class Solution:

def decodeString(self, s: str) -> str:

i = 0 # Pointer to iterate over the string

open\_, close = 0, 0 # Counters for matching brackets

ans = "" # Final decoded answer

while i < len(s):

# If it's a letter (not digit or bracket), just add to answer

if s[i] not in "0123456789[]":

ans += s[i]

# If we encounter a digit, we need to process the pattern k[...]

elif s[i] in "0123456789":

num = "" # To collect multi-digit numbers, like '12'

while s[i] in "0123456789":

num += s[i]

i += 1

# After the number, we expect a '[' — start index of substring

start = i + 1 # Skip the '['

i += 1

open\_ += 1 # We just saw an opening bracket '['

# Now find the corresponding closing bracket ']'

while i < len(s) and open\_ != close:

if s[i] == "[":

open\_ += 1

elif s[i] == "]":

close += 1

i += 1

# Recursively decode the substring inside the brackets

temp = self.decodeString(s[start: i - 1])

# Repeat the decoded substring num times and add to answer

for \_ in range(int(num)):

ans += temp

# Step back one index because the outer loop also increments i

i -= 1

# Move to next character

i += 1

return ans