Week 1

Write a function that randomly shuffles an array of integers and explain the rationale behind its implementation.

#ask = list(input("Enter list of integers"))

import random

ask=[1,2,5,9,6,9,4,2,6,5,8,7,5,1,2,6,5,9,6] # 1

empty = [] # 1

print (len(ask)) # 1

while len(ask) > 0:

rnd=random.randint(0,len(ask)-1) #BigO (N) Run time

empty.append(ask[rnd])

ask.pop(rnd)

print(empty)

Count the number of trailing 0s in a factorial number.

def fact(x): #Finds Factorial

if (x >1): #1 BigO

return x \* fact(x-1)

else:

return 1

print (fact(15))

def count (x): #Counts trailing 0s

i = 5

zeros = 0

while x / i >= 1: #N BigO

zeros += x // i

i \*= 5

return zeros

print(count(15))

Week 2

Write the pseudocode for a function which returns the highest perfect square which is less or equal to its parameter (a positive integer). Implement this in a programming language of your choice.

def SQRT(X)

set answer to 0

if x is <- to or more than 0

while loop

answer\*answer smaller than x

answer <- answer + 1

if answer isnt <- to x

then print x, ' is not a perfect square'

else

print x, 'is a perfect square'

else

print x, 'is not a positive number.'

Look back at last week's tasks. Describe the run-time bounds of these algorithms using Big O notation.

Write the pseudocode corresponding to functions for addition, subtraction and multiplication of two matrices, and then compute A=B\*C –2\*(B+C), where B and C are two quadratic matrices of order n. What is the run-time?

m < 3

A < input MATRIX

B < input MATRIX

for i <- 1 to range(n):

for i <- 1 to range(n):

print A[i,j]

for i <- 1 to range(n):

print L[i]

L[0]=2

L[1]=1

L[2]=2

Week 3

Write the pseudocode and code for a function that reverses the words in a sentence. Input: "This is awesome" Output: "awesome is This". Give the Big O notation.

def rec(n, i):

value="" #1 BigO Runtime

if i<len(b)-1:

value=rec(n,i+1)

return value+" "+n[i]

a="awesome is This"

b=a.split(' ')

print(rec(b,0))

#BigO Runtime 1

value <- empty value

if i < length of b then - 1

value <- REC

return value

a <- user input

b <- function to reverse string

print REC

Write a recursive function (pseudocode and code) to check if a number n is prime (hint: check whether n is divisible by any number below n).

def primez(n, n2):

#Recursive #BigO (N)

while n2 >= 2: #BigO (N)

if n % n2 == 0: #Base case

print ("This is a prime number")

return False

else:

return primez(n, n2-1) #Calling itself

else:

print ("This number is not a prime number")

return 'True'

primez(7,3)

primez(5,4)

primez(6,5)

def PRIMEZ(N,N2)

while n2 >= 2

if n % n2 = 0

print is prime

return False

else

return PRIMEZ(N. N2-1)

else

print not prime

return True

8. Write a recursive function (pseudocode and code) that removes all vowels from a given string s. Input: "beautiful" Output: "btfl".

def vowel(s):

if not s:

return s

elif s[0] in "aeiouAEIOU":

return vowel(s[1:]) #Removes vowels in string

return s[0] + vowel(s[1:]) #Removes vowels in string

#Recursive code = BigO (N)

print(vowel("My Name Is DJ Stutz"))

#BigO (N)

def VOWEL(S)

if not s

return s

elif s[0] in "aeiouAEIOU"

return VOWEL

print VOWEL "users input"