Nibila Amutha

1. Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where:

'.' Matches any single character.â€‹â€‹â€‹â€‹
'*' Matches zero or more of the preceding element.
The matching should cover the entire input string (not partial).

Example 1:

Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
Example 2:

Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".
Example 3:

Input: s = "ab", p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".

Constraints:

1 <= s.length <= 20
1 <= p.length <= 30
s contains only lowercase English letters.
p contains only lowercase English letters, '.', and '*'.
It is guaranteed for each appearance of the character '*', there will be a previous valid character to match.

==========

2. Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

 1   2   3
    abc def

```
 4   5   6
ghi jkl mno

 7    8   9
prqs tuv wxyz
```

Example 1:

```
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

Example 2:

```
Input: digits = ""
Output: []
```

Example 3:

```
Input: digits = "2"
Output: ["a","b","c"]
```

Constraints:

```
0 <= digits.length <= 4
digits[i] is a digit in the range ['2', '9']
```

==========

3. Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.

Return the sum of the three integers.

You may assume that each input would have exactly one solution.

Example 1:

```
Input: nums = [-1,2,1,-4], target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

Example 2:

```
Input: nums = [0,0,0], target = 1
```

```
Output: 0

Constraints:

3 <= nums.length <= 1000
-1000 <= nums[i] <= 1000
-104 <= target <= 104

==========================================================
```

4. You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise).

You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.

Example 1:

```
1 2 3      7 4 1
4 5 6 ==== 8 5 2
7 8 9      9 6 3

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]
Example 2:
 5  1  9 11      15 13  2  5
 2  4  8 10      14  3  4  1
13  3  6  7 === 12  6  8  9
15 14 12 16      16  7 10 11

Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]
Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]


Constraints:

n == matrix.length == matrix[i].length
1 <= n <= 20
-1000 <= matrix[i][j] <= 1000
==========================================================
```