**PROBLEM #1** Something like anagram but with numbers not letters

**PROBLEM #2**

def parChecker(symbolString):

s = Stack()

balanced = True

index = 0

while index < len(symbolString) and balanced:

symbol = symbolString[index]

if symbol == "(":

s.push(symbol)

else:

if s.isEmpty():

balanced = False

else:

s.pop()

index = index + 1

if balanced and s.isEmpty():

return True

else:

return False

**PROBLEM #3**

**(built-in permutation function in itertools library??)**

def permutation(lst):

    # If lst is empty then there are no permutations

    if len(lst) == 0:

        return []

    # If there is only one element in lst then, only

    # one permuatation is possible

    if len(lst) == 1:

        return [lst]

    # Find the permutations for lst if there are

    # more than 1 characters

    l = [] # empty list that will store current permutation

    # Iterate the input(lst) and calculate the permutation

    for i in range(len(lst)):

       m = lst[i]

       # Extract lst[i] or m from the list.  remLst is

       # remaining list

       remLst = lst[:i] + lst[i+1:]

       # Generating all permutations where m is first

       # element

       for p in permutation(remLst):

           l.append([m] + p)

    return l

or

**def** permutations(word):

**if** len(word)&lt;=1:

**return** [word]

*#get all permutations of length N-1*

perms=permutations(word[1:])

char=word[0]

result=[]

*#iterate over all permutations of length N-1*

**for** perm **in** perms:

*#insert the character into every possible location*

**for** i **in** range(len(perm)+1):

result.append(perm[:i] + char + perm[i:])

**return** result