*”

Output: True

Let's match this backwards to understand this better. The last ‘*’ matches with sequence “de”. ‘?’ matches with “b”. Then 'a' matches with “a”. We have an extra ‘*’, which can match with an (imaginary) empty sequence before “a”.

Example 3: String: “abcde”, pattern: “ad”

Output: False

The pattern does not match with the string

Example 4: String: “abcde”, pattern: “ad?”

Output: False

The patten does not match with the string

Problem C:

Name your function: **getTesla(M)**

File name: **GetTesla.py**

Mr.X is in a maze. Tesla’s latest model car’s keys (K) are in the bottom right corner of the maze. Mr.X is located at the top left corner of the room. The rooms are connected in such a way that Mr.X can move only right or down in a single step.

At each step there is a energy booster (positive integers) or a energy exhausting space (represented by negative integers) or neutral space (represented by 0). As X enters that space, X would either gain health or lose health. The first room and the last room would also impact Mr.X's energy levels.

To begin with Mr.X starts with some health points. X might lose or gain health points as X enters into different spaces in the maze.

Find the initial health points needed for Mr.X to be able to win the keys (K). Your function should return the health points value. The input maze M is provided as a 2-D array as shown in the example below.

Example:

-1 (X)	-2	2
10	-8	1
-5	-2	-3 (K)

The minimum health points needed are 2. X would follow the path

down->down->right->right

Sample Input: M: [[-1, -2, 2], [10, -8, 1], [-5, -2, -3]]; Output: 2

```
Debriefing (required!): --------------------------
```

Report:

1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. How deeply do you feel you understand the material it covers (0%–100%)?
4. Any other comments?

```
Note: ‘Debriefing’ section is intended to help us calibrate the assignments.
```