Part 1

1. (4 points) What, if anything, is wrong with the following method that attempts to determine the number of items in L that are multiples of n:

```python
def multiple(L, n):
    if len(L) == 0:
        return 0
    if L[0] % n == 0:
        return 1 + multiple(L[1:], n)
    return multiple(L, n)
```

Answer: Recursive calls are not made to a simpler instance of the problem.

2. (4 points) What, if anything, is wrong with the following method that attempts to determine if list L is sorted:

```python
def isSorted(L):
    if L[0] > L[1]:
        return False
    if len(L) < 2:
        return True
    return isSorted(L[1:])
```

Answer: Inappropriate base case - it does not process correctly the simplest possible instance of the problem (for example when L is empty).

3. (4 points) What, if anything, is wrong with the following method that attempts to determine if string S is a palindrome:

```python
def palindrome(S):
    if len(S) < 2:
        return True
    if S[0] != S[-1]:
        return False
    palindrome(S[1:-1])
```

Answer: It breaks the design rule: even if the recursive calls work properly, the original call does not work.

4. (5 points) The running time of a method that prints the smallest element in a SORTED list is (mark all that apply):
Answer: $O(n), O(\log n), O(1), \Omega(1), \Theta(1)$

5. (5 points) The running time of Towers of Hanoi is (mark all that apply):
Answer: $\Omega(n^2), \Omega(n\log n), \Omega(n)$

6. (2 points) Determine the big-O running time of the following function:

```python
def p1(A):
    print(A)
```

Answer: $O(n)$

7. (2 points) Determine the big-O running time of the following function:

```python
def p1(A):
    for a in A:
        print(a)
    for b in A:
        for c in A:
            print(b, c)
```
Answer: $O(n^2)$

8. (2 points) Determine the big-O running time of the following function:

```python
def p2(A):
  for a in A:
    print(a)
  i = len(A)-1
  while i>0:
    print(A[i])
    i = i//2
```

Answer: $O(n)$

9. (2 points) Determine the big-O running time of the following function:

```python
def p3(A):
  for a in A:
    print(a)
  i = len(A)-1
  while i>0:
    print(A[i])
    i = i//2
```

Answer: $O(n \log n)$

10. (2 points) Determine the big-O running time of the following function:

```python
def p8(a):
  i = len(a)-1
  while i>=0:
    print(a[i])
    i -= 2302
```

Answer: $O(n)$

11. (2 points) Determine the big-O running time of the following function:

```python
def p4(A):
  for i in range(len(A)):
    print(A[:i])
```

Answer: $O(n^2)$

12. (2 points) Determine the big-O running time of the following function:

```python
def p10(a):
  i =1
  while i<len(a):
    for t in a:
      print(t)
    i = i*2
```

Answer: $O(n \log n)$

13. (2 points) Determine the big-O running time of the following function:

```python
def p10(a):
  i = 1024
  while i>0:
    print(a[0])
    i = i//2
```
14. (2 points) What is the recurrence equation that describes the running time of the following function?

```python
def r0(a):
    if len(a)>0:
        print(a[0])
        r0(a[1:])
```

Answer: \( T(n) = T(n-1) + 1 \)

15. (2 points) What is the recurrence equation that describes the running time of the following function?

```python
def r1(a):
    if len(a)<2:
        print(a)
    else:
        mid = len(a)//2
        r1(a[mid:])
        print(a[mid])
```

Answer: \( T(n) = T(n/2) + 1 \)

16. (2 points) What is the recurrence equation that describes the running time of the following function?

```python
def r2(a):
    if len(a)<2:
        print(a)
    else:
        mid = len(a)//2
        r2(a[mid:])
        for i in range(len(a)):
            print(a[i],end=' ')
        r2(a[:mid])
```

Answer: \( T(n) = 2T(n/2) + n \)

17. (2 points) What is the recurrence equation that describes the running time of the following function?

```python
def r3(a):
    if len(a)<2:
        print(a)
    else:
        mid = len(a)//2
        for i in range(len(a)):
            print(a[i],end=' ')
        for i in range(4):
            r3(a[mid:])
```

Answer: \( T(n) = 4T(n/2) + n \)

18. (2 points) What is the recurrence equation that describes the running time of the following function?

```python
def r4(a):
    if len(a)<2:
        print(a)
    else:
        mid = len(a)//2
        for i in range(len(a)):
            for j in range(len(a)):
                print(a[i],end=' ')
        for i in range(2):
            r4(a[mid:]
```
Answer: \( T(n) = 2T(n/2) + n^2 \)

19. (2 points) What is the recurrence equation that describes the running time of the following function?

```python
def r5(a):
    if len(a)>0:
        print(a[0])
        r5(a[1:])
        r5(a[:-2])
```

Answer: \( T(n) = 2T(n-1) + 1 \)

20. (2 points) What is the solution to the recurrence \( T(n) = 2T(n-1) + 1 \)?

Answer: \( O(2^n) \)

21. (2 points) What is the solution to the recurrence \( T(n) = T(n-1) + n \)?

Answer: \( O(n^2) \)

22. (2 points) What is the solution to the recurrence \( T(n) = 2T(n/2) + n \)?

Answer: \( O(n\log n) \)

23. (2 points) What is the solution to the recurrence \( T(n) = 2T(n/2) + n^2 \)?

Answer: \( O(n^2) \)

24. (2 points) What is the solution to the recurrence \( T(n) = 8T(n/2) + n \)?

