 **Northwestern Polytechnic University**

**Python Programming**

**Homework Assignment #4**

**Due day: 10/28/2021**

**Instruction:**

1. **Push the source code to GitHub or answer sheet in word file**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue homework submission could not be accepted.**

**4. Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. Write a high order function to make withdrawal from the balance in the bank account by nonlocal variable.

def mk\_wd(x):

global y

y=x

def balance(amount):

global y

y=y-amount

if y<0:

return "Insufficient Funds"

else:

return y

return balance

rem = mk\_wd (100)

rem(10) # 90

rem(20) # 70

rem(100) # ‘Insufficient Funds’

2. Write a function that deletes all instances of an element from a list.

def rm\_all(n, lst):

try:

while True:

lst.remove(n)

except ValueError:

pass

return lst

x = [3, 1, 2, 1, 5, 1, 1, 7]

rm\_all (1, x) # [3,2,5,7]

3. Write a function that takes in three arguments x, elem, and a list, and adds as many "elem"s to the end of the list as there are x’s.

def add\_many(x, elem, lst):

for i in (0,x):

lst.append(elem)

return lst

lst = [1, 2, 4, 2, 1]

add\_many (2, 5, lst)

lst # [1, 2, 4, 2, 1, 5, 5]

4. Write a function to create a new list from given a "suits" list and a number list

def f(suits, numbers):

x=[ [a,b] for a in suits for b in numbers ]

return x

f (['S', 'C'], [1, 2, 3]) # [['S', 1], ['S', 2], ['S', 3], ['C', 1], ['C', 2], ['C', 3]]

f (['S', 'C'], [3, 2, 1]) # [['S', 3], ['S', 2], ['S', 1], ['C', 3], ['C', 2], ['C', 1]]

f ([], [3, 2, 1]) # []

f (['S', 'C'], []) # []

5. Write a function to merge 2 sorted lists a and b, and then return a new list with a sorted order by RECURSIVE calls.

def mrg(ls1, ls2):

if ls1==[]:

return ls2

elif ls2==[]:

return ls1

elif ls1[0]>ls2[0]:

return [ls2[0]]+mrg(ls1,ls2[1:])

else:

return [ls1[0]]+mrg(ls1[1:],ls2)

mrg ([1, 3, 5], [2, 4, 6])

mrg ([], [2, 4, 6])

mrg ([1, 2, 3], [])

mrg ([5, 7], [2, 4, 6])

6. Write the function to flatten the deep list.

def fltn(lst):

if type(lst) != list:

return [lst]

if not lst:

return []

return fltn(lst[0]) + fltn(lst[1:])

print(fltn([1, 2, 3]))

x = [1, [2, 3], 4]

print(fltn (x))

x = [[1, [1, 1]], 1, [1, 1]]

print(fltn (x))

7. Define a function to check if the element exists in the list or not.

def chk\_elm(lst, n):

def flatten(lst):

if type(lst) != list:

return [lst]

if not lst:

return []

return flatten(lst[0]) + flatten(lst[1:])

lst1=flatten(lst)

if n in lst1:

return True

else:

return False

a = [ [1,[2]], 3, [ [4], [5,[6] ] ] ]

chk\_elm (a, 6)

8. Write a function to check whether the input argument list is symmetric or not in recursive call.

def sym(l):

def rev(l):

if len(l) == 0: return []

return [l[-1]] + rev(l[:-1])

if rev(l)==l:

return True

else:

return False

sym ([])

sym ([1])

sym ([1, 4, 5, 1])

sym ([1, 4, 4, 1])

sym (['l', 'o', 'l'])

9. Write a function in recursive call that takes in a list lst, a function g, and an initial value m. This function will fold lst starting at the beginning. If lst is [1, 2, 3, 4, 5] then the function g is applied as follows:

g (g (g (g (g (m, 1), 2), 3), 4), 5)

from operator import add, sub, mul

def fld (lst, g, m):

for x in lst:

m=g(m,x)

return m

s = [3, 2, 1]

fld (s, sub, 0)

fld (s, add, 0)

fld (s, mul, 1)

fld ([], sub, 100)

10. Implement a function to create 2D array as follows

def crte\_2d\_arr(rows, columns):

arr = [[0] \* columns for i in range(rows)]

for i in range(rows):

for j in range(columns):

arr[i][j] = '-'

return arr

crte\_2d\_arr(3, 5)