NAME : KOVI SAI GANESH

REGISTER NUMBER : 192124028

DAY 3 PROGRAMS

1Q In daily share trading, a buyer buys shares in the morning and sells them on the same day. If the trader is allowed to make at most 2 transactions in a day, whereas the second transaction can only start after the first one is complete (Buy->sell->Buy->sell). Given stock prices throughout the day, find out the maximum profit that a share trader could have made. Test Case: 1.Input: prices = [7,1,5,3,6,4] Output: 7 2.Input: prices = [7,6,4,3,1] Output: 0 3.Input: [10, 22, 5, 75, 65, 80] Output:87 4.Input: [2, 30, 15, 10, 8, 25, 80] Output:100 5.Input: [10, 22, 5, 75, 65, 80] Output:0

def maxProfit(price, n):

profit = [0]\*n

max\_price = price[n-1]

for i in range(n-2, 0, -1):

if price[i] > max\_price:

max\_price = price[i]

profit[i] = max(profit[i+1], max\_price - price[i])

min\_price = price[0]

for i in range(1, n):

if price[i] < min\_price:

min\_price = price[i]

profit[i] = max(profit[i-1], profit[i]+(price[i]-min\_price))

result = profit[n-1]

return result

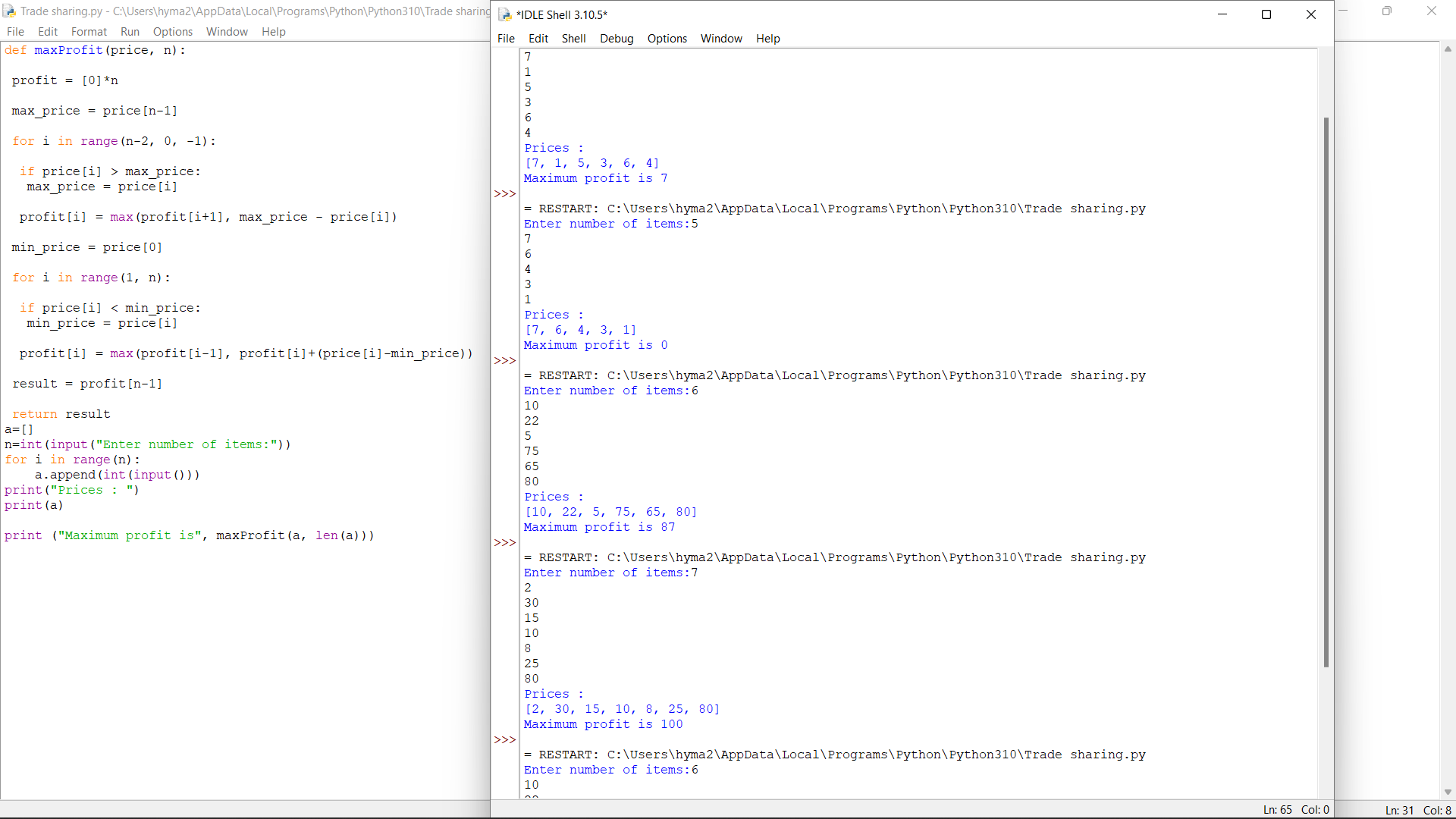
a=[]

