Q) Given a number n, print all natural nos till n. Code the solution recursively

Ex $n = 4$

Output →
1
2
3
4

(1) Base Case

(2) Recursion Intuition

(3) Self work

f(n) =

assume this works correctly

It prints natural no's till n.

f(n-1) → after this → print(n)

this also works perfectly

assume

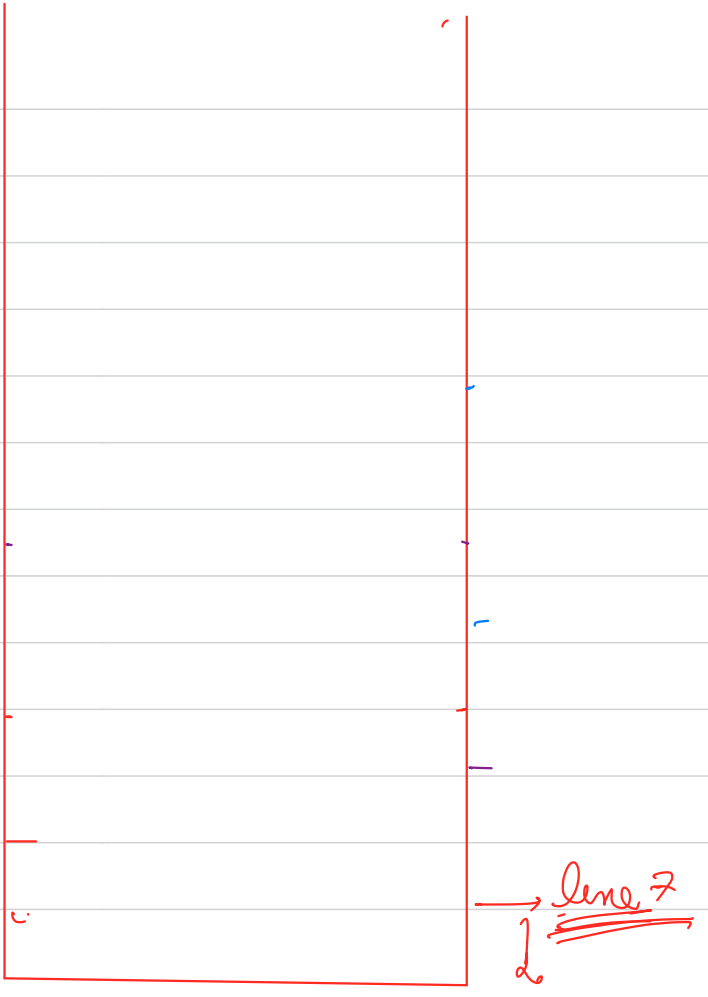
self work

Base Case → if (n == 1)
print(1)
return

```python
def print_increasing(n):
    if(n == 1):
        print(1)
        return

    # Recursive Assumption
    print_increasing(n-1)
    print(n)
    return


n = int(input())
print_increasing(n)
```

