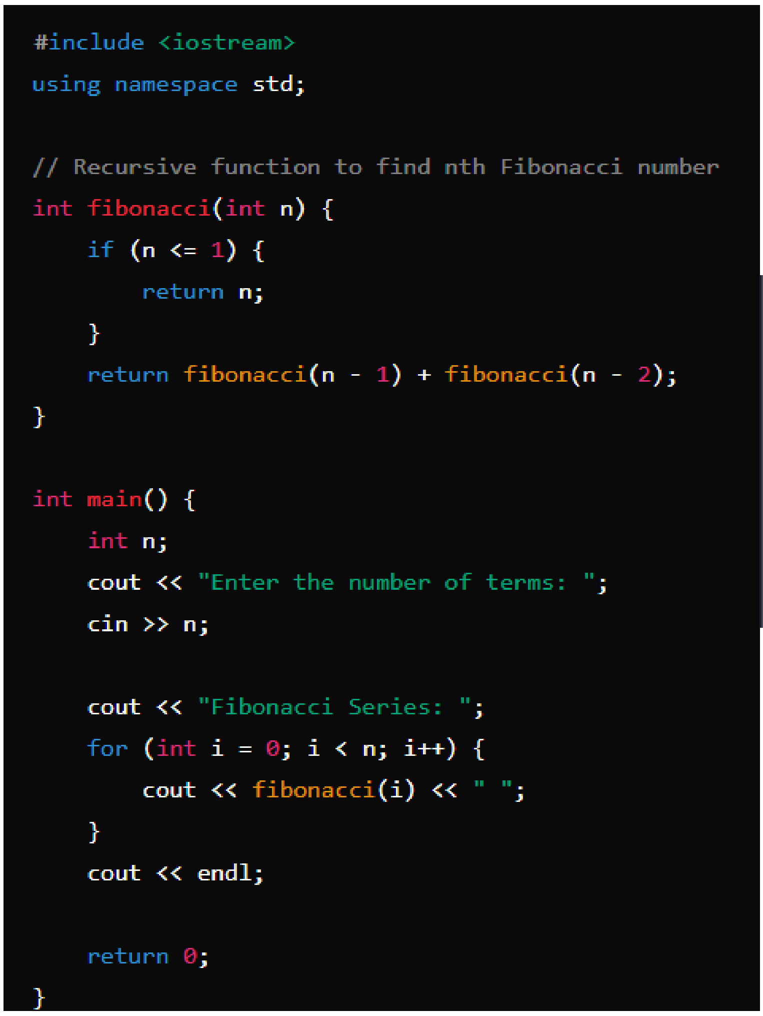
1. Two Sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

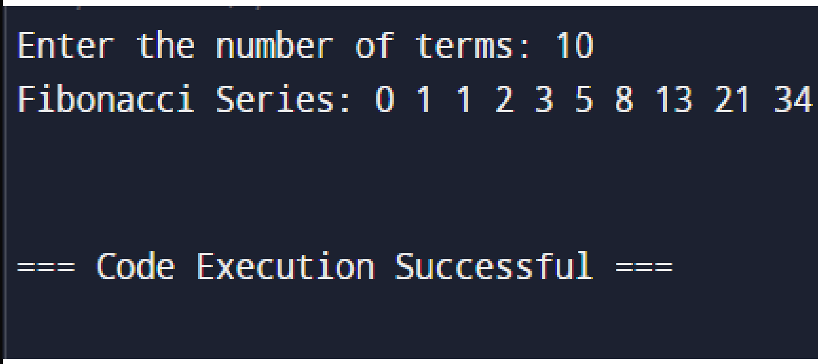
Example 1: Input: nums = [2,7,11,15], target = 9 Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1]

Program:



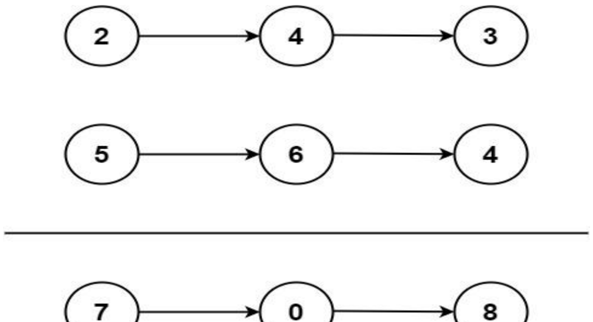
```cpp
#include <iostream>
using namespace std;

// Recursive function to find nth Fibonacci number
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n;
    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Fibonacci Series: ";
    for (int i = 0; i < n; i++) {
        cout << fibonacci(i) << " ";
    }
    cout << endl;

    return 0;
}
```



```
Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34


=== Code Execution Successful ===
```

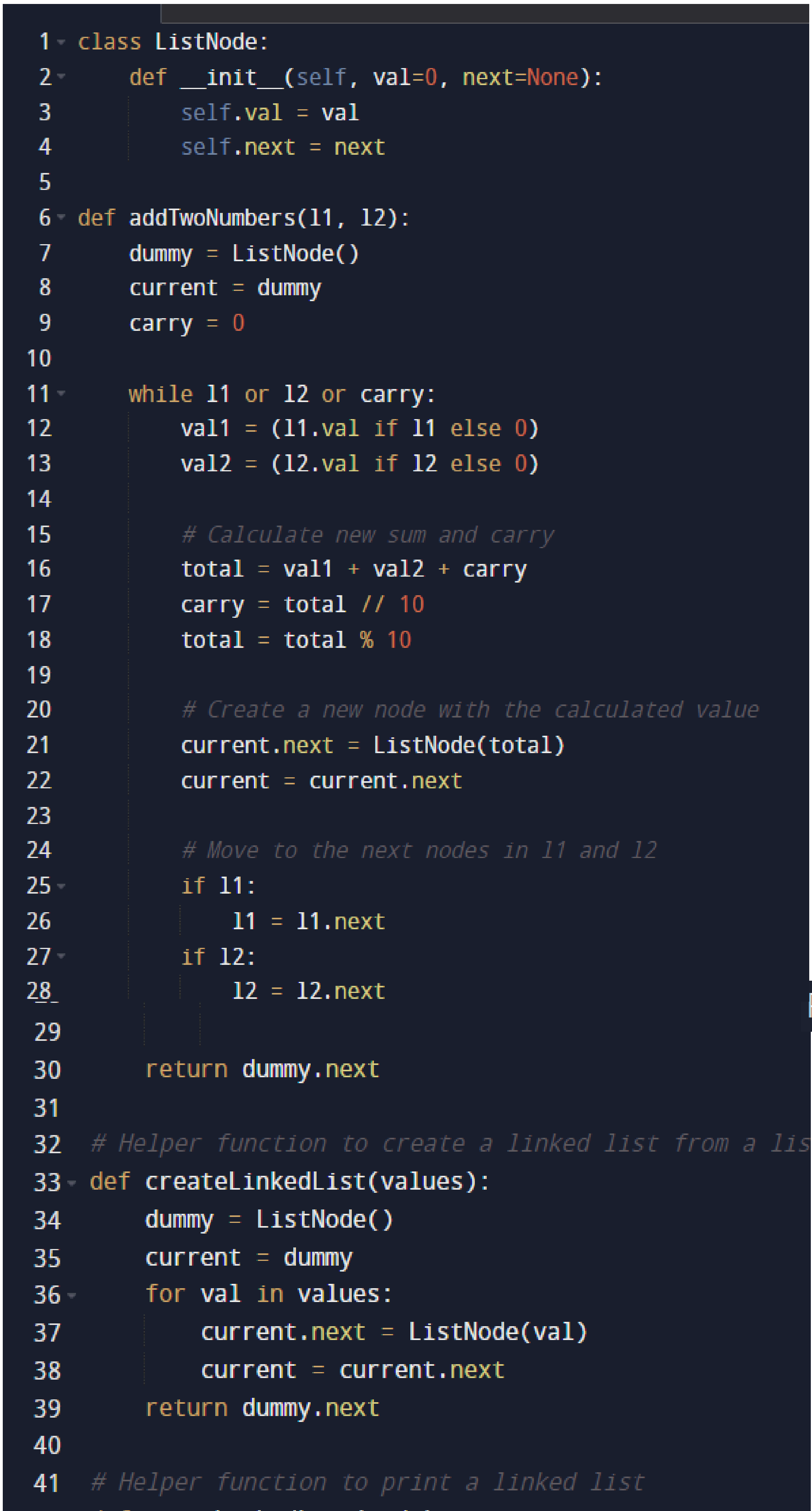
2. Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list. You may assume the two numbers do not contain any leading zero, except the number 0 itself.
Example 1:

