# VSEC TEST

1.The Fibonacci numbers, commonly denoted **F(n)** form a sequence, called Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from **0** and **1**. That is,

F(0) = 0, F(1) = 1

F(n) = F(n-1) + F(n-2), for n > 1

Given n, calculate F(n).

*Example 1:*

**Input:** n = 2

**Output:** 1

**Explanation:** F(2) = F(1) + F(0) = 1 + 0 = 1.

*Example 2:*

**Input:** n = 3

**Output:** 2

**Explanation:** F(3) = F(2) + F(1) = 1 + 1 = 2.

*Example 3:*

**Input:** n = 4

**Output:** 3

**Explanation:** F(4) = F(3) + F(2) = 2 + 1 = 3.

Constraints:

0 <= n <= 30

Test Cases

1: 5 → 5

2: 12 → 144

3: 17 → 1597

2.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 **10^5**.

*Example 1:*

**Input:** s = "3[a]2[bc]"

**Output:** "aaabcbc"

*Example 2:*

**Input:** s = "3[a2[c]]"

**Output:** "accaccacc

*Example :*

**Input:** s = "2[abc]3[cd]ef"

**Output:** "abcabccdcdcdef"

3.Given a string s representing a valid expression, implement a basic calculator to evaluate it, and return the result of the evaluation.

Note: You are not allowed to ues any built-in function which evaluates strings as mathematical expressions, such as eval().

*Example 1:*

**Input:** s = "1 + 1"

**Output:** 2

*Example 2:*

**Input:** s = "2-1 + 2"

**Output:** 3

*Example 3:*

**Input:** s = "(1+(4+5+2)-3)+(6+8)"

**Output:** 23

Constraints:

1 <= s.length <= 3 * 10^5

S consists of digit, "+", "-", "(", ")", and " "

S represents a valid expression

"+" is not used as any unary operation ("-1" and "-(2 + 3)" is valid)

There will be no two consecutive operators in the input

Every number and running calculation will fit a signed 32-bit integer