Console

1
2
3
4

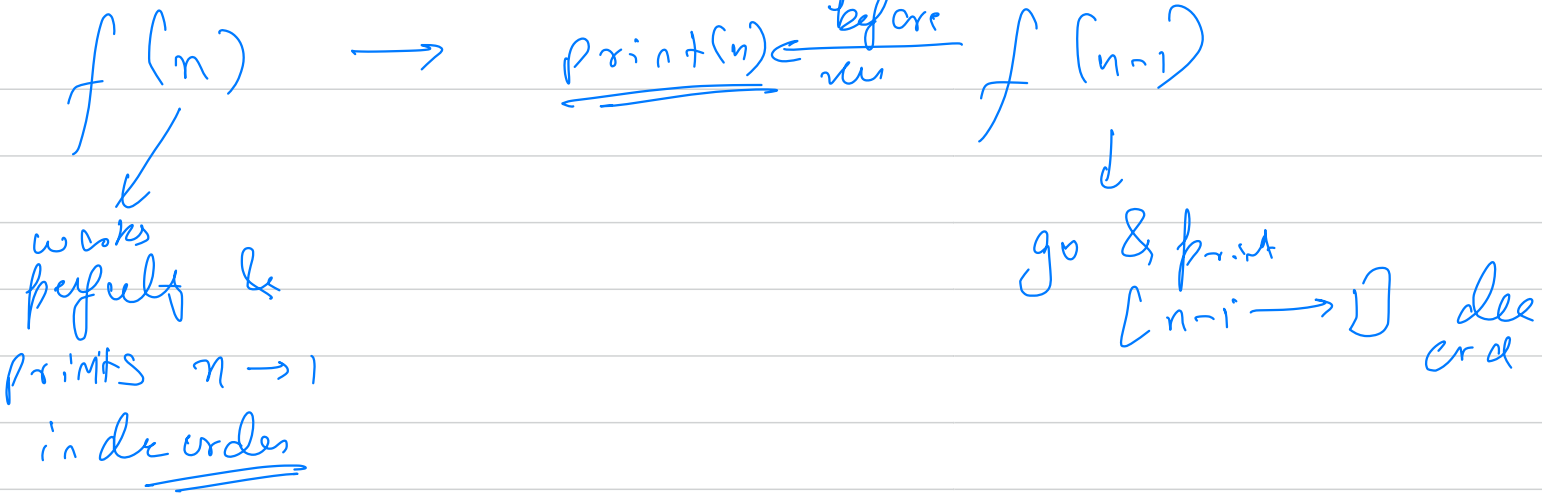


Q. Given a number n, print all natural no's till n in decreasing order. Solve it recursively

n = 4 ⟶ 4
3
2
1

assume anyhow you're able to print (n-1) → 1 in decreasing order

I will print 'n' myself before every thing


f(n) → print(n) before f(n-1)
works perfectly & prints n → 1 in decreasing order
go & print [n-1 → 1] decreasing

TC → O(n)

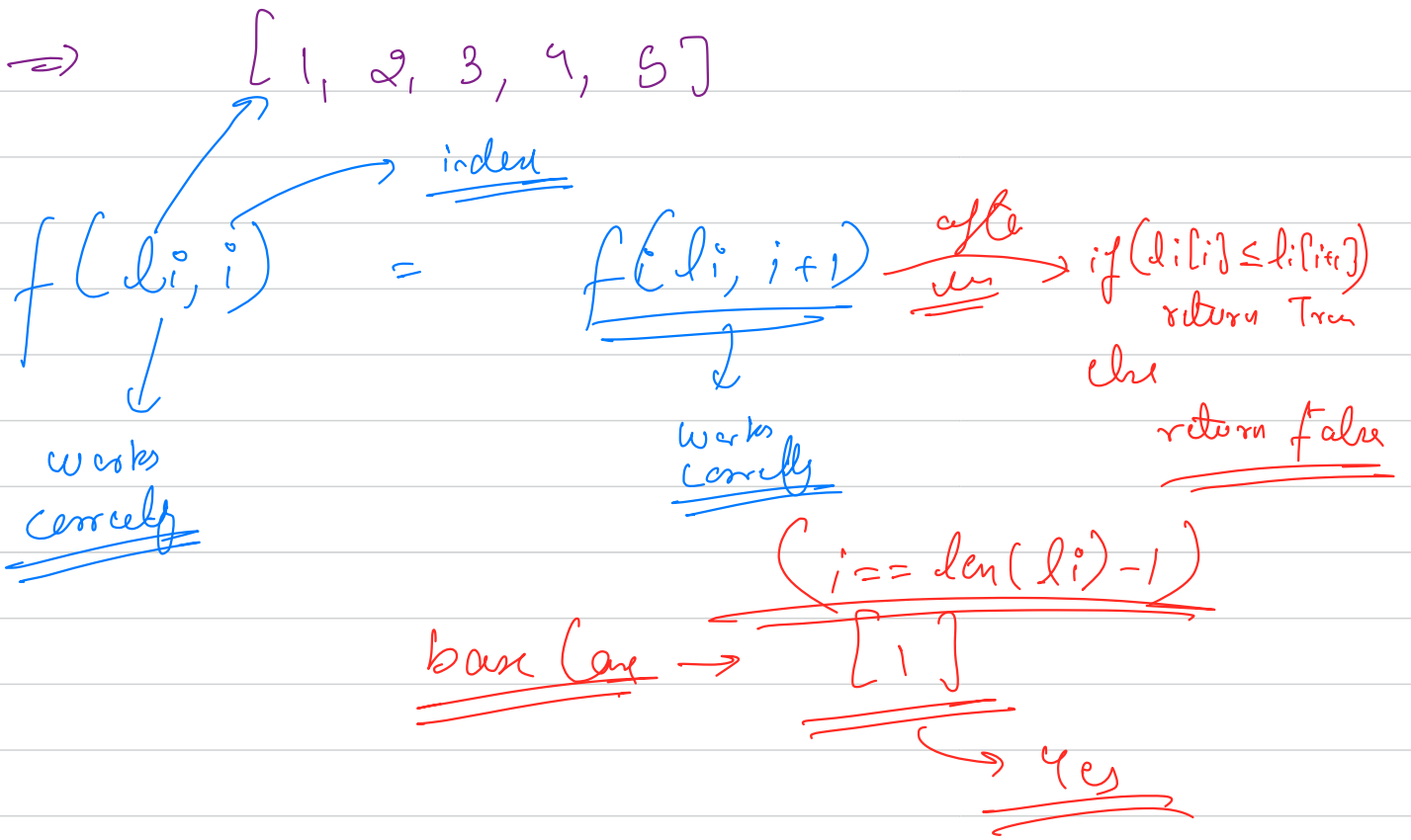
SC → O(n)

Q» Given a list of integers, write a function to check if the list is arranged in increasing order. Solve it recursively.

