1.Given a string S which is of the format USERNAME@DOMAIN.EXTENSION, the program must print the EXTENSION, DOMAIN, USERNAME in the reverse order.

Input Format:

The first line contains S.

Output Format:

The first line contains EXTENSION.
The second line contains DOMAIN.
The third line contains USERNAME.

Boundary Condition:

1 <= Length of S <= 100

Example Input/Output 1:

Input:

abcd@gmail.com

Output:

com
gmail
abcd

For example:

Input	Result

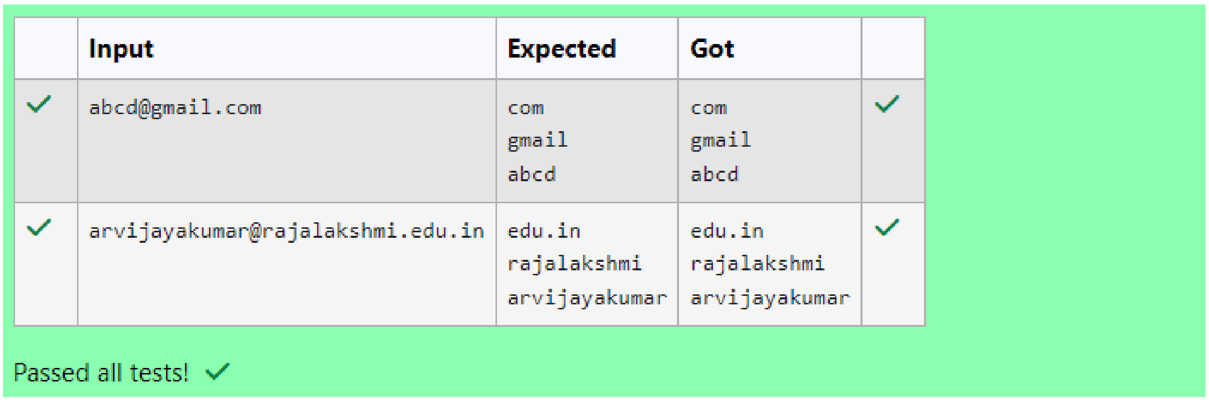
`arvijayakumar@rajalakshmi.edu.in`	`edu.in`
	`rajalakshmi`
	`arvijayakumar`

Coding:

```
s = input().strip()

name, dom = s.split('@')

i= dom.find('.')
```

```
d = dom[i + 1:]
d1 = dom[:i]

print(d)
print(d1)
print(name)
```

Output:



	Input	Expected	Got	
✓	abcd@gmail.com	com gmail abcd	com gmail abcd	✓
✓	arvijayakumar@rajalakshmi.edu.in	edu.in rajalakshmi arvijayakumar	edu.in rajalakshmi arvijayakumar	✓

Passed all tests! ✓

2. Given a **non-empty** string s and an abbreviation abbr, return whether the string matches with the given abbreviation.

A string such as "word" contains only the following valid abbreviations:

["word", "1ord", "w1rd", "wo1d", "wor1", "2rd", "w2d", "wo2", "1o1d", "1or1", "w1r1", "1o2", "2r1", "3d", "w3", "4"]

Notice that only the above abbreviations are valid abbreviations of the string "word". Any other string is not a valid abbreviation of "word".

Note:

Assume s contains only lowercase letters and abbr contains only lowercase letters and digits.

Example 1:

Input

internationalization

i12iz4n

Output

true

Explanation

Given **s** = "internationalization", **abbr** = "i12iz4n":

Return true.

Example 2:

Input

apple

a2e

Output

false

Explanatio

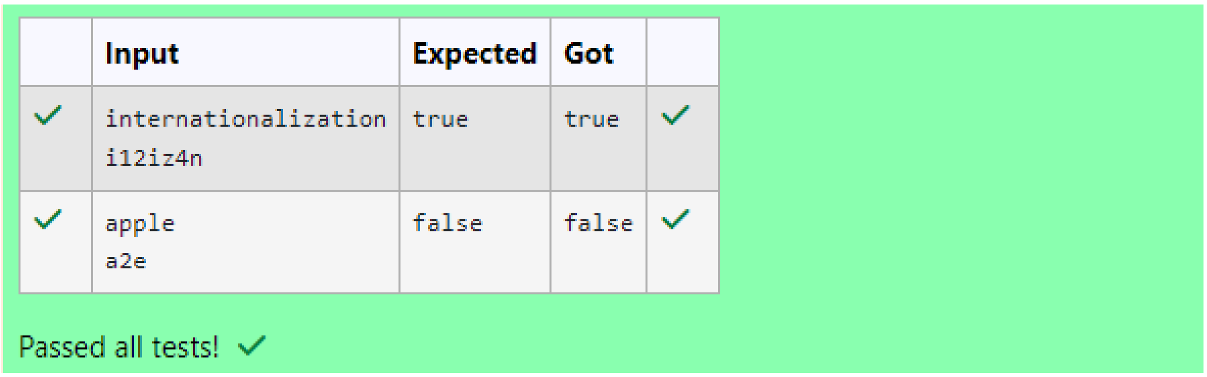
Given **s** = "apple", **abbr** = "a2e":

