DATA STRUCTURES AND ALGORITHMS

# ASSIGNMENT #1

## Name: Muhammad Roll Number: K17-2421 SEC:B

Please Note: All codes are user independent. Multiple possible values are taken to test the functions.

# Question 1:

Write a short Python function, is multiple(n, m), that takes two integer values and returns True if n is a multiple of m, that is, n = mi for some integer i, and False otherwise.

Code:

def is\_multiple(n,m):

if m//n == m/n or n//m == n/m:

return True

else:

return False

print(is\_multiple(3,5))

print(is\_multiple(3,6))

print(is\_multiple(6,18))

Output:

False

True

True

# Question 2:

Write a short Python function, is even(k), that takes an integer value and returns True if k is even, and False otherwise. However, your function cannot use the multiplication, modulo, or division operators.

Code:

e = ['0' , '2' , '4' , '6' , '8']

def is\_even(k):

t = False

for a in e:

if k[len(k)-1] == a:

t = True

return t

print(is\_even('22'))

print(is\_even('23'))

print(is\_even('-24'))

print(is\_even('-25'))

Output:

True

False

True

False

# Question 3:

Write a short Python function, minmax(data), that takes a sequence of one or more numbers, and returns the smallest and largest numbers, in the form of a tuple of length two. Do not use the built-in functions min or max in implementing your solution.

Code:

def minmax(data):

if len(data) > 0:

mn = data[0]

mx = data[0]

for a in data[1:]:

if mx < a:

mx = a

if mn > a:

mn = a

return (mn,mx)

else:

print("Data is out of range")

print(minmax([2,3,1,5,11,87,1,0,45]))

print(minmax([]))

print(minmax([2,3,1,5,11,87,1,0,-45]))

Output:

(0, 87)

Data is out of range

None

(-45, 87)

# Question 4:

Write a short Python function that takes a positive integer n and returns the sum of the squares of all the positive integers smaller than n.

Code:

def squaresum(n):

t = 0

for a in range(1,n):

t += a\*a

return t

print(squaresum(5))

print(squaresum(-10))

print(squaresum(15))

Output:

30

0

1015

# Question 5:

Give a single command that computes the sum from Exercise R-1.4 (Question 4), relying on Python’s comprehension syntax and the built-in sum function.

Code:

def squaresum(n):

return sum([x\*x for x in range(1,n) if x > 0])

print(squaresum(5))

print(squaresum(-10))

print(squaresum(10))

print(squaresum(15))

Output:

30

0

285

1015