Ex [1, 2, 3, 4, 5] → Yes

[1, 11, 21, 31] → Yes

[1, 2, 5, 3, 4] → NO


[1, 2, 3, 4, 5]
index
f(li, i) = f(li, i+1)
works correctly
works correctly
after
if (li[i] ≤ li[i+1])
return True
else
return False
(i == len(li) - 1)
base case →
[1]
Yes

[1, 2, 3, 4, 5]

↓

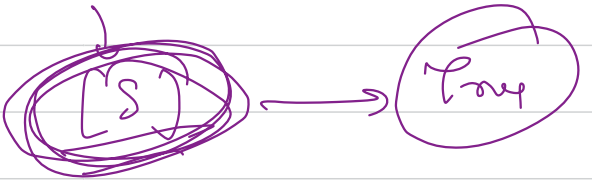
[2, 3, 4, 5]

↓

[3, 4, 5]

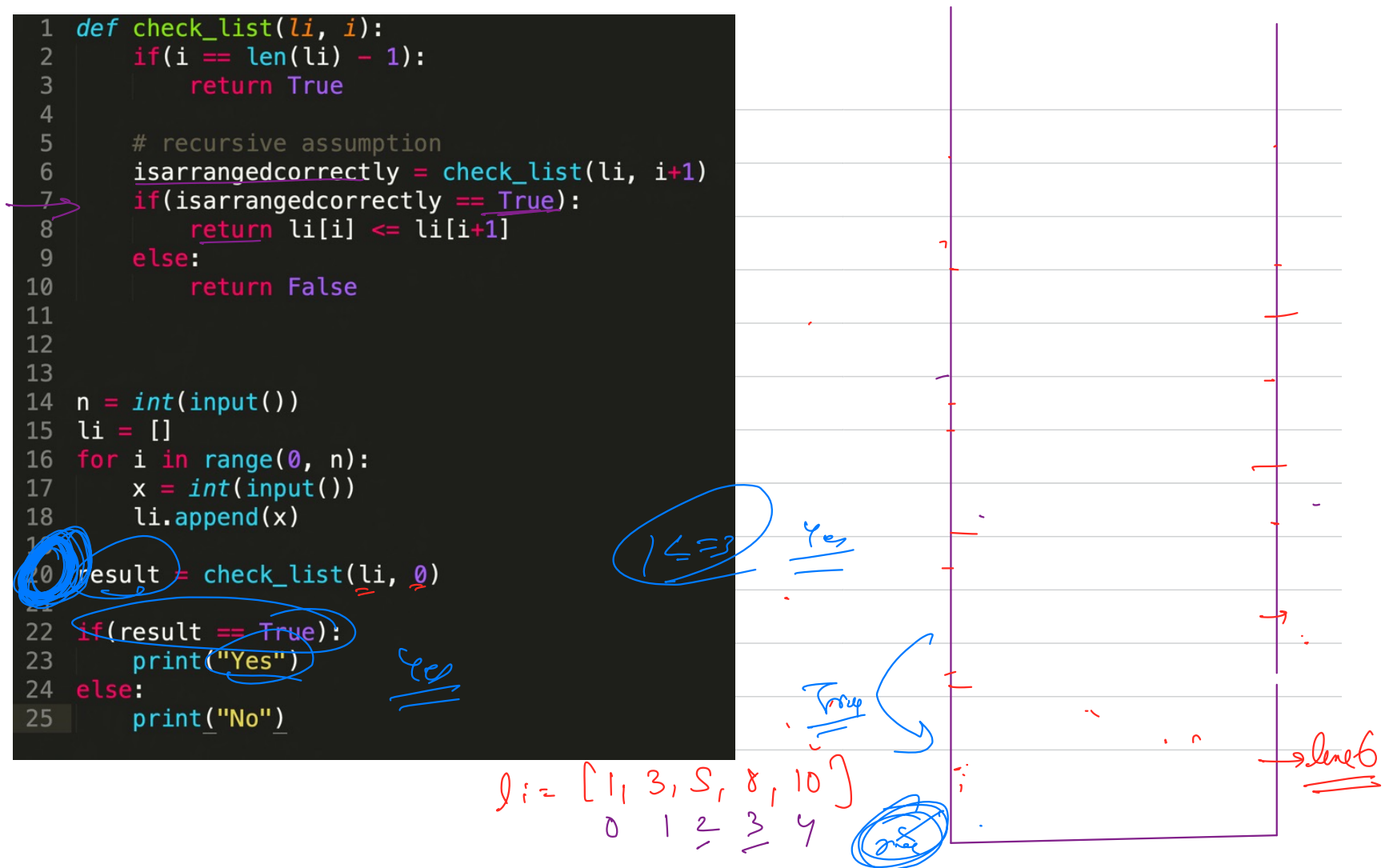
↓

[4, 5]