n=int(input("Enter number of items:"))

for i in range(n):

a.append(int(input()))

print ("Maximum profit is", maxProfit(a, len(a)))



2Q The Project manager has to submit the project within a week period. He didn’t find the proper combinations of team members to work on the project, Help him in finding the possible combinations available. Accept 3 digits and find all the combinations Sample Input: 123 Sample Output: 123 132 213 231 312 321 Test Cases: 1. 789 2. 1456 3.-856 4. 1001 5. 555

def comb(L):

for i in range(3):

for j in range(3):

for k in range(3):

# check if the indexes are not

# same

if (i!=j and j!=k and i!=k):

print(L[i], L[j], L[k])

a=[]

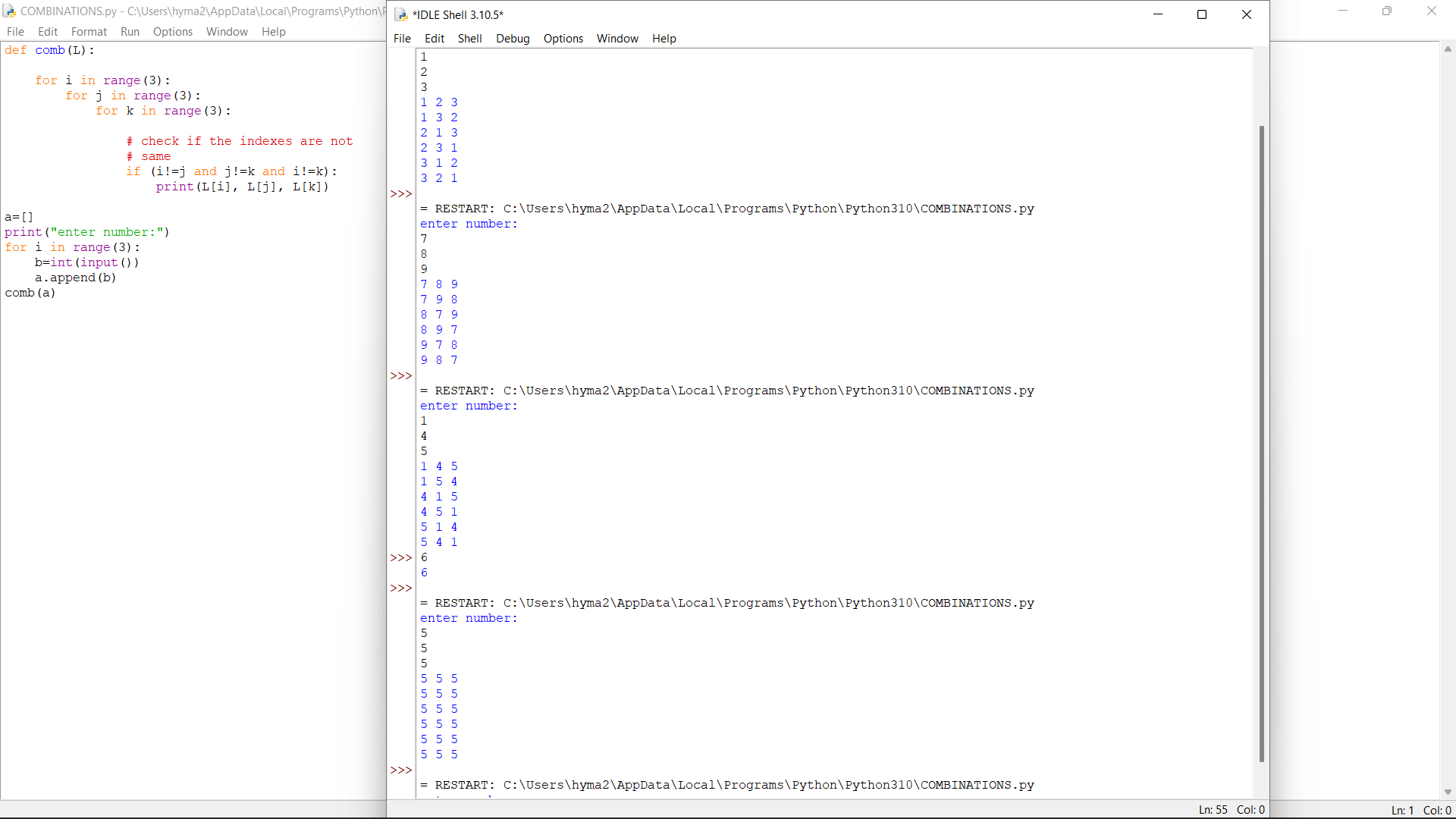
print("enter number:")