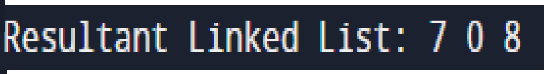


Input: l1 = [2,4,3], l2 = [5,6,4] Output: [7,0,8]
Explanation: 342 + 465 = 807.

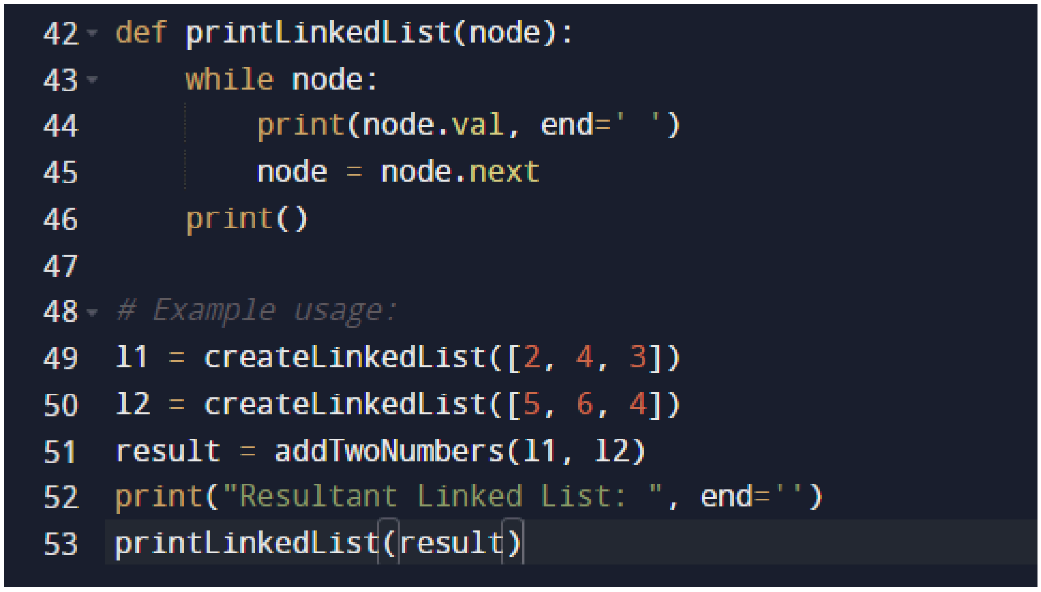
Program:



```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def addTwoNumbers(l1, l2):
    dummy = ListNode()
    current = dummy
    carry = 0

    while l1 or l2 or carry:
        val1 = (l1.val if l1 else 0)
        val2 = (l2.val if l2 else 0)

        # Calculate new sum and carry
        total = val1 + val2 + carry
        carry = total // 10
        total = total % 10

        # Create a new node with the calculated value
        current.next = ListNode(total)
        current = current.next

        # Move to the next nodes in l1 and l2
        if l1:
            l1 = l1.next
        if l2:
            l2 = l2.next

    return dummy.next

# Helper function to create a linked list from a lis
def createLinkedList(values):
    dummy = ListNode()
    current = dummy
    for val in values:
        current.next = ListNode(val)
        current = current.next
    return dummy.next

# Helper function to print a linked list
```



```
Resultant Linked List: 7 0 8
```



