Programming Problem. Modify BreakpointReversalSort.py as follows:

The given version of the code outputs only one of many possible solutions. The way to generate multiple solutions should be that if at any stage of the program, there is more than one reversal that removes two breakpoints, progam should accept all such reversals and output all solutions. Turn in Your listing for following inputs:

0 1 2 10 9 3 4 7 6 5 8
0 9 2 1 6 8 7 5 3 4 10

Solution:

*The given code works iteratively (linear fashion) such that it chooses one single reversal at each step, applies that reversal to the sequence. It takes this kind of steps until sequence is sorted. If there is more than one equally good reversal then given code just picks the first one. But we want to learn what would be the solution if we take apply each equally good reversal at each step. So execution would be more like a tree rather than linear. For this purpose we make two changes:

1. We make `pickReversal` function return return list of reversals rather than a single reversal

2. we make `improvedBreakpointReversalSort` function recursive to traverse all solutions in a depth-first fastion

PYTHON CODE

```
import random

def makePermutation(n):
    """ generates a random permuation of the numbers 1..n-1 sandwiched between 0
and n """
    seq = range(1,n)
    random.shuffle(seq)
    return [0] + seq + [n]

def hasBreakpoints(seq):
    """ returns True if sequnces in not strictly increasing by 1 """
    for i in xrange(1, len(seq)):
        if (seq[i] != seq[i-1] + 1):
            return True
    return False

def getStrips(seq):
    """ find contained intervals where sequence is ordered, and return intervals
    in as lists, increasing and decreasing. Single elements are considered
    decreasing. "Contained" excludes the first and last interval. """
    deltas = [seq[i+1] - seq[i] for i in xrange(len(seq)-1)]
    increasing = list()
    decreasing = list()
    start = 0
    for i, diff in enumerate(deltas):
```

```
        if (abs(diff) == 1) and (diff == deltas[start]):
            continue
        if (start > 0):
            if deltas[start] == 1:
                increasing.append((start, i+1))
            else:
                decreasing.append((start, i+1))
        start = i+1
    return increasing, decreasing

def pickReversal(seq, decreasing):
    """ test each decreasing interval to see if it leads to a reversal that
    removes two breakpoints, otherwise, return a reversal that removes only one
"""
    reversals = list()
    IntervalStarts = [i for i, j in decreasing]
    for i, j in decreasing:
        endValue = seq[j-1]                     # ending value of decreasing
interval
        predIndex = seq.index(endValue-1)       # index of endValue's predecessor
        k = predIndex+1                         # index of value following
predecessor
        if (predIndex in IntervalStarts):       # indirectly verifies that
predcessor
            continue                            # is at the end of an increasing
interval
        if (j > k):
            if (seq[k] + 1 == seq[j]):
                print "2:",
                return (k, j)                   # if reversal removes two
breakpoints, do it add to reversal list
                reversals.append((k,j))
        else:
            if (seq[j] + 1 == seq[k]):
                print "2:",
                return (j, k)                   # if reversal removes two
breakpoints, do it add to reversal list
                reversals.append((k,j))

    if (j > k):
        j, k = k,j
    print "1:",
    return (j, k)                               # otherwise, settle for removing
only one
    if len(list()) == 0:   #If list is empty -> no reversal removing 2 bp. Settle
for one
        return [(j, k)]
    else:
        return reversals

def doReversal(seq,(i,j)):
    return seq[:i] + [element for element in reversed(seq[i:j])] + seq[j:]

def improvedBreakpointReversalSort(seq):
    while hasBreakpoints(seq):
    if hasBreakpoints(seq): #recursive case
```

```
        increasing, decreasing = getStrips(seq)
        if len(decreasing) > 0:
            reversals = pickReversal(seq, decreasing)
        else:
            print "0:",
            reversals = [increasing[0]]
        print seq, "reversals", reversals
        seq = doReversal(seq,reversal)

        for reversal in reversals: #For each reversal, apply reversal and call
improvedBreakpointReversalSort so that we go through each possibility by  depth-
first manner.
            seq2 = doReversal(seq,reversal)
            improvedBreakpointReversalSort(seq2)
    else
        print seq, "Sorted" #base case
    return

if __name__ == "__main__":
    print "Python implementation of breakpoint reversal sort on page 135"

    while True:
        input = raw_input('Enter a list, the size of list, or 0 to exit:')
        if (input.find(',') > 0):
            L = [int(v) for v in input.split(',')]
        else:
            n = int(input)
            if (n == 0):
                break
            L = makePermutation(n)
        improvedBreakpointReversalSort(L)
```