for i in range(3):

b=int(input())

a.append(b)

comb(a)



3Q Given an array of integers nums, return the number of good pairs. A pair (i, j) is called good if nums[i] == nums[j] and i < j.

def solve(nums):

count=0

n=len(nums)

for i in range(n):

for j in range(i+1,n):

if nums[i] == nums[j]:

count+=1

return count

a=[]

n=int(input("Enter number of elements:"))

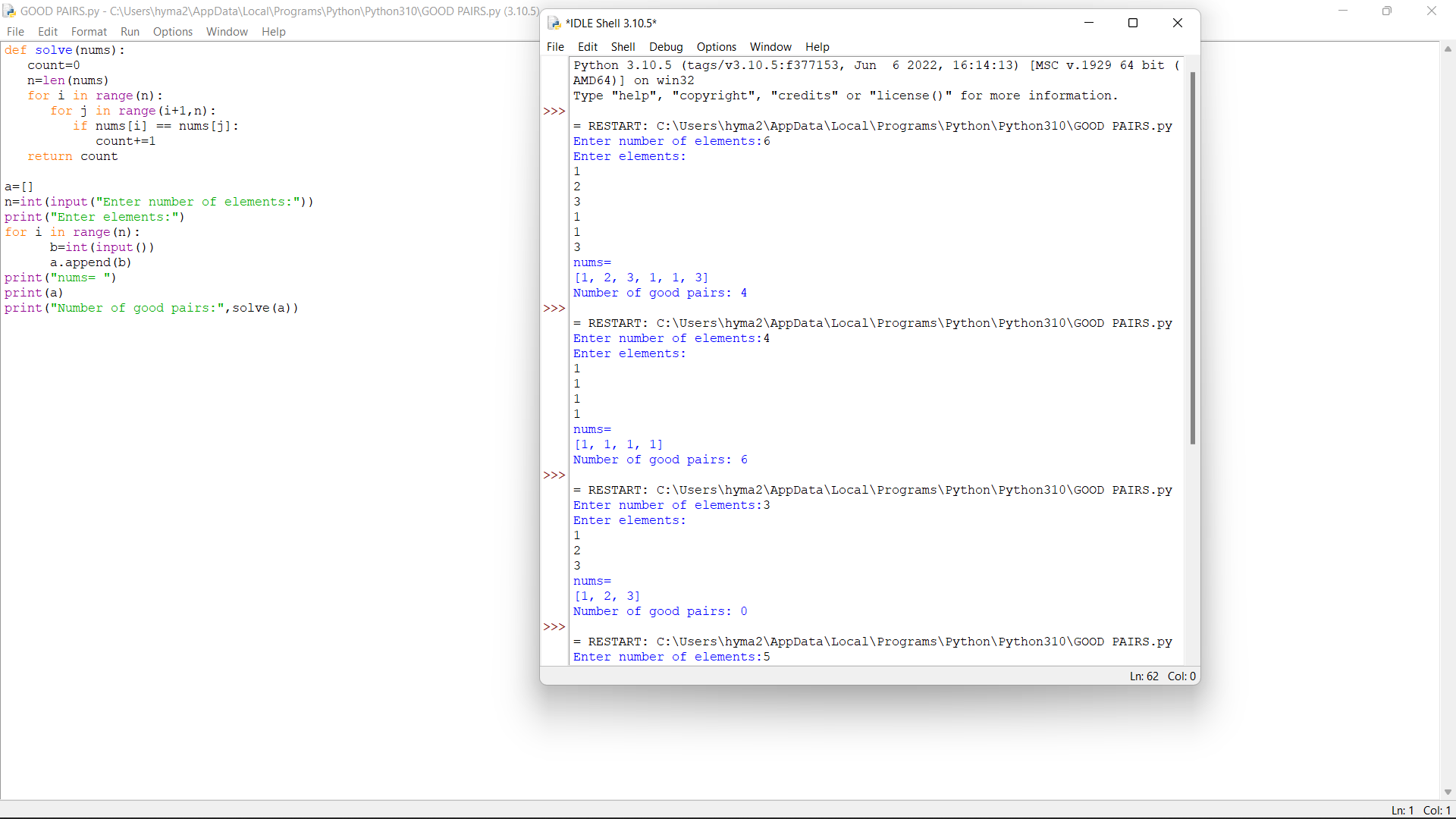
print("Enter elements:")