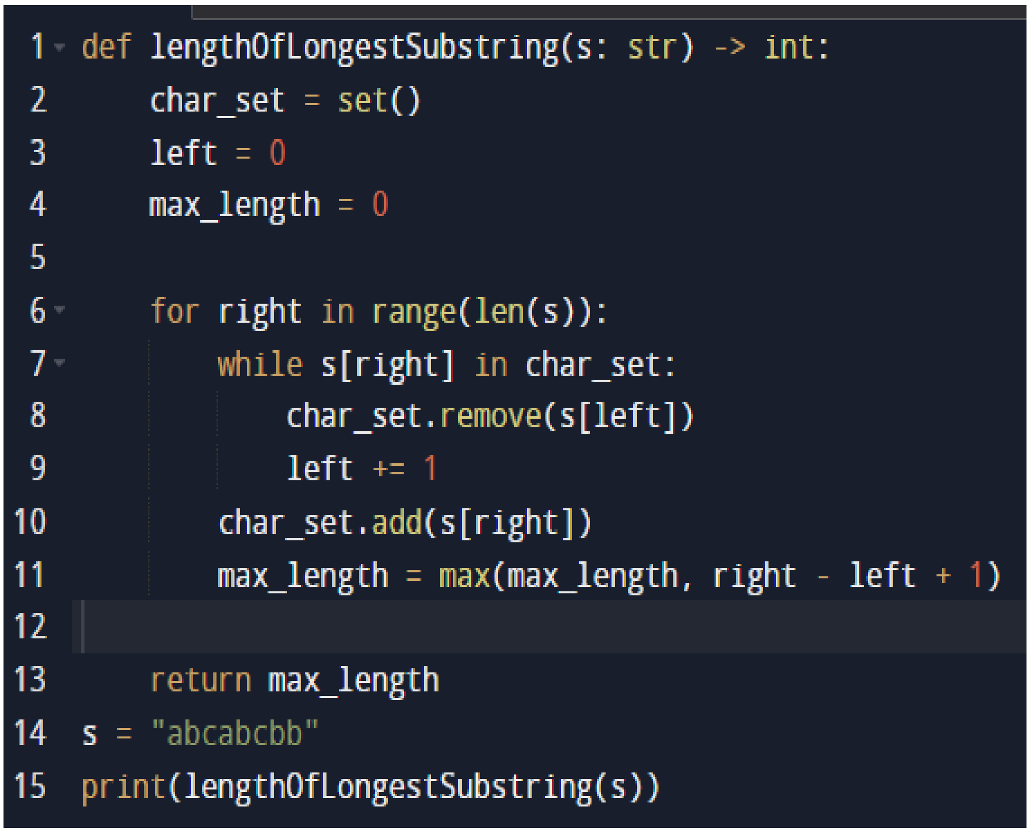
```python
def printLinkedList(node):
    while node:
        print(node.val, end=' ')
        node = node.next
    print()

# Example usage:
l1 = createLinkedList([2, 4, 3])
l2 = createLinkedList([5, 6, 4])
result = addTwoNumbers(l1, l2)
print("Resultant Linked List: ", end='')
printLinkedList(result)
```

3. Longest Substring without Repeating Characters

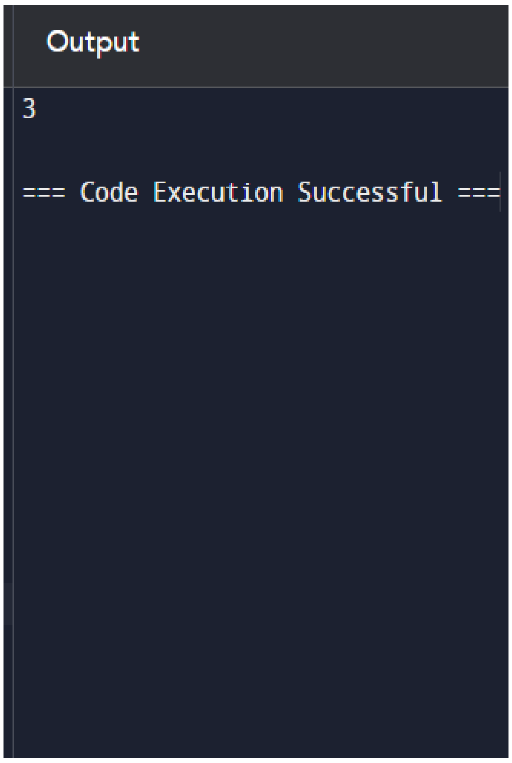
Given a string s, find the length of the longest substring without repeating characters. Example 1: Input: s = "abcabcbb" Output: 3

Explanation: The answer is "abc", with the length of 3.

Program:



```python
def lengthOfLongestSubstring(s: str) -> int:
    char_set = set()
    left = 0
    max_length = 0

    for right in range(len(s)):
        while s[right] in char_set:
            char_set.remove(s[left])
            left += 1
        char_set.add(s[right])
        max_length = max(max_length, right - left + 1)

    return max_length
s = "abcabcbb"
print(lengthOfLongestSubstring(s))
```



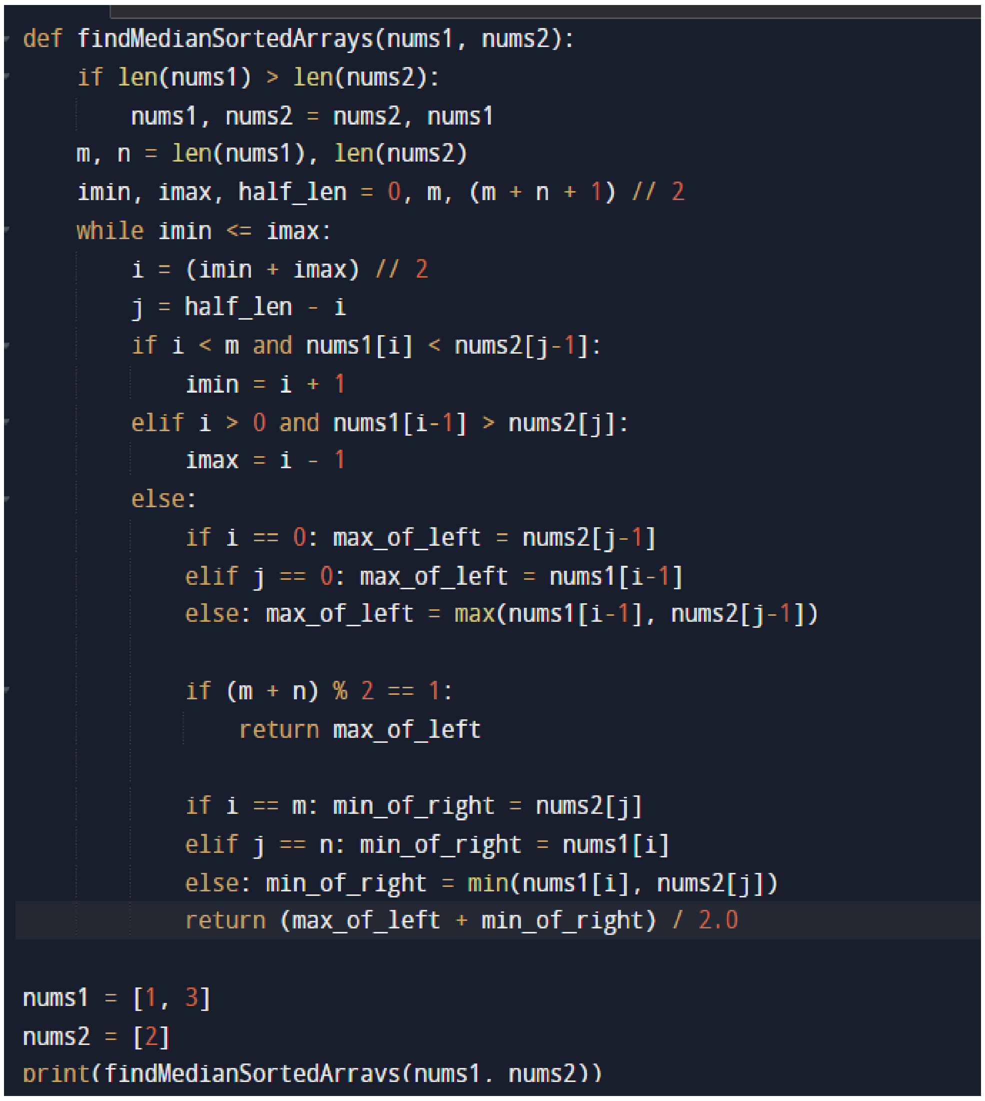
Output

```
3

=== Code Execution Successful ===
```