Return false.

Coding:

```
def abb(w: str,a: str) -> str:
    i=0
    j=0
    while i < len(w) and j < len(a):
        if a[j].isdigit():
            if a[j] == '0':
                return "false"

            num=0
            while j < len(a) and a[j].isdigit():
                num=num*10+int(a[j])
                j += 1
            i += num
        else:
            if i >= len(w) or w[i] != a[j]:
                return "false"

            i +=1
            j +=1

    return "true" if i == len(w) and j == len(a) else "false"

w=input()
a=input()
res=abb(w,a)
print(res)
```

Output:



	Input	Expected	Got	
✓	internationalization i12iz4n	true	true	✓
✓	apple a2e	false	false	✓

Passed all tests! ✓

3. Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

Input:
A man, a plan, a canal: Panama

Output:
1

Example 2:

Input:
race a car

Output:
0

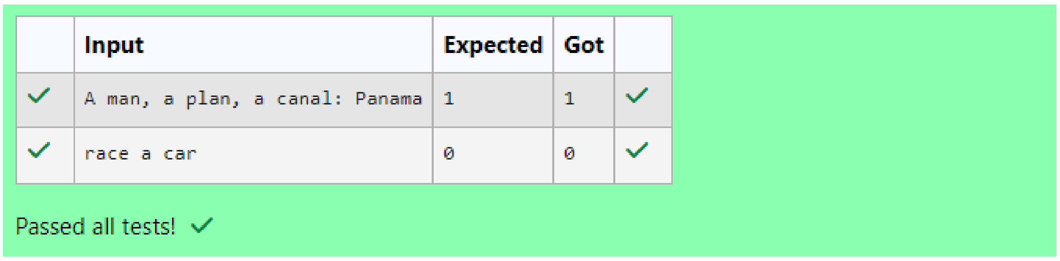
Constraints:

s consists only of printable ASCII characters.

Coding:

```
s = input()

f = ''.join(char.lower() for char in s if char.isalnum())

p = f == f[::-1]

print(1 if p else 0)
```

Output:



	Input	Expected	Got	
✓	`A man, a plan, a canal: Panama`	1	1	✓
✓	`race a car`	0	0	✓

Passed all tests! ✓

4. Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

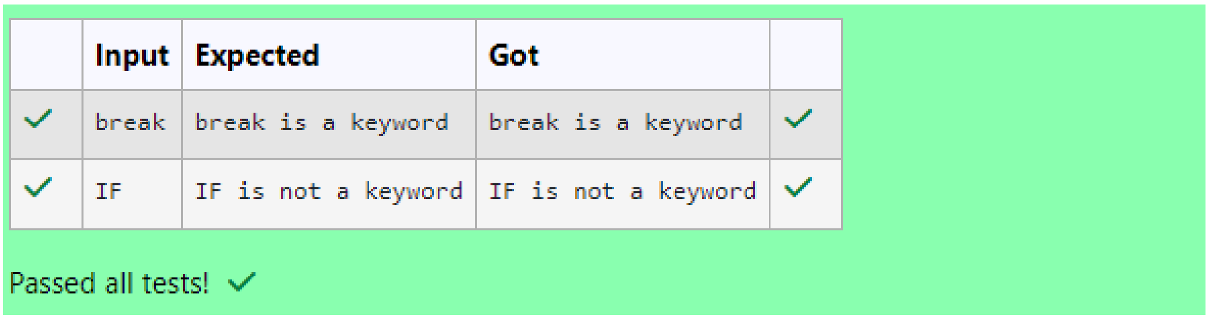
For example:

Input	Result
`break`	`break is a keyword`
`IF`	`IF is not a keyword`

Coding:

```
keywords =
{"break","case","continue","default","defer","else","for","func","goto","if","map","range","return",
"struct","type","var"}
i = input().strip()

if i in keywords:
    print(f"{i} is a keyword")
else:
    print(f"{i} is not a keyword")
```

Output:



	Input	Expected	Got	
✓	`break`	`break is a keyword`	`break is a keyword`	✓
✓	`IF`	`IF is not a keyword`	`IF is not a keyword`	✓

Passed all tests! ✓

5. A pangram is a sentence where every letter of the English alphabet appears at least once.

Given a string sentence containing only lowercase English letters, return true if sentence is a pangram, or false otherwise.

Example 1:

Input:

thequickbrownfoxjumpsoverthelazydog

Output:

true

Explanation: sentence contains at least one of every letter of the English alphabet.

Example 2:

Input:

arvijayakumar

Output: false

Constraints:

1 <= sentence.length <= 1000

sentence consists of lowercase English letters.

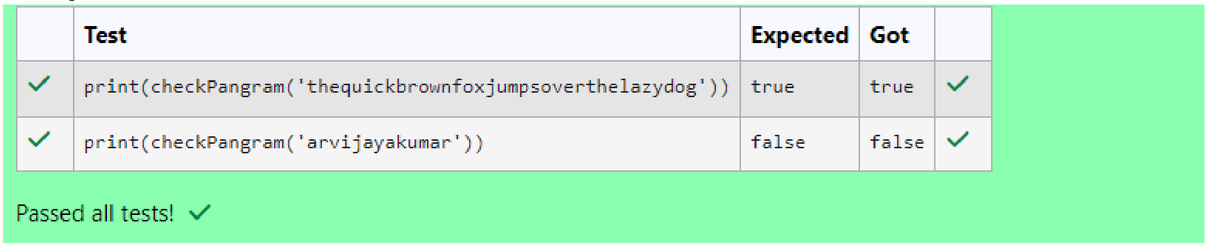
For example:

Test	Result
`print(checkPangram('thequickbrownfoxjumpsoverthelazydog'))`	`true`
`print(checkPangram('arvijayakumar'))`	`false`

Coding:

```
def checkPangram(s):
    a=set('abcdefghijklmnopqrstuvwxyz')
    b=set(s)
    return "true" if a.issubset(set(s)) else "false"
```

Output:



	Test	Expected	Got	
✓	`print(checkPangram('thequickbrownfoxjumpsoverthelazydog'))`	true	true	✓
✓	`print(checkPangram('arvijayakumar'))`	false	false	✓

Passed all tests! ✓

6. Write a Python program to get one string and reverses a string. The input string is given as an array of characters char[].

You may assume all the characters consist of printable ascii characters.

Example 1:

Input:
hello
Output:
olleh

Example 2:

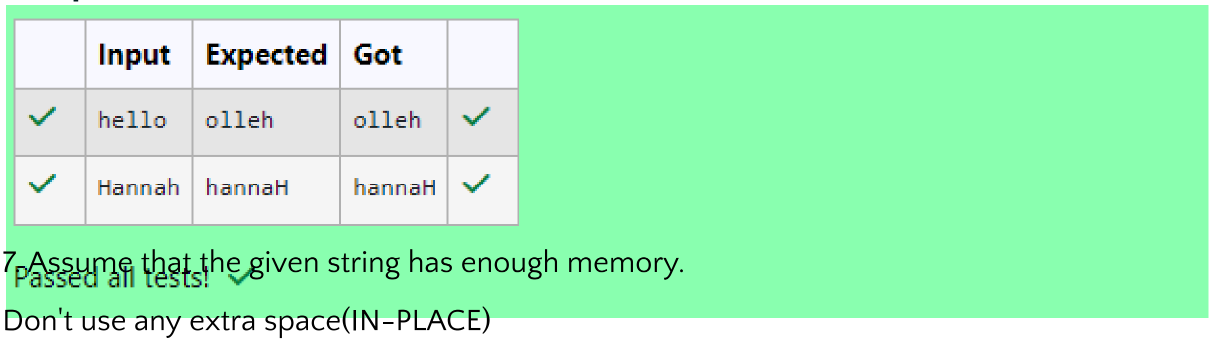
Input:
Hannah

Output:
hannaH

Coding:

```
def r(s):
    return s[::-1]
n =input()
rs=r(n)
print(rs)
```

Output:



	Input	Expected	Got	
✓	hello	olleh	olleh	✓
✓	Hannah	hannaH	hannaH	✓

Passed all tests! ✓

7.Assume that the given string has enough memory.

