# Lab6.py answers

import sys

# Produce a Xmas tree shape

def Lab6\_1():

print ( sys.\_getframe().f\_code.co\_name )

rows = 5

starCount = 1

biggest = 9

while ( rows > 0 ):

spaces = biggest / 2 - ( starCount / 2 )

while ( spaces > 0 ):

print ( " ", end="")

spaces -= 1

printCount = 0

while ( printCount < starCount ):

print ( "\*", end="")

printCount = printCount + 1

rows = rows - 1

starCount += 2

print ("")

spaces = biggest / 2 - 1

while ( spaces > 0 ):

print (" ", end="")

spaces -= 1

print ("\*")

# Produce a Xmas tree shape

def Lab6\_2():

print ( sys.\_getframe().f\_code.co\_name )

while ( True ):

rows = 0

response = input ( "Enter number of rows (1-10) Enter 0 to exit: " )

if ( response == "0" ):

print ("Done")

return

elif ( response != "" ):

rows = eval(response)

if ( rows >= 1 and rows <= 10 ):

starCount = 1

biggest = 30

while ( rows > 0 ):

spaces = biggest / 2 - ( starCount / 2 )

while ( spaces > 0 ):

print ( " ", end="")

spaces -= 1

printCount = 0

while ( printCount < starCount ):

print ( "\*", end="")

printCount = printCount + 1

rows = rows - 1

starCount += 2

print ("")

spaces = biggest / 2

while ( spaces > 0 ):

print (" ", end="")

spaces -= 1

print ("\*")

# Count positive and negative numbers entered by the user

def Lab6\_3():

print ( sys.\_getframe().f\_code.co\_name )

print("Enter a set of numbers, seperated by commas and ending with a 0. (each on its own line)")

positiveCount = 0

negativeCount = 0

while ( True ):

response = input ("")

if ( response == "0" ):

print ( "You entered %d positive number(s) and %d negative number(s)." % (positiveCount, negativeCount))

return

else:

numericValue = eval ( response )

if ( numericValue > 0 ):

positiveCount += 1

else:

negativeCount += 1

# Lab6\_4() Print multiplication table 1 - 9

'''

1 2 3 4 5 6 7 8 9

1 1

2 2 4

3 3 6 9

4 4 8 12 16

5 5 10 15 20 25

6 6 12 18 24 30 36

7 7 14 21 28 35 42 49

8 8 16 24 32 40 48 56 64

9 9 18 27 36 45 54 63 72 81

'''

def Lab6\_4():

print (" ", end = "")

for value in range ( 1, 10 ) :

print ( "%4d" % value, end = "" )

print ("")

for x in range ( 1, 10 ):

print ( "%d" % x, end = "" )

for y in range ( 1, x + 1):

print ( "%4d" % ( y \* x ), end = "" )

print ( "")

# Read a text file and print contents

def Lab6\_5():

print ( sys.\_getframe().f\_code.co\_name )

inputFile = open ( "lab6.txt" )

for line in inputFile:

print (line, end = "")

inputFile.close()

######################

inputFile = open ( "lab6.txt" )

print (inputFile.read(), end = "")

inputFile.close()

# Read in a file with numbers, print the MIN and MAX

def Lab6\_6():

print ( sys.\_getframe().f\_code.co\_name )

inputFileName = "Lab6\_6.txt"

minValue = 99999999999

maxValue = -minValue

inputFile = open ( inputFileName, "r" )

for line in inputFile:

if ( line == "" ):

break

value = eval ( line )

if ( value > maxValue ):

maxValue = value

elif ( value < minValue ):

minValue = value

print ( "The MAX value is %d, the MIN value is %d." % ( maxValue, minValue ))

inputFile.close()

# Read in a file with numbers, print the average

def Lab6\_7a():

print ( sys.\_getframe().f\_code.co\_name )

inputFileName = "Lab6\_6.txt"

total = 0

count = 0

inputFile = open ( inputFileName, "r" )

for line in inputFile:

if ( line == "" ):

break

count += 1

value = eval ( line )

total += value

print ( "The Total of %d numbers is %d, the average is %d." % ( count, total, (total / count) ))

inputFile.close()

# Read in a file with numbers, print the median

def Lab6\_7b():

print ( sys.\_getframe().f\_code.co\_name )

inputFileName = "Lab6\_6.txt"

listOfNumber = list()

inputFile = open ( inputFileName, "r" )

for line in inputFile:

if ( line == "" ):

break

listOfNumber.append ( eval ( line ) )

listOfNumber.sort()

index = int(len(listOfNumber) / 2)

median = listOfNumber [ index ]

print ( "The Median of the %d numbers is %d." % ( len(listOfNumber), median ))

inputFile.close()

#######################################################

#Lab6\_1()

#Lab6\_2()

#Lab6\_3()

#Lab6\_4() # Print multiplication table 1 - 9

#Lab6\_5() # Read a text file and print contents

#Lab6\_6() # Read in a file with numbers, print the MIN and MAX

Lab6\_7a() # Read in a file with numbers, print the median

Lab6\_7b() # Read in a file with numbers, print the median

# Lab7.py answers

def Lab7\_1\_Mypow ( x, y ):

retValue = 1

for loop in range ( 0, abs(y) ):

retValue = retValue \* x

if ( y < 0 ):

if ( retValue != 0 ):

retValue = 1 / retValue;

return retValue

# Prompt the user for two numeric values

def getValues():