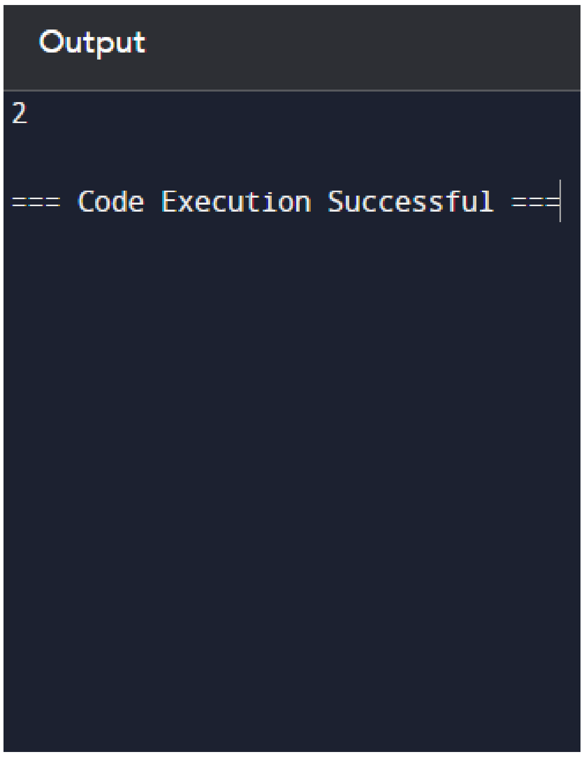
4. Median of Two Sorted Arrays

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).
Example 1: Input: nums1 = [1,3], nums2 = [2] Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.

Program:



```python
def findMedianSortedArrays(nums1, nums2):
    if len(nums1) > len(nums2):
        nums1, nums2 = nums2, nums1
    m, n = len(nums1), len(nums2)
    imin, imax, half_len = 0, m, (m + n + 1) // 2
    while imin <= imax:
        i = (imin + imax) // 2
        j = half_len - i
        if i < m and nums1[i] < nums2[j-1]:
            imin = i + 1
        elif i > 0 and nums1[i-1] > nums2[j]:
            imax = i - 1
        else:
            if i == 0: max_of_left = nums2[j-1]
            elif j == 0: max_of_left = nums1[i-1]
            else: max_of_left = max(nums1[i-1], nums2[j-1])

            if (m + n) % 2 == 1:
                return max_of_left

            if i == m: min_of_right = nums2[j]
            elif j == n: min_of_right = nums1[i]
            else: min_of_right = min(nums1[i], nums2[j])
            return (max_of_left + min_of_right) / 2.0

nums1 = [1, 3]
nums2 = [2]
print(findMedianSortedArrays(nums1, nums2))
```



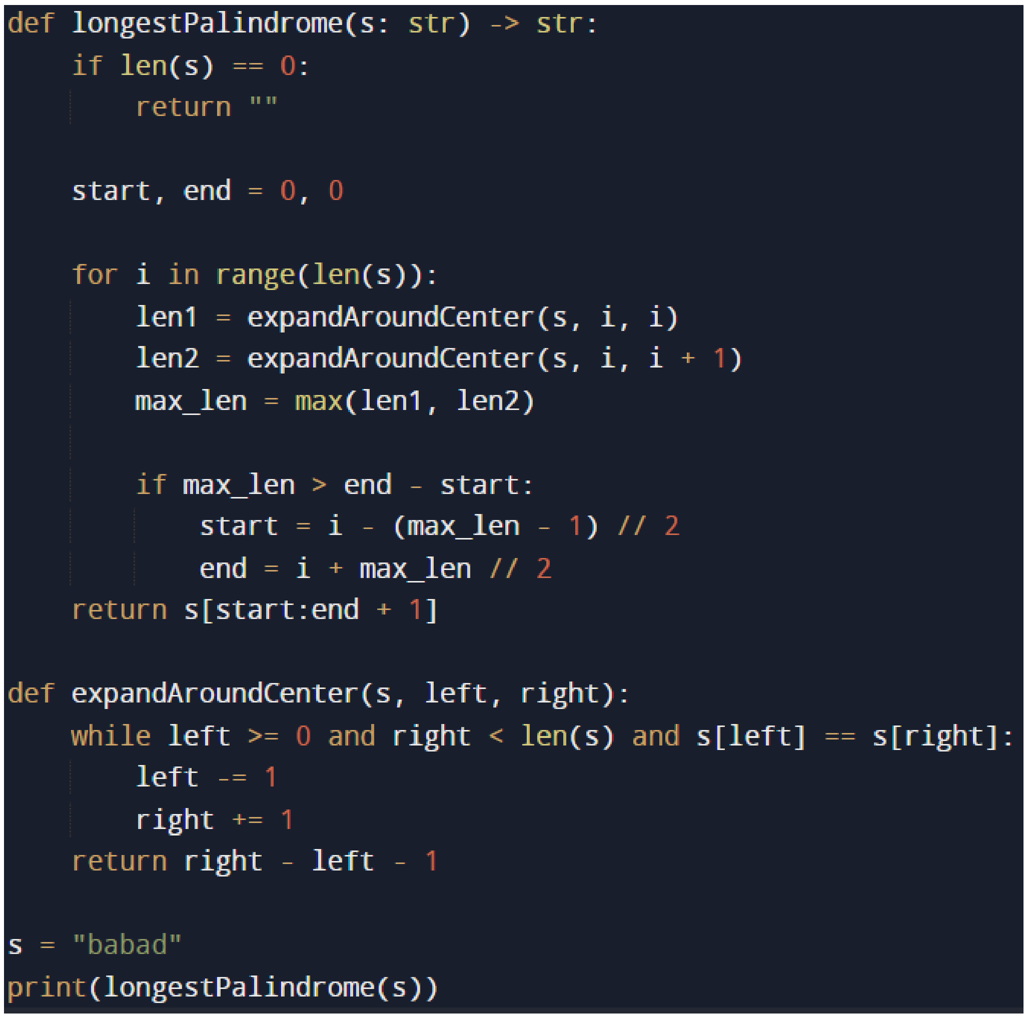
Output

```
2

=== Code Execution Successful ===
```

