Tutorial 4: Stacks

ELEC 278: Fundamentals of Information Structures

The learning goals for Tutorial 4 are:

- Practice stacks and see stacks in-application by building a string decoder.

Problem 1. Replicated from `https://leetcode.com/problems/decode-string/`

Given an encoded string, return its decoded string.

The encoding rule is: `k[encoded_string]`, where the `encoded_string` inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4].

The test cases are generated so that the length of the output will never exceed 10^3.

Example 1:
Input: s = "3[a]2[bc]"
Output: "aaabcbc"

Example 2:
Input: s = "3[a2[c]]"
Output: "accaccacc"

Example 3:
Input: s = "2[abc]3[cd]ef"
Output: "abcabccdcdcdef"

Constraints:

- 1 <= s.length <= 30
- s consists of lowercase English letters, digits, and square brackets '[]'.
- s is guaranteed to be a valid input.
- All the integers in s are in the range [1, 300].

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

// Function to decode the string
char* decodeString(char* s) {
    // to do: complete
}

int main() {
    char s1[] = "3[a]2[bc]";
    char s2[] = "3[a2[c]]";
    char s3[] = "2[abc]3[cd]ef";

    char* s = decodeString(s1);
    printf("Decoded string [%s]: %s\n", s1, s);
    free(s);

    s = decodeString(s2);
    printf("Decoded string [%s]: %s\n", s2, s);
    free(s);

    s = decodeString(s3);
    printf("Decoded string [%s]: %s\n", s3, s);
    free(s);

    return 0;
}
```