**=== reverse\_string using a Stack ===**

def reverse\_string(s):  
 stack = []  
 reversed\_string = ""  
  
 # push each character of the string onto the stack  
 for char in s:  
 stack.append(char)  
  
 # pop each character from the stack and append to reversed\_string  
 while len(stack) != 0:  
 reversed\_string += stack.pop()  
  
 return reversed\_string  
  
print(reverse\_string(input()))

**=== reverse\_string using Slicing ===**

return s[::-1]

* **Push:**Adds a new element on the stack.
* **Pop:**Removes and returns the top element from the stack.
* **Peek:**Returns the top (last) element on the stack.
* **isEmpty:**Checks if the stack is empty.
* **Size:**Finds the number of elements in the stack.

stack = []

# Push

stack.append('A')

stack.append('B')

stack.append('C')

print("Stack: ", stack)

# Peek

topElement = stack[-1]

print("Peek: ", topElement)

# Pop

poppedElement = stack.pop()

print("Pop: ", poppedElement)

# Stack after Pop

print("Stack after Pop: ", stack)

# isEmpty

isEmpty = not bool(stack)

print("isEmpty: ", isEmpty)

# Size

print("Size: ",len(stack))

**=== shift right with deque ===**

from collections import deque  
  
kids = deque(input().split())  
toss = int(input())  
  
while len(kids) > 1:  
 counter = toss % len(kids)  
 if counter == 0:  
 print(f"Removed {kids.pop()}")  
 else:  
 for i in range(counter - 1):  
 kid = kids.popleft()  
 kids.append(kid)  
 print(f"Removed {kids.popleft()}")  
  
print(f"Last is {''.join(kids)}")