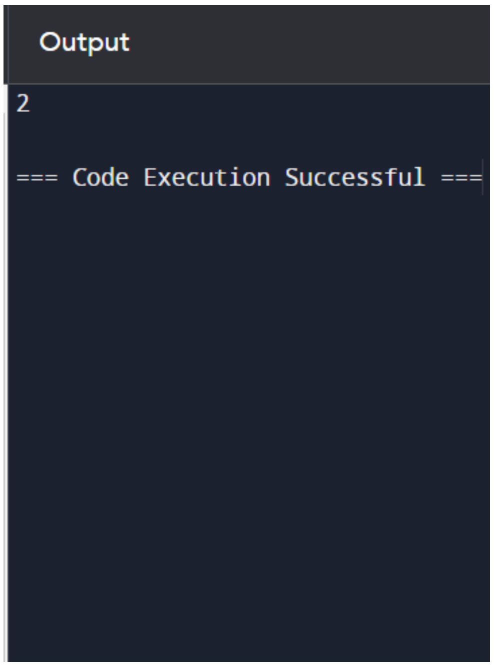
5. Longest Palindromic Substring

Given a string s, return the longest palindromic substring in s.
Example 1: Input: s = "babad" Output: "bab"
Explanation: "aba" is also a valid answer.

Program:



```python
def longestPalindrome(s: str) -> str:
    if len(s) == 0:
        return ""

    start, end = 0, 0

    for i in range(len(s)):
        len1 = expandAroundCenter(s, i, i)
        len2 = expandAroundCenter(s, i, i + 1)
        max_len = max(len1, len2)

        if max_len > end - start:
            start = i - (max_len - 1) // 2
            end = i + max_len // 2
    return s[start:end + 1]

def expandAroundCenter(s, left, right):
    while left >= 0 and right < len(s) and s[left] == s[right]:
        left -= 1
        right += 1
    return right - left - 1

s = "babad"
print(longestPalindrome(s))
```



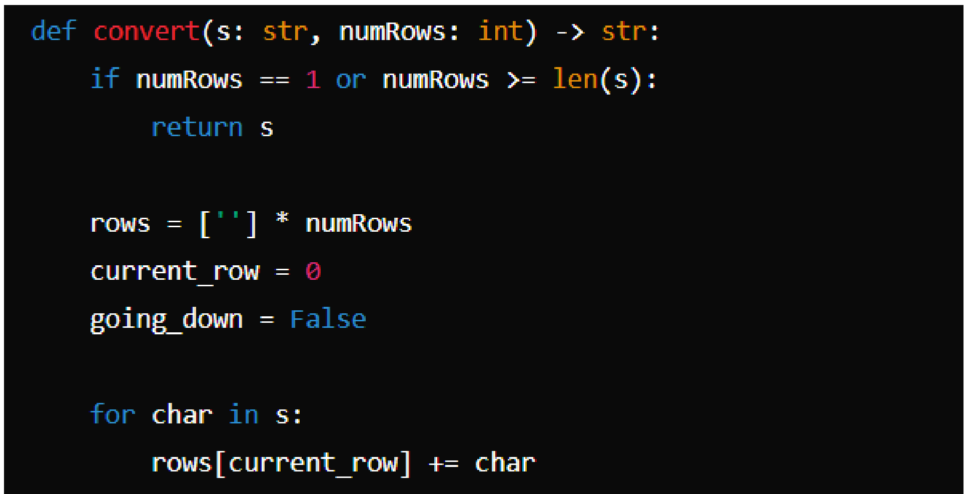
Output

```
2

=== Code Execution Successful ===
```

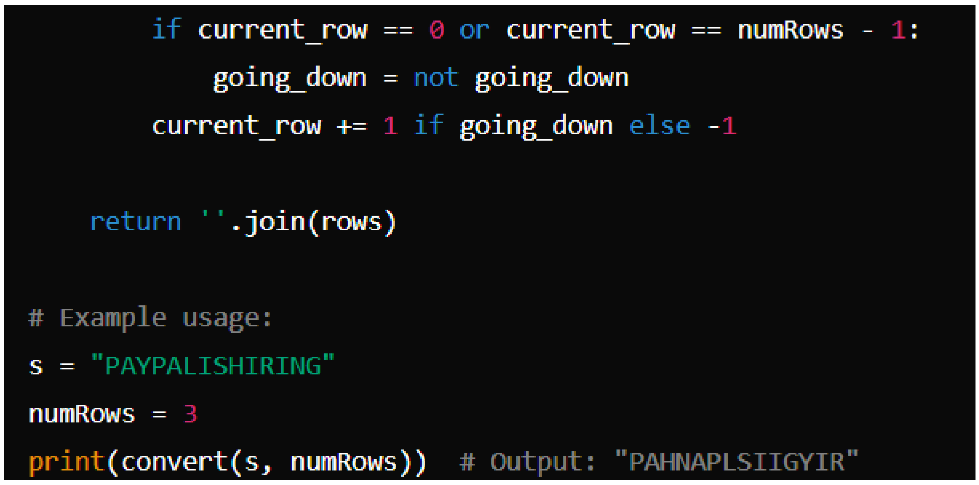
6. Zigzag Conversion

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility) P A H N A P L S I I G Y I R And then read line by line: "PAHNAPLSIIGYIR" Write the code that will take a string and make this conversion given a number of rows: string convert(string s, int numRows); Example 1: Input: s = "PAYPALISHIRING", numRows = 3 Output: "PAHNAPLSIIGYIR"

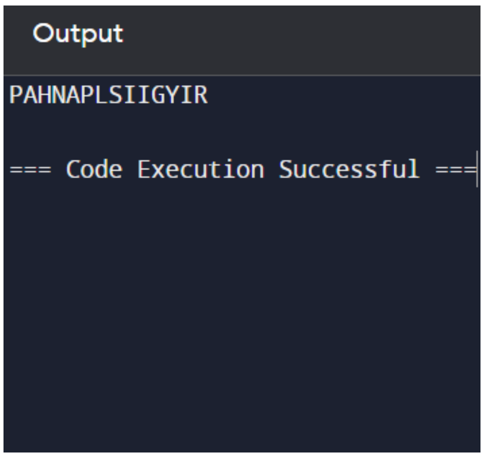
Program:



```python
def convert(s: str, numRows: int) -> str:
    if numRows == 1 or numRows >= len(s):
        return s

    rows = [''] * numRows
    current_row = 0
    going_down = False

    for char in s:
        rows[current_row] += char
```



```
        if current_row == 0 or current_row == numRows - 1:
            going_down = not going_down
        current_row += 1 if going_down else -1

    return ''.join(rows)

# Example usage:
s = "PAYPALISHIRING"
numRows = 3
print(convert(s, numRows))  # Output: "PAHNAPLSIIGYIR"
```



