1. Give the Θ for the following and justify your answer:
   (a) $5x^2 + 4x + 3$

   (b) $2^n + n!$

   (c) $n^2 + 2^n$

   (d) $\log(n) + n$

   (e) $\log(n!)$

2. Give a closed form for the following, then give the Θ

   (a) $a_0 = 5$
   $a_n = 3a_{n-1}$

   (b) $a_4 = 2$
   $a_n = a_{n-1} + \log_2(n)$

   (c) $a_1 = 1$
   $a_n = 2a_{n-2} + 1$

   (d) $T(1) = 1$
   $T(k) = 3T(k/2) + 1$

   (e) $T(1) = 4$
   $T(k) = T(k/3) + 4$

3. (Extra Credit):

   $T(0) = 1$
   $T(k) = 3T(k-2) + 4(k-2) + 2$

4. Prove **theorem 2:** $x^k \in \mathbf{O}(x^{k+c})$

5. Prove **theorem 3:** $x^k + c \cdot x^{k-r} \in \mathbf{O}(x^k)$

6. Prove **theorem 5:** if $f(n) \in \mathbf{O}(g(n))$ and $g(n) \in \mathbf{O}(h(n))$, then $f(n) \in \mathbf{O}(h(n))$

7. Give the Θ running time for the following selection sort algorithm

```
def selSort(l):
    for i in range(len(l)):
        min = l[i]
        minI = i
        for j in range(i,len(l)):
            if l[j] < min:
                minI = j
                min = l[j]
            #end if
        # end for
        (l[i], min) = (min, l[i])
    # end for
```

8. Give the recurrence relation for badSort. Remember `l[a:b]` copies the elements from `l[a]` to `l[b]`, so even though it's an expression `l[a:b]` runs in $n - 2$ time.

```
def badSort(l): n = len(l)

    if n == 1:
        return l

    first = badSort(l[0:n-2])
    middle = badSort(l[1:n-1])
    end = badSort(l[2:n])

    return [first[0]] + middle + [end[n-1]]
```

9. The following algorithm is the merge sort we way in class

```
def merge(low, high):
    i = 0
    j = 0
    merged = []
    while i < len(low) and j < len(high):
        if low[i] < high[j]:
            merged += [low[i]]
            i += 1
        else:
            merged += [high[j]]
            j += 1
    return merged + low[i:] + high[j:]
```

```
def mergeSort(lst):
    n = len(lst)
    n2 = int(n/2)

    # base case:
    if n <= 1:
        return lst

    # recursive case:
    low = mergeSort(lst[0:n2])
    high = mergeSort(lst[n2:n])
    lst = merge(low,high)

    return lst
```

(a) Give the Θ running time for merge.
hint: what is the input size for merge?

(b) Use (a) to give a recurrence relation for the running time of mergeSort.

(c) Solve the recurrence to get a Θ running time for mergesort.