Answer: \( O(n^3) \)

25. (4 points) Consider the program below.

```python
1 def writeBinary(bitsLeft,stringSoFar):
  2   if bitsLeft>0:
  3       writeBinary(bitsLeft-1,stringSoFar+'0')
  4       writeBinary(bitsLeft-1,stringSoFar+'1')
  5   else:
  6       print(stringSoFar)
  7   writeBinary(3,'')
```

When the first output is produced, the stack of activation records contains the following:

- [writeBinary, ip=6, bitsLeft=0, stringSoFar = '000']
- [writeBinary, ip=3, bitsLeft=1, stringSoFar = '00']
- [writeBinary, ip=3, bitsLeft=2, stringSoFar = '0']
- [writeBinary, ip=3, bitsLeft=3, stringSoFar = '']
- [main, ip= 7]

What are the contents of the stack when the second output is produced? Answer:

- [writeBinary, ip=6, bitsLeft=0, stringSoFar = '001']
- [writeBinary, ip=4, bitsLeft=1, stringSoFar = '00']
- [writeBinary, ip=3, bitsLeft=2, stringSoFar = '0']
- [writeBinary, ip=3, bitsLeft=3, stringSoFar = '']
- [main, ip= 7]

26. (4 points) Consider the program below.

```python
1 def writeBinary(bitsLeft,stringSoFar):
  2   if bitsLeft>0:
  3       writeBinary(bitsLeft-1,stringSoFar+'0')
  4       writeBinary(bitsLeft-1,stringSoFar+'1')
  5   else:
  6       print(stringSoFar)
  7   writeBinary(3,'')
```
At some point during execution, the stack contains the following:

[writeBinary, ip=4, bitsLeft=2, stringSoFar = '1']
[writeBinary, ip=4, bitsLeft=3, stringSoFar = '']
[main, ip= 7]

What is the next output that will be produced?

Answer: 110

Part 2

1. Write the recursive function allEqual(L1) that receives a (native) list L and determines if all its elements are equal. For example, allEqual([]), allEqual([2302]), and allEqual([1,1,1,1,1]) should return True and allEqual([2302,2]) and allEqual([1,3,1,1,2,1]) should return False.

2. Write the recursive function multiples(L,n) that receives a (native) list L and an integer n and returns a list containing the elements of L that are multiples of n, in the same order that they appear in L. For example, multiples([1,2,3,4,5,6,7,8,9,10],2) should return the list [2, 4, 6, 8, 10], multiples([1,2,3,4,5,6,7,8,9,10],5) should return the list [5, 10] and multiples([3,6,9,12],5) should return [].

3. Write the function sumFirstN(L,n) that receives a reference-based list L and and integer n and returns the sum of the first n elements in the list, or the sum of the whole list if it has less that n elements. For example, if L = BuildList([2,4,6]), sumFirstN(L,0) should return 0, sumFirstN(L,2) should return 6 and sumFirstN(L,20) should return 12.

4. Write the function sumLast2(L) that receives a reference-based list L and returns the sum of the last two elements in L, or the sum of the whole list if it has less that 2 elements. For example, if L = BuildList([2,4,6]), sumLast2(L) should return 10, if L = BuildList([9]), sumLast2(L) should return 9, and if L = BuildList([]), sumLast2(L) should return 0.

Answers:

class Node(object):
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

class List(object):
    def __init__(self):
        self.head = None
        self.tail = None

def Append(L,x):
    if L.head is None:
        L.head = Node(x)
        L.tail = L.head
    else:
        L.tail.next = Node(x)
        L.tail = L.tail.next

def BuildList(pList):
    L = List()
    for d in pList:
        Append(L,d)
    return L

def allEqual(L):
    if len(L) < 2:
        return True
    if L[0] != L[1]:
        return False
return allEqual(L[1:])

def multiples(L, n):
    if len(L) == 0:
        return []
    c = multiples(L[:-1], n)
    if L[-1] % n == 0:
        c.append(L[-1])
    return c

def sumFirstN(L, n):
    s = 0
    t = L.head
    count = 0
    while t is not None and count < n:
        s += t.data
        t = t.next
        count += 1
    return s

def sumLast2(L):
    t = L.head
    if t is None:
        return 0
    if t.next is None:
        return t.data
    while t.next != L.tail:
        t = t.next
    return t.data + L.tail.data

if __name__ == '__main__':
    print('Testing allEqual(L)')
    print(allEqual([]))
    print(allEqual([2]))
    print(allEqual([2, 2, 2, 2, 2, 2, 2, 2]))
    print(allEqual([2, 2, 2, 12, 2, 2, 2, 2]))
    print(allEqual([2, 21, 2, 2, 2, 2, 2, 2]))

    print('Testing multiples(L, n)')
    print(multiples([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 2))
    print(multiples([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 5))
    print(multiples([], 2))
    print(multiples([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 9))
    print(multiples([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 11))

    print('Testing sumFirstN(L, n)')
    L0 = BuildList([])
    L1 = BuildList([2, 4, 6, 8, 10])
    print(sumFirstN(L0, 2))
    print(sumFirstN(L1, 0))
    print(sumFirstN(L1, 2))
    print(sumFirstN(L1, 4))
    print(sumFirstN(L1, 6))

    print('Testing sumLast2(L)')
    L0 = BuildList([])
    L1 = BuildList([2])
    L2 = BuildList([2, 4])
    L3 = BuildList([2, 4, 6, 8])
    L4 = BuildList([2, 4, 6, 8, 10])
    print(sumLast2(L0))
print(sumLast2(L1))
print(sumLast2(L2))
print(sumLast2(L3))
print(sumLast2(L4))