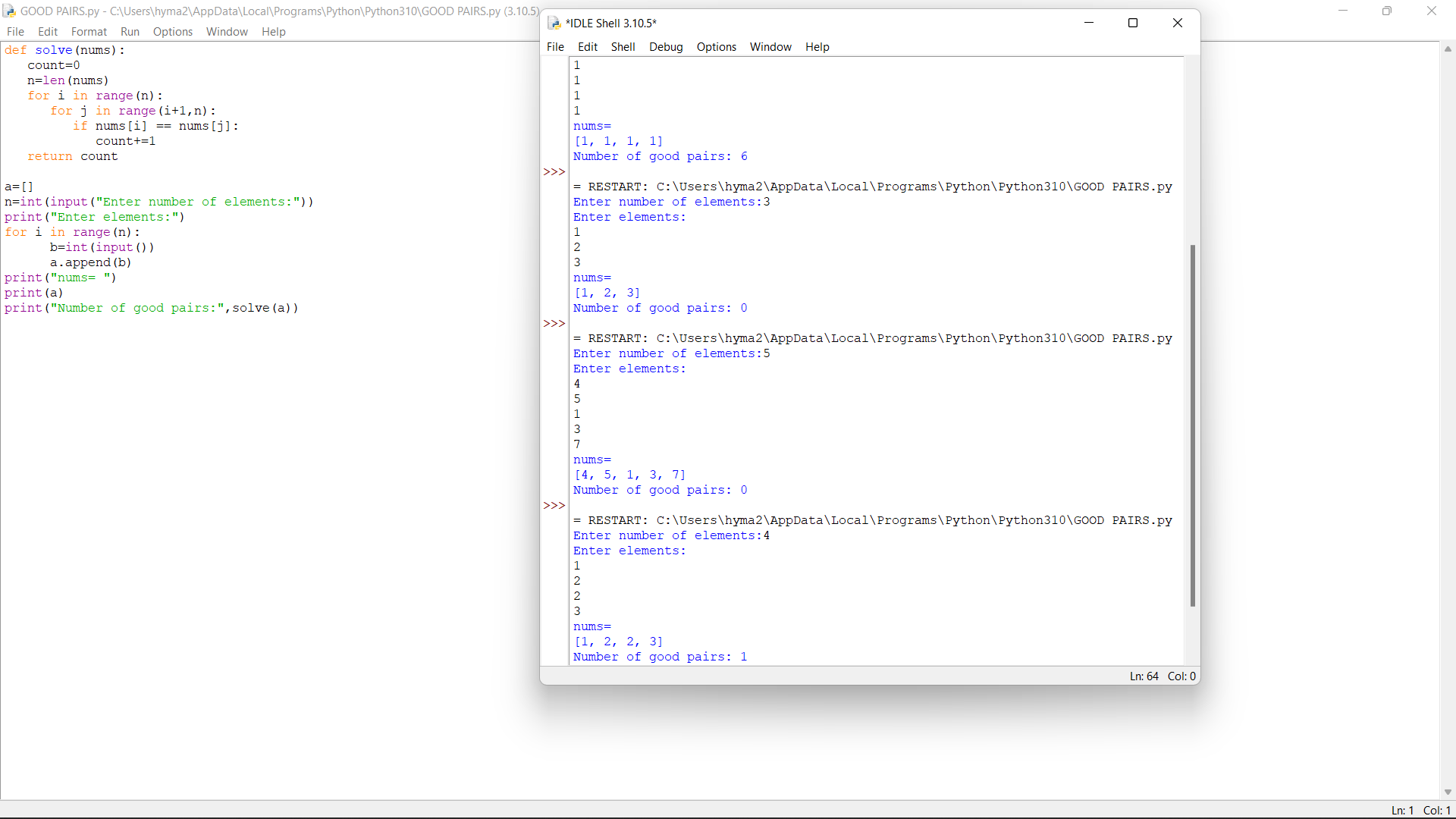
for i in range(n):

b=int(input())

a.append(b)

print("Number of good pairs:",solve(a))





6Q Raju, has again started troubling people in your city. The people have turned on to you for getting rid of Raju. Raju presents to you a number consisting of numbers from 0 to 9 characters. He wants you to reverse it from the final answer such that the number becomes Mirror number. A Mirror is a number which equals its reverse. The hope of people are on you so you have to solve the riddle. You have to tell if some number exists which you would reverse to convert the number into Mirror Sample input: Enter the number: 123456 Sample output: Mirror image: 654321

def reverse(s):

str = " "

for i in s:

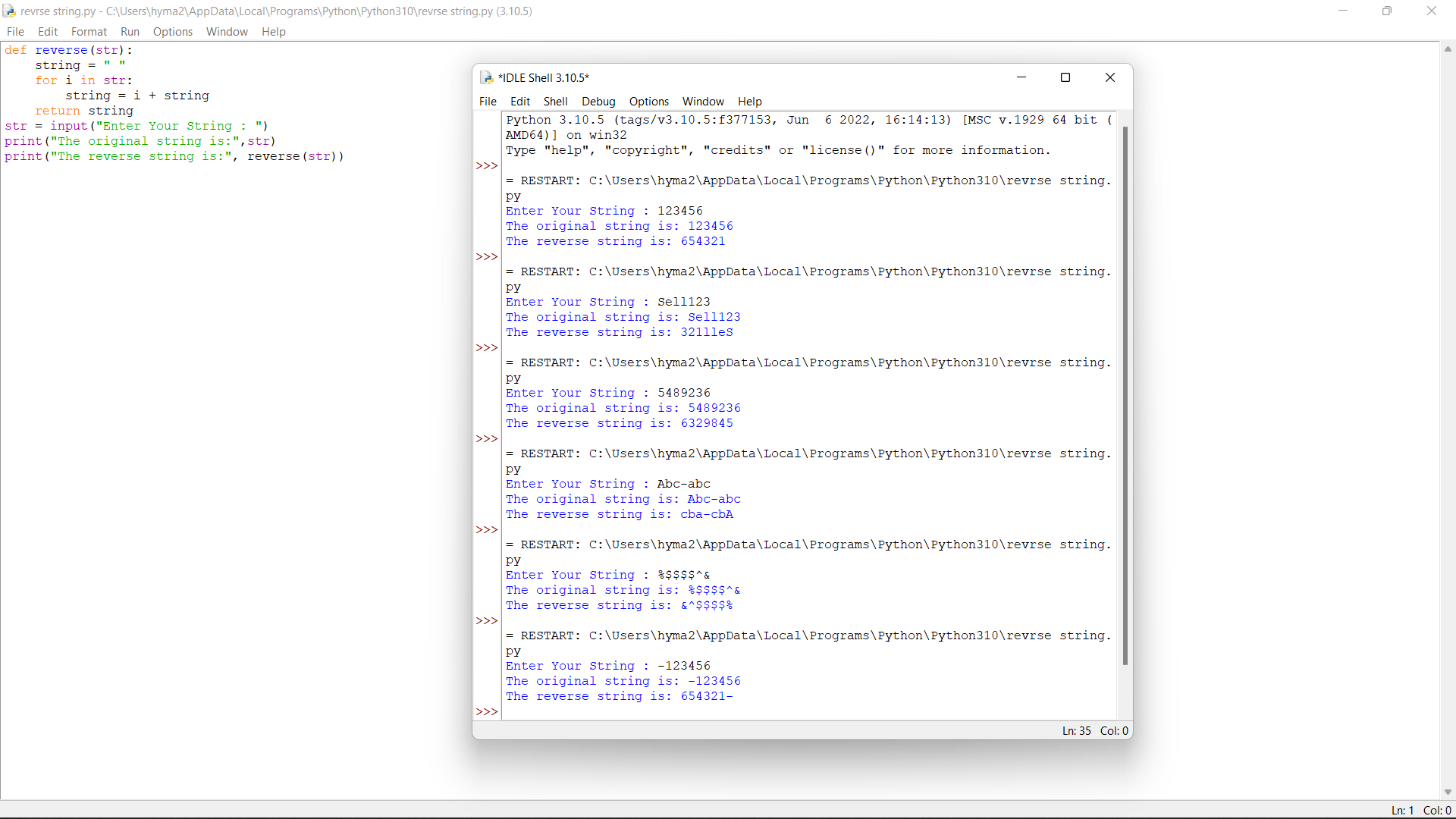
str= i+str

return str

s=input("Enter Your String : ")

print("Your String is : ", s)

print("Your Reverse String is : ", reverse(s))



10Q Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2. You have the following three operations permitted on a word: • Insert a character • Delete a character • Replace a character Test case: 1.Input: word1 = "horse", word2 = "ros" Output: 3 2.Input: word1 = "intention", word2 = "execution" Output: 5 3.Input: str1 = “sunday”, str2 = “saturday” Output: 3 4.Input: str1 = “cat”, str2 = “cut” Output: 1 5.Input: str1 = “girl”, str2 = “grill” Output: 2

def editDistance(str1, str2, m, n):

if m == 0:

return n

if n == 0:

return m

if str1[m-1] == str2[n-1]:

return editDistance(str1, str2, m-1, n-1)