#

return

def Lab7\_1():

exponents = [ 0, 1, 3, -5, 10, -3 ]

numbers = [ 0, 1, 2, -4, 8, -3]

for number in numbers:

for exponent in exponents:

value = Lab7\_1\_Mypow ( number, exponent )

print ( "%d ^ %d = %f" % ( number, exponent, value ) )

def sortem( val1, val2, val3 ):

Top = 0

Middle = 0

Bottom = 0

# Find Top value

if ( val1 > val2 and val1 > val3 ):

Top = val1

elif ( val2 > val1 and val2 > val3 ):

Top = val2

elif ( val3 > val1 and val3 > val2 ):

Top = val3

if ( val1 > val2 and val1 < val3 ):

Middle = val1

elif ( val2 > val1 and val2 < val3 ):

Middle = val2

elif ( val3 > val1 and val3 < val2 ):

Middle = val3

if ( val1 < val1 and val1 < val2 ):

Bottom = val1

elif ( val2 < val1 and val2 < val3 ):

Bottom = val2

elif ( val3 < val1 and val3 < val2 ):

Bottom = val3

results = [ Top, Middle, Bottom ]

return results

def Lab7\_2():

numbers = [ 3, -2, 5 ]

results = sortem ( numbers[0], numbers[1], numbers[2] );

print ( "Lab7\_2() -> " + str(results) )

numbers = [ -3, 2, 0 ]

results = sortem ( numbers[0], numbers[1], numbers[2] );

print ( "Lab7\_2() -> " + str(results) )

# expectedIndexCount = 5

def Lab7\_3\_getValue():

while ( True ):

response = input ( "Enter air quality index (0-100): " )

intResponse = eval ( response )

if ( intResponse >= 0 and intResponse <= 100 ):

return intResponse

def Lab7\_3\_getValuesA():

responses = []

while ( len ( responses ) < expectedIndexCount ):

responses.append ( Lab7\_3\_getValue() )

return responses

def Lab7\_3\_getValues():

response = input ( "Enter 5 air quality indicies separated by commas: ")

indiciesList = response.split(",")

if ( len ( indiciesList ) == expectedIndexCount ):

return indiciesList

else:

return None

def Lab7\_3\_DrawStars ( value ):

print ( "%3d" % ( value ), end = "")

for starIndex in range ( 0, value // 10 ):

print ( " \* ", end = "")

print ( "")

def Lab7\_3\_DrawHistogram ( indiciesList ):

indexIndex = 0

print ( "Time" )

for index in range ( 10, 15 ) :

Lab7\_3\_DrawStars ( indiciesList [ indexIndex ] )

indexIndex = indexIndex + 1

for index in range ( 0, 33 ) :

print ( "-", end = "" )

print ("")

print (" ", end = "")

for index in range ( 10, 101, 10 ) :

print ( "%d " % ( index) , end = "" )

print ("")

print (" Index")

def Lab7\_3():

indiciesList = Lab7\_3\_getValuesA()

if ( indiciesList != None ):

print ( "Indicies: " + str(indiciesList) )

Lab7\_3\_DrawHistogram ( indiciesList )

Lab7\_1()

Lab7\_2()

Lab7\_3()

# COMP 115 - Lab 8 answers

import random

def fib1 ( value ):

f1, f2 = 1, 1

print ( "Fib %d = %d, %d" % ( value, f1, f2), end = "" )

for index in range ( value - 2 ): # first two printed above

newFib = f1 + f2

print ( ", " + str(newFib), end = "" )

f1 = f2

f2 += 1

print ("")

def Lab8\_1():

response = 5

response = eval ( input ( "Enter an integer: " ))

fib1 ( response )

def fib2 ( value ):

f1, f2 = 1, 1

total = f1 + f2

print ( "Fib %d = %d, %d" % ( value, f1, f2), end = "" )

for index in range ( value - 2 ): # first two printed above

newFib = f1 + f2

total += newFib

print ( ", " + str(newFib), end = "" )

f1 = f2

f2 += 1

print ("")

return total

def Lab8\_2():

response = 8

#response = eval ( input ( "Enter an integer: " ))

total = fib2 ( response )

print ( "Total = %d" % ( total ) )

def die():

value = random.randint(1, 6)

return value

def Lab8\_3():

print ( "Random number = %d " % ( die() ))

def Lab8\_4():

for index in range ( 10 ):

print ( "Random number = %d " % ( die() ))

def DoRoll(guess, bet):

winnings = 0

for roll in range ( 3 ):

rollValue = die() + die()

print ( "Rolled: " + str(rollValue) )

if ( rollValue == guess ):

# Win!

if (guess in [ 2, 3, 5, 9, 11, 12 ]):

winnings = 4 \* bet

elif ( guess in [ 4, 6, 8, 10 ] ):

winnings = 3 \* bet

elif ( guess in [ 7 ] ):

winnings = 2 \* bet

return winnings

# Lose

return 0

def Lab8\_5():

#response = input ( "Enter your guess and a bet separated by a comma or space: " )

response = "8 12"

guessInt = 0

betInt = 0

parts = response.split(",")

if ( len(parts) != 2 ):

parts = response.split ( " " )

if ( len(parts) == 2 ):

guessInt = eval ( parts [ 0 ] )

betInt = eval ( parts [ 1 ] )

winnings = DoRoll(guessInt, betInt)

print ( "Winnings: $%d" % winnings )

#Lab8\_1()

#Lab8\_2()

#Lab8\_4()

#Lab8\_4()

Lab8\_5()