Don't use any extra space(IN-PLACE)

Sample Input 1

a2b4c6

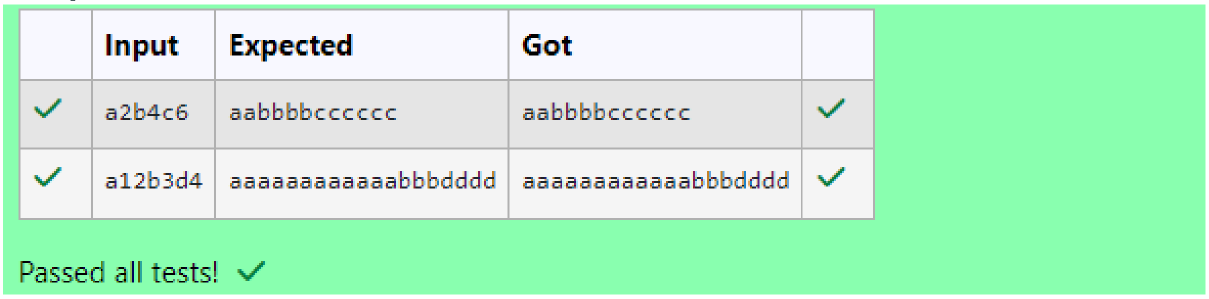
Sample Output 1

aabbbbcccccc

Coding:

```
def r(s):
  s=list(s)
  n=len(s)
  i=0
  j=0
  res=[]

  while i < n:
    char = s[i]
    i+=1
    count=0

    while i<n and s[i].isdigit():
      count=count*10+int(s[i])
      i+=1

    res.extend([char]*count)

  return ''.join(res)

n=input()
a=r(n)
print(a)
```

Output:



	Input	Expected	Got	
✓	a2b4c6	aabbbbcccccc	aabbbbcccccc	✓
✓	a12b3d4	aaaaaaaaaaaabbbdddd	aaaaaaaaaaaabbbdddd	✓

Passed all tests! ✓

8. Find if a String2 is substring of String1. If it is, return the index of the first occurrence. else return -1.

Sample Input 1

thistest123string

123

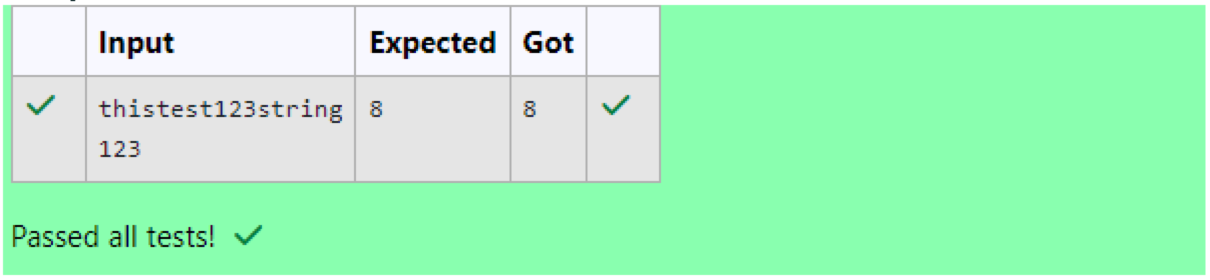
Sample Output 1

8

Coding:

```
def res(s1,s2):
    i = s1.find(s2)
    return i


s1=input()
s2=input()
print(res(s1,s2))
```

Output:



	Input	Expected	Got	
✓	thistest123string 123	8	8	✓

Passed all tests! ✓

9. The program must accept **N** series of keystrokes as string values as the input. The character ^ represents undo action to clear the last entered keystroke. The program must print the string typed after applying the undo operations as the output. If there are no characters in the string then print **-1** as the output.

Boundary Condition(s):

1 <= N <= 100
1 <= Length of each string <= 100

Input Format:

The first line contains the integer N.
The next N lines contain a string on each line.

Output Format:

The first N lines contain the string after applying the undo operations.

