**Northwestern Polytechnic University**

**Python Programming**

**Homework Assignment #2**

**1. Create a function that takes an integer *m* greater than *1* and returns the largest integer smaller than *m* that evenly divides *m*.**

def lrgst\_factor(m):

for i in range(m-1,0,-1): # range to check factors from largest to smallest

if m % i == 0:

return i

lrgst\_factor (15) # factors are 1, 3, 5 Output: 5

lrgst\_factor (80) # factors are 1, 2, 4, 5, 8, 10, 16, 20, 40 Output: 40

**2. Define a function which takes in a number *m* and determines whether the number is a perfect number. A perfect number is equal to the sum of all its factors. For instance, *6* is a perfect number since *6 = 1 + 2 + 3*.**

def pfct\_num(m):

sum = 0

for i in range(1, m):

if(m % i == 0):

sum = sum + i # To calculate sum of the factors

if (sum == m):

return True

else:

return False

pfct\_num (6) # Output: True

pfct\_num (8) # Output: False

pfct\_num (28) # Output: True

**3. Implement a function to check if the number of digits from two positive input parameters is the same or not.**

def same\_ord(a, b):

count1 = 0

count2 = 0

while a != 0:

a //= 10

count1 += 1

while b !=0:

b//=10

count2 += 1

if (count1==count2):

return True

else:

return False

same\_ord(50, 70) # Output: True

same\_ord(50, 100) # Output: False

same\_ord(1000, 100000) # Output: False

**4. Write a function that takes in a number and determines if the digits contain two adjacent *5s*.**

def two\_5(n): # Return true if n has two fives in a row.

x=str (n)

y="55"

return y in x

two\_5 (5) # Output: False

two\_5 (55) # Output: True

two\_5 (550055) # Output: True

two\_5 (12345) # Output: False

two\_5 (50505050) # Output: False

**5. Design a function that returns the number of unique digits in a positive integer.**

def uniq\_digits(x):

list1=[]

while x != 0:

r=x%10

if r not in list1:

list1.append(r)

x //= 10

return len(list1)

uniq\_digits (8675309) # Output: 7

uniq\_digits (1313131) # Output: 2

uniq\_digits (13173131) # Output: 3

uniq\_digits (10000) # Output: 2

uniq\_digits (101) # Output: 2

uniq\_digits (10) # Output: 2

**6. Write a *def* function *"amc"* with a positive integer *"n"* input parameter. It returns the smallest amicable number greater than *"n".* Two different numbers are both amicable if the sum of the proper divisors of each is equal to the other. Any number that's part of such a pair is an amicable number.**

***Hint: You may want to create a separate function to sum proper divisors.***

def sumOfdiv(a):

sum=0

for i in range(1,a):

if (a%i==0):

sum=sum+i

return sum

def amc(n):

i = 1

while i>0:

sum1 = sumOfdiv (n+i)

if sum1 != n+i:

sum2 = sumOfdiv (sum1)

if sum2 == n+i :

print ( n+i )

break

i += 1

amc (5) # Output: 220

amc(220) # Output: 284

amc(284) # Output: 1184

amc(5000)  # Output: 5020