Group work on divide and conquer
CS24, Dr. Ostheimer

(1) Here is a program that searches for a number x in a list a of numbers, returning **True** if x is in a and **False** if not.

```
def linearSearch(a, x):
  if len(a) == 0:
    answer = False
  else:
    if a[0] == x:
      answer = True
    else:
      answer = linearSearch (a[1:], x)
  return answer
```

Let $f(n)$ be the number of comparisons performed (in the worst case) if the length of a is n.

(a) What is $f(0)$?
(b) Find a recursive formula for $f(n)$ if $n > 0$.
(c) Find a big-O estimate for f.

(2) Here is another program that searches for a number x in a list a of numbers, returning **True** if x is in a and **False** if not.

```
def binarySearch(a, x):
  # a is sorted from least to greatest
  if len(a) == 0:
    answer = False
  else:
    m = len(a)/2
    if a[m] == x:
      answer = True
    else:
      if a[m] < x:
        answer = binarySearch(a[m+1:], x)
      else:
        answer = binarySearch(a[0:m], x)
  return answer
```

Let $f(n)$ be the number of comparisons performed (in the worst case) if the length of a is n.

(a) What is $f(0)$?
(b) Find a recursive formula for $f(n)$ if $n > 0$.
(c) Find a big-O estimate for f.

(3) Here is a program to merge two sorted lists.

```
def mergeRec(list1, list2):
   if len(list1) == 0:
      bigList = list2
   else:
      if len(list2) == 0:
         bigList = list1
      else:
         if list1[0] < list2[0]:
            bigList = [list1[0]] + mergeRec(list1[1:], list2)
         else:
            bigList = [list2[0]] + mergeRec(list1, list2[1:])
   return bigList
```

Find a big-O estimate for the time complexity.

(4) Here is a program to sort a list. Note that it calls **mergeRec** above.

```
def mergeSort(list):
   if len(list) <= 1:
      answer = list
   else:
      m = len(list)/2
      list1 = mergeSort(list[0:m])
      list2 = mergeSort(list[m:])
      answer = mergeRec(list1, list2)
   return answer
```

Find a big-O estimate for the time complexity.