Example Input/Output 1:

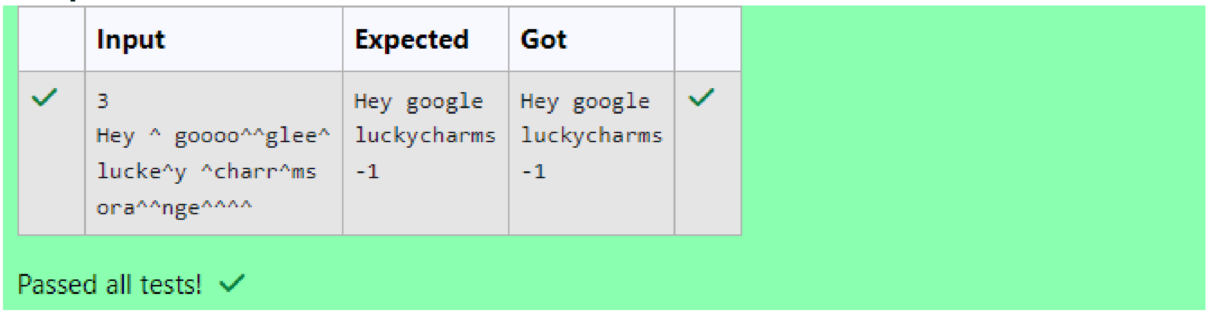
```
Input:
3
Hey ^ goooo^^glee^
lucke^y ^charr^ms
ora^^nge^^^^
```

```
Output:
Hey google
luckycharms
-1
```

Coding:

```
def r(s):

    a=[]

    for char in s:

        if char == '^':

            if a:

                a.pop()

        else:

            a.append(char)


    return ''.join(a) if a else '-1'


def main():
    import sys
    input=sys.stdin.read
    d=input().strip().split('\n')

    n=int(d[0])
    res=[]

    for i in range(1,n+1):
        res.append(r(d[i]))

    for res1 in res:
        print(res1)

if __name__ == "__main__":
    main()
```

Output:



	Input	Expected	Got	
✓	3 Hey ^ goooo^^glee^ lucke^y ^charr^ms ora^^nge^^^^	Hey google luckycharms -1	Hey google luckycharms -1	✓

Passed all tests! ✓

10. Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

Constraints:

1 <= s.length <= 10^4

s consists of parentheses only '()[]{}'.

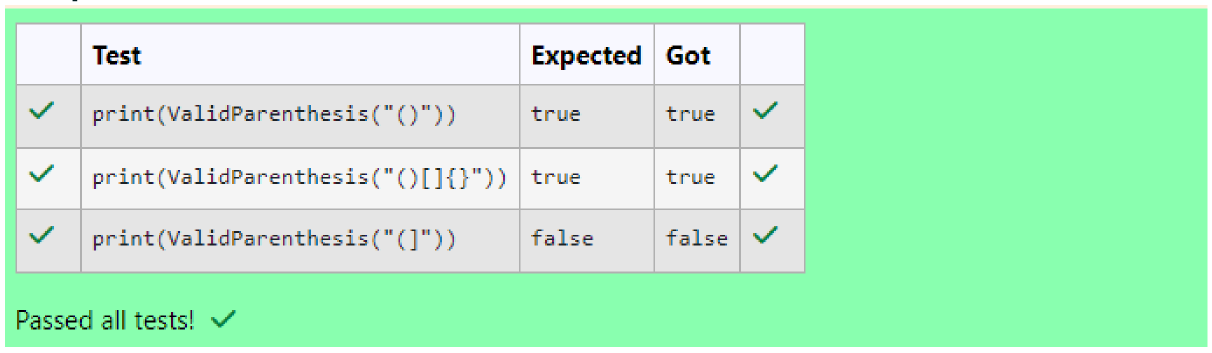
For example:

Test	Result
`print(ValidParenthesis("()"))`	`true`
`print(ValidParenthesis("()[]{}"))`	`true`
`print(ValidParenthesis("(]"))`	`false`

Coding:

```
def ValidParenthesis(s):
    a={')':'(', '}':'{', ']':'['}


    st=[]


    for char in s:
        if char in a:
            e = st.pop() if st else '#'
            if a[char] != e:
                return "false"
        else:
            st.append(char)
    return "true" if not st else "false"
```

Output:



	Test	Expected	Got	
✓	`print(ValidParenthesis("()"))`	`true`	`true`	✓
✓	`print(ValidParenthesis("()[]{}"))`	`true`	`true`	✓
✓	`print(ValidParenthesis("(]"))`	`false`	`false`	✓

Passed all tests! ✓