i == len(li) - 1

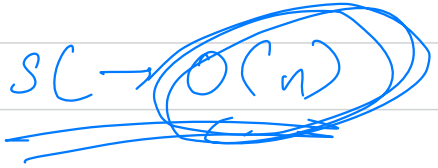
```python
def check_list(li, i):
    if(i == len(li) - 1):
        return True

    # recursive assumption
    isarrangedcorrectly = check_list(li, i+1)
    if(isarrangedcorrectly == True):
        return li[i] <= li[i+1]
    else:
        return False



n = int(input())
li = []
for i in range(0, n):
    x = int(input())
    li.append(x)

result = check_list(li, 0)

if(result == True):
    print("Yes")
else:
    print("No")
```

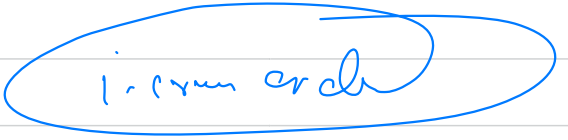


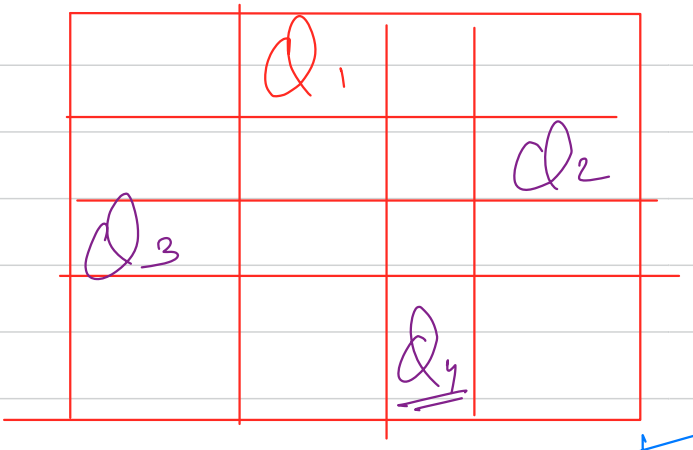
```python
def check_list(li, i):
    if(i == len(li) - 1):
        return True

    # recursive assumption
    isarrangedcorrectly = check_list(li, i+1)
    if(isarrangedcorrectly == True):
        return li[i] <= li[i+1]
    else:
        return False



n = int(input())
li = []
for i in range(0, n):
    x = int(input())
    li.append(x)

result = check_list(li, 0)

if(result == True):
    print("Yes")
else:
    print("No")
```

TC → O(n)

SC → O(n)







$n = 4$

$n \times n$ → board

Queen → n queens

$n = 4$

→ 2

No queen attack the other

Q. From any step, you can either jump by 1 step or 2 step.

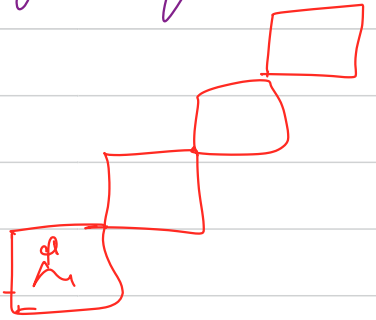
In how many different ways you can reach the last step ??

n=3

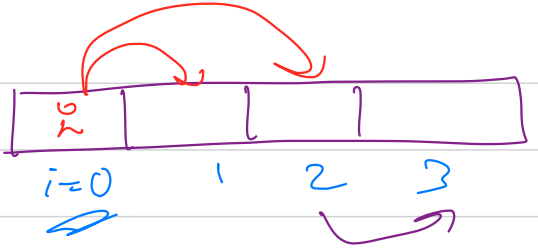
→ 1 1 1

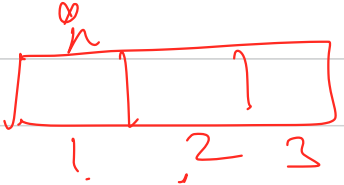
1 2

2 1



- Base Case
- Self work
- Recursion





1 1 } (2→3)

2 3 (3-3)

1 1 1 } (1→3)
1 2

2 1 } (2→3)

```python
def ways(n, i, path):
    # i ->integer -> represents where
    # path -> string -> store the pat
    if(i > n):
        # if you have made an invalid
        return
    if(i == n):
        # this base case shows u r at
        # if u r at the destination j
        print(path)
        return

    ways(n, i+1, "1 " + path)
    ways(n, i+2, "2 " + path)

n = int(input())
ways(n, 0, "")
```