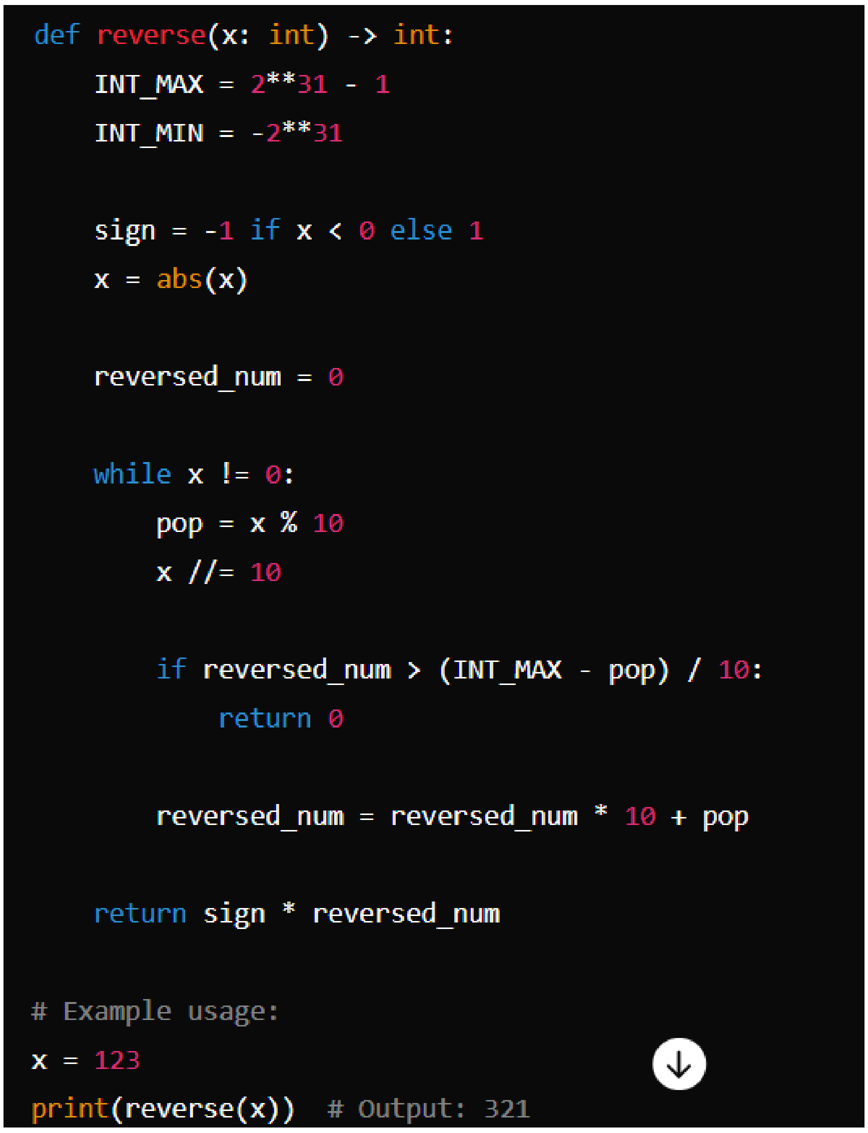
```
Output
PAHNAPLSIIGYIR

=== Code Execution Successful ===
```

7. Reverse Integer

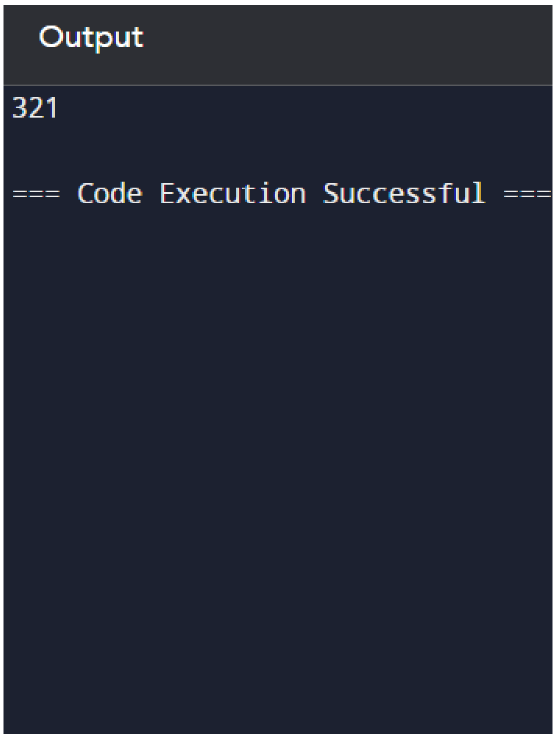
Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0. Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1: Input: x = 123 Output: 321

Program:



```
def reverse(x: int) -> int:
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31

    sign = -1 if x < 0 else 1
    x = abs(x)

    reversed_num = 0

    while x != 0:
        pop = x % 10
        x //= 10

        if reversed_num > (INT_MAX - pop) / 10:
            return 0

        reversed_num = reversed_num * 10 + pop

    return sign * reversed_num

# Example usage:
x = 123
print(reverse(x))  # Output: 321
```

