## Task01

class stack:

stack=[]

pointer=-1

def push (self,ele):

self.stack.append(ele)

self.pointer+=1

def peek (self):

return (self.stack[self.pointer])

def pop (self):

value =self.stack[self.pointer]

self.stack=self.stack[:-1]

self.pointer-=1

return value

def faka(self):

if self.pointer==-1:

return True

else:

return False

def paranthesis(self,s=""):

stack1 = stack()

c = ["(","{","["]

d = [")","}","]"]

for i in s:

if i in c:

stack1.push(i)

if i in d:

if stack1.faka() == True:

return f"This expression is NOT correct.\nError at character #{s.index(i)+1}. '{i}'- not closed."

ele = stack1.pop()

index1 = c.index(ele)

index2 =d.index(i)

if (index1 != index2):

return f"This expression is NOT correct.\nError at character #{s.index(ele)+1}. '{ele}'- not closed."

return "This expression is correct"

a=input(" ")

kun=stack()

b=kun.paranthesis(a)

print(b)

## Task 02

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.next = None

class linklist:

def \_\_init\_\_(self):

self.head =Node(None)

def faka(self):

if self.head == None:

return True

else:

return False

def push(self, k):

if self.head == None:

self.head = Node(k)

else:

b = Node(k)

b.next = self.head

self.head = b

def pop(self):

if self.head == None:

return "Empty"

else:

off = self.head.value

self.head = self.head.next

return off

def paranthesis(self,s=""):

stack = linklist()

c = ["(","{","["]

d = [")","}","]"]

for i in s:

if i in c:

stack.push(i)

if i in d:

if stack.faka() == True:

return f"This expression is NOT correct.\nError at character #{s.index(i)+1}. '{i}'- not closed."

ele = stack.pop()

index1 = c.index(ele)

index2 =d.index(i)

if (index1 != index2):

return f"This expression is NOT correct.\nError at character #{s.index(ele)+1}. '{ele}'- not closed."

return "This expression is correct"

a=input(" ")

kun=linklist()

b=kun.paranthesis(a)

print(b)