return 1 + min(editDistance(str1, str2, m, n-1),

editDistance(str1, str2, m-1, n),

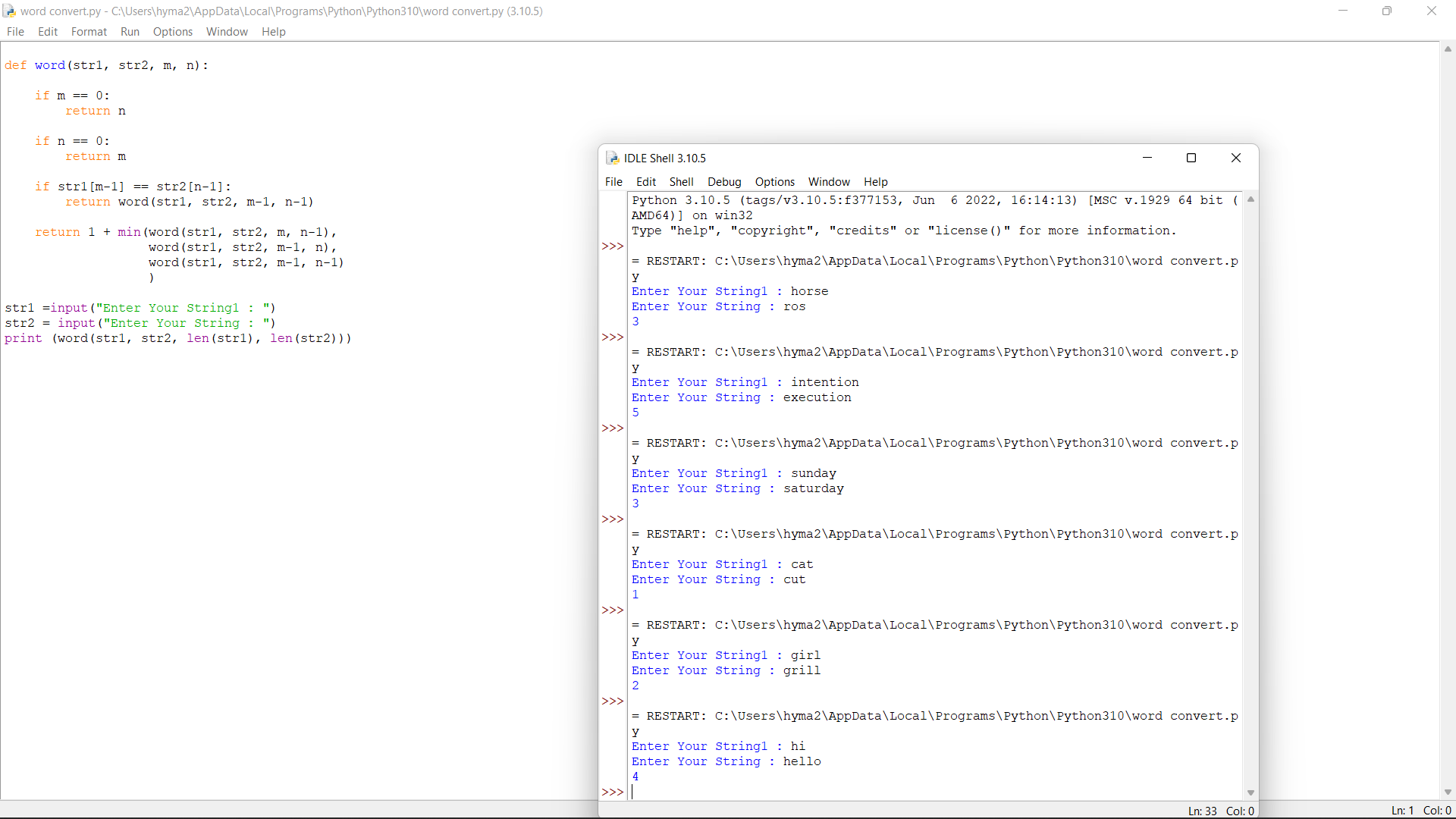
editDistance(str1, str2, m-1, n-1)

)

str1 =input("Enter Your String1 : ")

str2 = input("Enter Your String : ")

print (editDistance(str1, str2, len(str1), len(str2)))