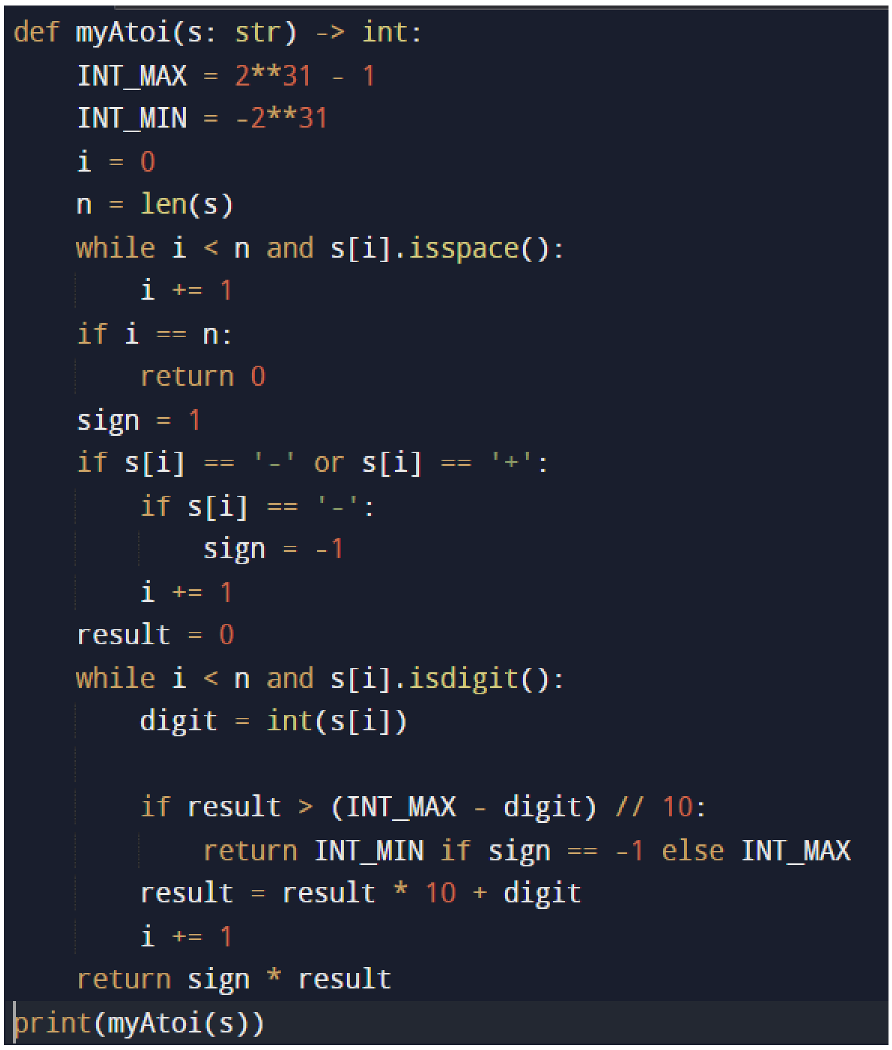


```
Output
321

=== Code Execution Successful ===
```

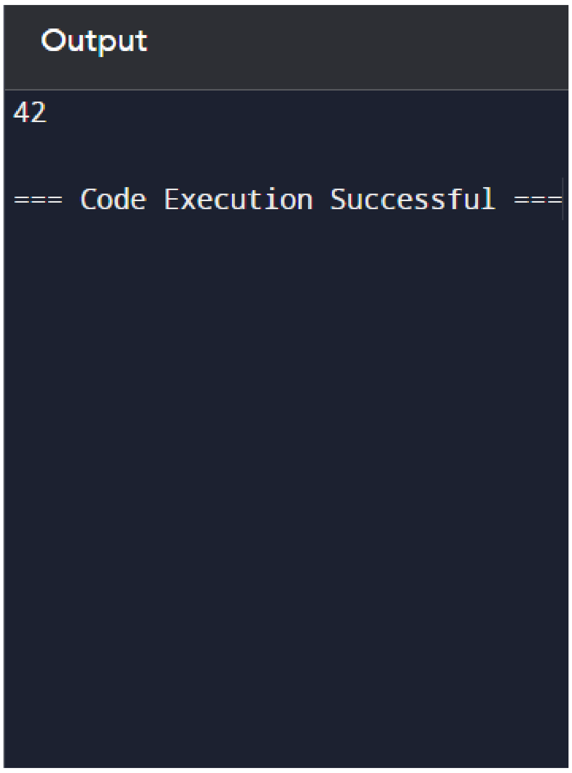
8. String to Integer

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function). The algorithm for myAtoi(string s) is as follows.
Example 1:
Input: s = "42"
Output: 42
Explanation: The underlined characters are what is read in, the caret is the current reader position. Step 1: "42" (no characters read because there is no leading whitespace) ^ Step 2: "42" (no characters read because there is neither a '-' nor '+') ^ Step 3: "42" ("42" is read in) ^ The parsed integer is 42. Since 42 is in the range [-231, 231 - 1], the final result is 42.

Program:



```python
def myAtoi(s: str) -> int:
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    i = 0
    n = len(s)
    while i < n and s[i].isspace():
        i += 1
    if i == n:
        return 0
    sign = 1
    if s[i] == '-' or s[i] == '+':
        if s[i] == '-':
            sign = -1
        i += 1
    result = 0
    while i < n and s[i].isdigit():
        digit = int(s[i])

        if result > (INT_MAX - digit) // 10:
            return INT_MIN if sign == -1 else INT_MAX
        result = result * 10 + digit
        i += 1
    return sign * result
print(myAtoi(s))
```



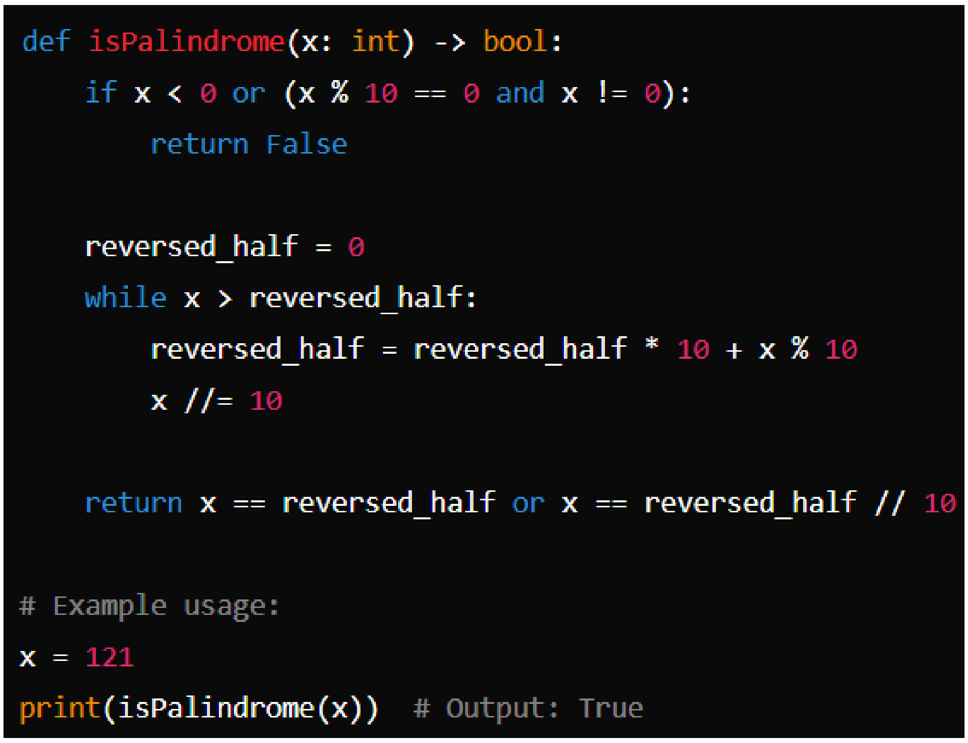
Output

```
42

=== Code Execution Successful ===
```

