#Palindrome Permutation: Given a string, write a function to check if it is a permutation of a palindrome. A palindrome is a word or phrase that is the same forwards and backwards. A permutation is a rearrangement of letters. The palindrome does not need to be limited to just dictionary words

#EXAMPLE

#Input: Tact Coa

#Output: True (permutations: "taco cat", "atco cta", etc.)

#The spaces do not matter at the palindrom A: No they don't

#Do caps matter? am I receiving caps on the imput A:they dont matter in the palindrome but

# you will be receiving them

# special caracters matter? like hyphens or else? A: No palindrome its just for letters but

# you could assume you will be getting them

#Can i use any other data structure like hash or classes? A:yes

# I would say that a palindrome ignoring spaces has the same amout of chars for the left

# and the right side, if is odd then there is a "center" that its char count must be odd

# asd ddas for EXAMPLE

# count of chars: a=2 d=3 s=2 this means that its permutations of multiple palindromes

# another example "counting continue" c=2 o=2 u=2 n=4 t=2 i=2 g=1 e=1 this was so close but it has 2 odd counts so id assume that there is palindrome here

def palindrome(str1):#Solution O(n)

ith=0

caps\_to\_lower = ord('a') - ord('A')

record=[0]\*255

while ith < len(str1):# this is O(n)

ith\_c = ord(str1[ith])

if not (ith\_c < ord('A') or (ith\_c > ord('Z') and ith\_c < ord('a')) or ith > ord('z')):

#this means im interested in this char

if ith\_c >= ord('A') and ith\_c <= ord('Z'):

ith\_c+=caps\_to\_lower

record[ith\_c]+=1

ith+=1

odd\_count=0

for char in record:# this is O(n)

if char%2 == 1:

odd\_count+=1

if odd\_count>1:

return False

return True

print (palindrome("asd dedas"))

print (palindrome("Tact Coa"))

# Another solution would be sorting and check letter by letter

def palindrome2(str1):

new\_str=sorted(str1.lower())#this is o(nlogn)

ith = 0

odds=0

while ith < len(new\_str):#this is O(n)

current=new\_str[ith]

if ord(current) < ord('a') or ord(current) > ord('z'):

ith+=1

continue

count\_current=0

while ith < len(new\_str):

ith+=1

if new\_str[ith]==current:

count\_current+=1

else:

if count\_current%2==1:

odds+=1

if odds > 1:

return False

break

ith+=1

print (palindrome2("asd dedas"))

print (palindrome2("Tact Coa"))