5Q Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, they cannot move through that element. If the end isn’t reachable, return -1. Test Case: 1. Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9] Output: 3 (1-> 3 -> 9 -> 9) 2. Input: arr[] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] Output: 10 3. Input: arr[] = [2,3,1,1,4] Output: 2 4. Input: arr[] = [1, 3, 6, 1, 0, 9] Output: 3 5. Input: arr[] = [2,3,0,1,4]

def minJumps(arr, l, h):

if (h == l):

return 0

if (arr[l] == 0):

return float('inf')

min = float('inf')

for i in range(l + 1, h + 1):

if (i < l + arr[l] + 1):

jumps = minJumps(arr, i, h)

if (jumps != float('inf') and

jumps + 1 < min):

min = jumps + 1

return min

arr=eval(input("Enter list:"))

n=len(arr)

print('Minimum number of jumps to reach',

'end is', minJumps(arr, 0, n-1))

8. Given an array of strings strs, group the anagrams together. You can return the answer in any order. An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. strs[i] consists of lowercase English letters. Test Cases:

1.Input: strs = ["eat","tea","tan","ate","nat","bat"] Output: [["bat"],["nat","tan"],["ate","eat","tea"]] 2.Input: strs = [""] Output: [[""]]

3.Input: strs = ["a"] Output: [["a"]]

4.strs= “banana”

5.strs= 12345

def groupAnagrams(words):

anagrams = []

if not words:

return anagrams

nums = [''.join(sorted(word)) for word in words]

d = {}

for i, e in enumerate(nums):

d.setdefault(e, []).append(i)

for index in d.values():

collection = tuple(words[i] for i in index)

if len(collection) > 1:

anagrams.append(collection)

return anagrams

if \_name\_ == '\_main\_':

words = ["eat","tea","tan","ate","nat","bat"]

anagrams = groupAnagrams(words)

for anagram in anagrams:

print(anagram)

9. 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). Test case: 1.Input: s = "aa", p = "a" Output: false 2.Input: s = "aa", p = "a\*" Output: true 3.Input: s = "ab", p = ".\*" Output: true 4.Input: s = " aaa", p = "aa" Output: true 5.Input: s = "aab", p = "c\*a\*b" Output: true

def strrmatch(strr, pattern, n, m):

# empty pattern can only match with

# empty string

if (m == 0):

return (n == 0)

# lookup table for storing results of

# subproblems

lookup = [[False for i in range(m + 1)] for j in range(n + 1)]

# empty pattern can match with empty string

lookup[0][0] = True

# Only '\*' can match with empty string

for j in range(1, m + 1):

if (pattern[j - 1] == '\*'):

lookup[0][j] = lookup[0][j - 1]

# fill the table in bottom-up fashion

for i in range(1, n + 1):

for j in range(1, m + 1):

# Two cases if we see a '\*'

# a) We ignore ‘\*’ character and move

# to next character in the pattern,

# i.e., ‘\*’ indicates an empty sequence.

# b) '\*' character matches with ith

# character in input

if (pattern[j - 1] == '\*'):

lookup[i][j] = lookup[i][j - 1] or lookup[i - 1][j]

# Current characters are considered as

# matching in two cases

# (a) current character of pattern is '?'

# (b) characters actually match

else if (pattern[j - 1] == '?' or strr[i - 1] == pattern[j - 1]):

lookup[i][j] = lookup[i - 1][j - 1]

# If characters don't match

else:

lookup[i][j] = False

return lookup[n][m]

# Driver code

strr = "baaabab"

pattern = "\*\*\*\*\*ba\*\*\*\*\*ab"

# char pattern[] = "ba\*\*\*\*\*ab"

# char pattern[] = "ba\*ab"

# char pattern[] = "a\*ab"

# char pattern[] = "a\*\*\*\*\*ab"

# char pattern[] = "\*a\*\*\*\*\*ab"

# char pattern[] = "ba\*ab\*\*\*\*"

# char pattern[] = "\*\*\*\*"

# char pattern[] = "\*"

# char pattern[] = "aa?ab"

# char pattern[] = "b\*b"

# char pattern[] = "a\*a"

# char pattern[] = "baaabab"

# char pattern[] = "?baaabab"

# char pattern[] = "\*baaaba\*"

if (strrmatch(strr, pattern, len(strr), len(pattern))):

print("Yes")

else:

print("No")