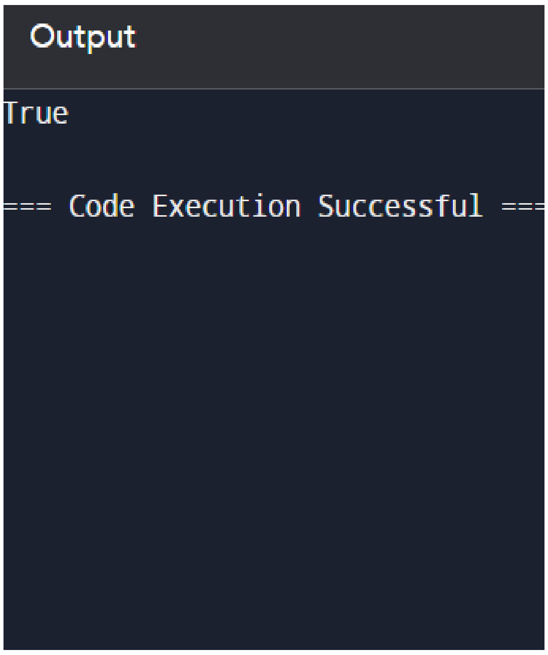
9. Palindrome Number

Given an integer x, return true if x is a palindrome, and false otherwise.
Example 1: Input: x = 121 Output: true
Explanation: 121 reads as 121 from left to right and from right to left.

Program:



```python
def isPalindrome(x: int) -> bool:
    if x < 0 or (x % 10 == 0 and x != 0):
        return False

    reversed_half = 0
    while x > reversed_half:
        reversed_half = reversed_half * 10 + x % 10
        x //= 10

    return x == reversed_half or x == reversed_half // 10

# Example usage:
x = 121
print(isPalindrome(x))  # Output: True
```

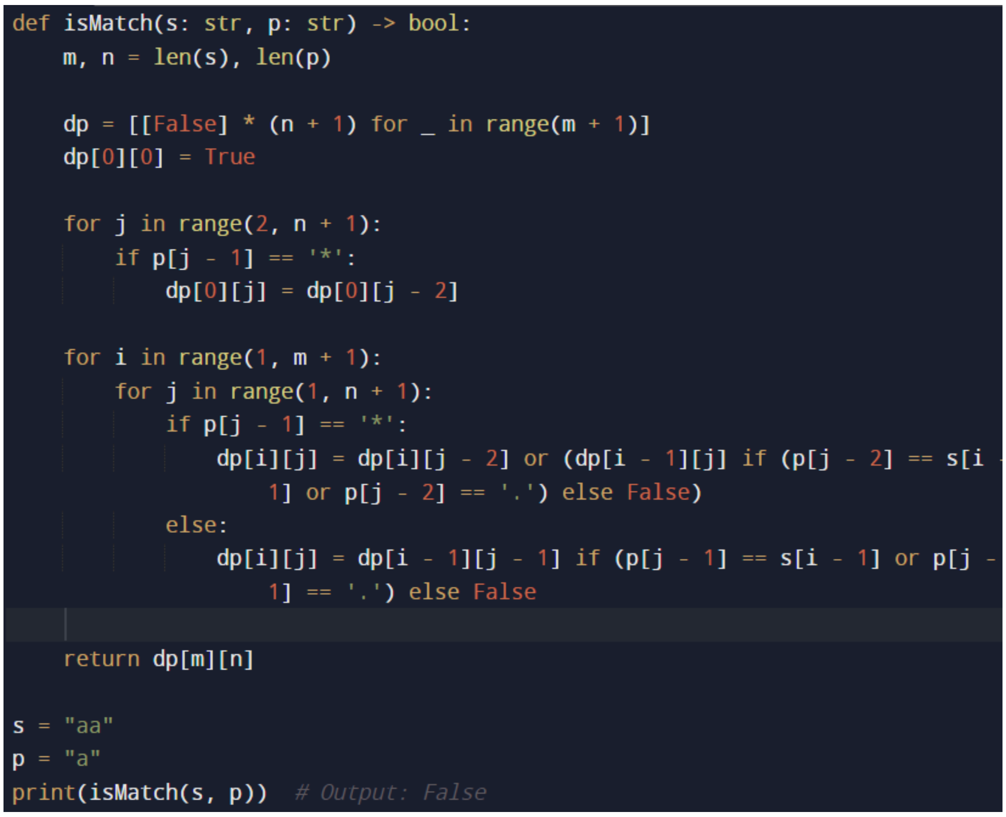


```
Output

True

=== Code Execution Successful ===
```

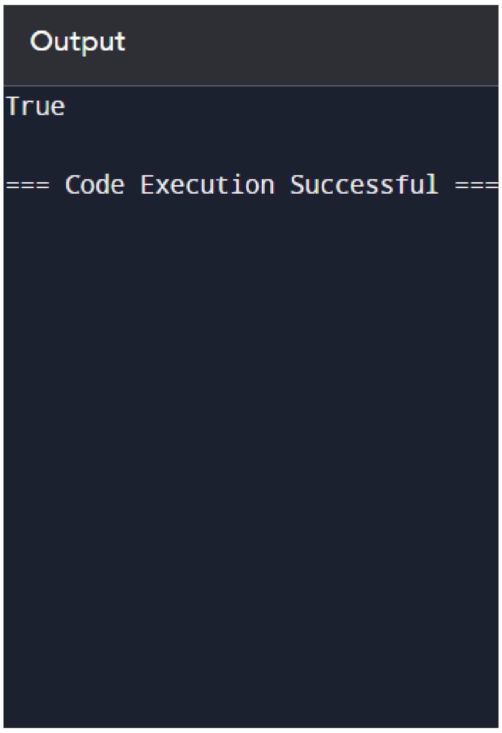
10. Regular Expression Matching

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where: • '.' Matches any single character. • '*' 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".

Program:



```python
def isMatch(s: str, p: str) -> bool:
    m, n = len(s), len(p)

    dp = [[False] * (n + 1) for _ in range(m + 1)]
    dp[0][0] = True

    for j in range(2, n + 1):
        if p[j - 1] == '*':
            dp[0][j] = dp[0][j - 2]

    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] if (p[j - 2] == s[i -
                    1] or p[j - 2] == '.') else False)
            else:
                dp[i][j] = dp[i - 1][j - 1] if (p[j - 1] == s[i - 1] or p[j -
                    1] == '.') else False

    return dp[m][n]

s = "aa"
p = "a"
print(isMatch(s, p))  # Output: False
```



```
Output

True

=== Code Execution Successful ===
```