1. Give the Θ for the following and justify your answer:

   (a) $5n^2 + 4n + 3$

   $f(n) \in \Theta(n^2)$ by theorem 3.

   (b) $2^n + n!$

   $f(n) \in \Theta(n!)$, since $2^n \in \mathcal{O}(n!)$

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

$$
\begin{aligned}
2^n + n^2 > 2^n &\in \Omega(2^n) \\
2^n + n^2 < 2^n + 2^n \in O(2^n) \implies f(n) &\in \Theta(2^n)
\end{aligned}
$$

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

$$
\begin{aligned}
\log(n) + n > n &\in \Omega(n) \\
\log(n) + n < n + n \in O(n) \implies f(n) &\in \Theta(n)
\end{aligned}
$$

   (e) $\log(n!)$

$$\log(n!) = \log\left(\prod_{i=1}^{n} k_i\right) = \sum_{k=1}^{n} \log(k) \in \Theta(n \log n)$$

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

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

$$
\begin{aligned}
&= 5 + 3(5) + 3(3(5)5) + 3(3(3(15))) \\
&= \sum_{k=0}^{n} 3^k(5) = \frac{5(3^{n+1} - 1)}{2} \in \Theta(3^n)
\end{aligned}
$$

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

$$
\begin{aligned}
&= 2 + (2 + \log_2(5)) + (2 + (2 + \log_2(6))) \\
&= \sum_{k=5}^{n} 2 + (2 + \log_2(k)) \\
&= \sum_{k=5}^{n} 2 + \sum_{k=5}^{n} 2 + \log_2(k) \\
&= \sum_{k=5}^{n} 2 + \sum_{k=5}^{n} 2 + \sum_{k=5}^{n} \log_2(k) \\
&= 2\sum_{k=5}^{n} 2 + \sum_{k=5}^{n} \log_2(k) \\
&= 4(n-4) + \log_2\left(\prod_{i=5}^{n} k_i\right) \\
&= 4(n-4) + \log_2(n!) \in \Theta(n\log_2((n)))
\end{aligned}
$$

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

$$
\begin{aligned}
&= 1 + (2(1) + 1) + (2(3) + 1) + (2(7) + 1) \\
&= \sum_{k=0}^{n} 2k + 1 \Longleftarrow k\%2 = 1 \qquad \text{i.e., sum over odd indices} \\
&= \sum_{i=1}^{n} 1 + \sum_{i=1}^{n} 2k + 1 \\
&= n + n(n+1) \\
&= 2n + n^2 \in \Theta(n^2)
\end{aligned}
$$

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

$$a = 3,\, b = 2,\, k = \log_2(n),\, f(n) = 1$$

$$\begin{aligned}
\implies T(n) &= 3^k + \sum_{i=0}^{k-1} 3^i(1) \\
&= 3^k + \sum_{i=0}^{k-1} 3^i \\
&= 3^k + \frac{3^k - 1}{2} \\
&= 3^{\log_2(n)} + \frac{3^{\log_2(n)} - 1}{2} \in \Theta\left(n^{\log_2(n)}\right)
\end{aligned}$$

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

$$a = 1,\, b = 3,\, k = \log_3(n),\, f(n) = 4$$

$$\begin{aligned}
\implies T(n) &= (1)(4) + \sum_{i=0}^{k-1}(1)(4) \\
&= 4 + \sum_{i=0}^{k-1} 4 \\
&= 4 + 4(k-1) \\
&= 4\log_3(n) \in \Theta(\log_3(n))
\end{aligned}$$

3. (Extra Credit):
   $T(0) = 1$
   $T(n) = 3T(n-2) + 4(n-2) + 2$

$$
\begin{aligned}
a = 3,\, b = 2,\, k = \log_2(n),\, f(n) &= 4(n-2) + 2 \\
\implies T(n) &= 3^k + \sum_{i=0}^{k-1} 3^i \frac{4n}{2^i} - 6 \\
&= 3^k + 4n \sum_{i=0}^{k-1} \left(\frac{3}{2}\right)^i - \sum_{i=0}^{k-1} 6 \\
&= 3^k + \frac{4n\left(\frac{3}{2}^k\right) - 1}{\frac{3}{2} - 1} - 6k - 6 \\
&= 3^{\log_2(n)} + \frac{4n\left(\frac{3}{2}^{\log_2(n)}\right) - 1}{\frac{3}{2} - 1} - 6\log_2(n) - 6 \\
&\in \Theta\left(n^{\log_2(n)}\right)
\end{aligned}
$$

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

   Proof.
   Let $n_0 = 1 \implies f(n_0^{k+c}) \leq f(n_0^{k+c})$

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

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

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
```

$\in \Theta(n^2)$, since there is a double for-loop, which dominants other factors.

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.