def print\_array(arr):

n = len(arr)

new\_array=[]

for i in range(n):

new\_array.append(arr[i])

return new\_array

print(print\_array([2,3,4,5]))

new\_array = [2,3,4,1]

def sort\_array(new\_array):

n = len(new\_array)

for i in range(n):

for j in range(i+1):

if new\_array[i]<new\_array[j]:

new\_array[i],new\_array[j]=new\_array[j],new\_array[i]

else:

None

return new\_array

print(sort\_array(new\_array))

new\_array = [2,3,4,1]

def binary\_search(new\_array , target):

n = len(new\_array)

left = 0

right = n-1

mid = (left+right)//2

for i in range(n):

if new\_array[i] == target:

return i

elif new\_array[i] > target :

right = mid-1

elif new\_array[i] <target:

left = mid+1

else:

return None

print(binary\_search(new\_array, 4))

new\_array = [2,1,3]

def sequential\_search(new\_array , target):

n = len(new\_array)

for i in range(n):

for j in range(i+1):

if new\_array[j]==target:

return j

print(sequential\_search(new\_array , 3))

def sum\_of\_two(new\_array , target):

n= len(new\_array)

for i in range(n):

for j in range(i+1 ,n):

if new\_array[i]+new\_array[j]==target:

return i,j

return None

print(sum\_of\_two([3,4,5,6] , 10))

def reverse\_array(array):

n = len(array)

left = 0

right = n-1

while(left<right):

array[left] , array[right]= array[right] , array[left]

return array

print(reverse\_array([3,4,5,6]))

def check\_palimdrome(array):

n= len(array)

left = 0

right = n-1

for i in range(n):

if array[left]!=array[right]:

return False

return True

print(check\_palimdrome([2,3,3,2]))

"""

array = []

n = int(input("enter the number of element in array:"))

for i in range(n):

x = int(input("enter the element"))

array.append(x)

print(array)"""

"""

def merge\_sorted\_arrays(array1, array2):

merged\_array = []

i, j = 0, 0

# This loop runs a maximum of min(len(array1), len(array2)) times.

# So, its time complexity contributes to O(n), where n is the minimum size between array1 and array2.

while i < len(array1) and j < len(array2):

if array1[i] <= array2[j]:

merged\_array.append(array1[i])

i += 1

else:

merged\_array.append(array2[j])

j += 1

def merge\_sort(array):

if len(array)<=1:

return array

mid = len(array)//2

left\_side = merge\_sort(array[:mid])

right\_side = merge\_sort(array[mid:])

return merge\_sorted\_arrays(left\_side , right\_side)

print(merge\_sort([3,4,5,32,2]))"""

def sorted\_square\_array(array):

n =len(array)

new\_array =[0]\*n

for i in range(n):

new\_array[i]= array[i]\*\*2

new\_array.sort()

return new\_array

print(sorted\_square\_array([5,4,3]))

# monotonic means either it is monotonically increse or monotoinc deresediing but non increse decrese

def check\_monotonic\_array(array):

n = len(array)

first = array[0]

last =array[n-1]

if first>last :

#monotonically decresing

for k in range(n-1):

if array[i]< array[i+1]:return False

elif first==last:

for k in range(n-1):

if array[k]!=array[k+1]:return False

else:

for k in range(n-1):

if array[k] > array[k+1]:return False

return True

print(check\_monotonic\_array([3,5,6,5]))

def rotated\_array(array , k):

n = len(array)

k=k%n

left = 0

right = n-1

temp = array[-k:]

for i in reversed(range(0,len(array)-k)):

array[i+k]=array[i]

for i in range(len(temp)):

array[i]=temp[i]

return array

print(rotated\_array([2,4,5,6] , 2))

def find\_max\_area(array):

max\_area = 0

for i in range(len(array)):

for j in range(i+1 , len(array)):

area = min(array[i] , array[j])\*(j-i)

max\_area = max(area , max\_area)

return max\_area

print(find\_max\_area([3,5,6,6]))

"""

def non\_rep\_char(array):

for i in range(len(array)):

repeat = False

for j in range(len(array)):

if i!=j and array[i]==array[j]:

repeat = True

if repeat == False:

return i

return None

print(non\_rep\_char('rrtgbb34t'))"""

def check\_pam(str):

new\_str = ''

for i in reversed(range(len(str))):

new\_str+=str[i]

if str==new\_str:

return True

return False

print(check\_pam('dgg'))

def is\_pam(str) :

array=[]

for i in reversed(range(len(str))):

array.append(str[i])

if ''.join(array)==str:

return True

return False

print(is\_pam('gjbg'))

def is\_spam(str):

n = len(str)

i = 0

j = n-1

while(i<j):

if str[i]!=str[j]:

return False

i+=1

j-+1

return True

print(is\_spam('fjnng'))