Problem 1

Give The Θ for the following.

Justify your answer.

$$f(n) \in \Theta(g(n)) \iff n_0 \in \mathbf{N},\ \forall n > n_0,\ \exists c_1, c_2 \in \mathbf{R}\colon c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$$

$$\begin{aligned}
f(n) &\in \Theta(g(n)) & c_1 && c_2 && (1)\\
5x^2 + 4x + 3 &\in \Theta(x^2) & 1 && 12 && (2)\\
2^n + n! &\in \Theta(n!) & 1 && 3 && (3)\\
n^2 + 2^n &\in \Theta(2^n) & 1 && 3 && (4)\\
\log n + n &\in \Theta(n) & 1 && 2 && (5)
\end{aligned}$$

The last one is the only one that cannot be verified by setting up a simple inequality.
Upper Bound:

$$\begin{aligned}
\log n! &= \log 1 + \log 2 + \log 3 + \cdots + \log n\\
&= \sum_{i=1}^{n} \log i\\
&\le n \log n && \text{There are } n \log x \text{ and each } \log x \text{ is } \le \log n\\
c_2 &= 1
\end{aligned}$$

Lower Bound:

$$\begin{aligned}
&= \sum_{i=1}^{n} \log i\\
&\ge \sum_{i=\frac{n}{2}}^{n} \log i && \text{chop off the first half of the sum}\\
&\ge \sum_{i=\frac{n}{2}}^{n} \log \frac{n}{2} && \frac{n}{2} \le i \text{ always}\\
&\ge \frac{n}{2} \log \frac{n}{2} && \text{reduce sum}\\
c_1 &= \frac{1}{2}
\end{aligned}$$

Problem 2

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

Master Theorem:

$$\begin{aligned}
T(n) &= aT\left(\frac{n}{b}\right) + f(n) \\
&= a^k \cdot T(1) + \sum_{i=0}^{k-1} a^i \cdot f\left(\frac{n}{b^i}\right)
\end{aligned}$$

$$T(n) \in \begin{cases} \Theta(n^{\log_b a}) & f(n) \in \mathbf{O}(n^{\log_b a}) \\ \Theta(n^{\log_b a} \cdot \log n) & f(n) \in \Theta(n^{\log_b a}) \\ \Theta(f(n)) & f(n) \in \Omega(n^{\log_b a}) \end{cases}$$

(a)

$$\begin{aligned}
a_0 &= 5 \\
a_n &= 3a_{n-1} \\
&= 3(3a_{n-2}) \\
&= 3(3(3a_{n-3})) \\
&= 3^n(a_0) \\
&= 3^n \cdot 5 \in \Theta(3^n)
\end{aligned}$$

(b)

$$\begin{aligned}
a_4 &= 2 \\
a_n &= a_{n-1} + \log_2(n) \\
&= a_{n-2} + \log_2(n-1) + \log_2(n) \\
&= a_{n-3} + \log_2(n-2) + \log_2(n-1) + \log_2(n) \\
&= a_4 + \log_2\left(\frac{n!}{4!}\right) \\
&= 2 + \log_2 \frac{n!}{24} \in \Theta(n \log n)
\end{aligned}$$

(c)

$$\begin{aligned}
a_1 &= 1 \\
a_n &= 2a_{n-2} + 1 \\
&= 2(2a_{n-4} + 1) + 1 \\
&= 2(2(2a_{n-6} + 1) + 1) + 1 \\
&= 2^{\frac{n-1}{2}} + \sum_{i=1}^{\frac{n-1}{2}} 2^{i-1} \\
&= a_1 \cdot 2^{\frac{n+1}{2}} \\
&= 2^{\frac{n+1}{2}} \in \Theta(2^n)
\end{aligned}$$

Problem 2 continued on next page. . .

(d)

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

(e)

$$
\begin{aligned}
T(1) &= 4 \\
T(n) &= T\left(\frac{k}{3}\right) + 4 \\
a &= 1 \\
b &= 3 \\
k &= \log_3 n \\
f(n) &= 4 \\
T(n) &= 1^k 4 + \sum_{i=1}^{\log_3 n-1} 1^i 4 \\
&= 4 + 4(\log_3 n - 1) \in \Theta(\log n)
\end{aligned}
$$

(f)

$$
\begin{aligned}
T(n) &= 3T(n-2) + 4(n-2) + 2 \\
&= 3(3T(n-4) + 4(n-4) + 2) + 4(n-2) + 2 \\
&= 3(3(3T(n-6) + 4(n-6) + 2) + 4(n-4) + 2) + 4(n-2) + 2 \\
&= 3^{\frac{n}{2}} + \sum_{i=1}^{\frac{n}{2}} 3^{\frac{n}{2}-i} \cdot (4(2i-2) + 2) \\
&= 3^{\frac{n}{2}+1} - 2n + 3^{\frac{n}{2}} - 3 \in \Theta(3^n)
\end{aligned}
$$

Problem 3

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

Let $c_1 = 1$ and assuming $k \in \mathbf{N}$
If $x^k \leq c_1 \cdot x^{k+c}$ then $x^k \in \mathbf{O}(x^{k+c})$
$\frac{x^k}{x^{k+c}} \leq c_1$
$\frac{1}{x^c} \leq c_1$ is true for all positive c and $x > k$.

Problem 4

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

Assuming $k, r, x \in \mathbf{N}$
Let $c_1 = 1 + c$
If $x^k + c \cdot x^{k-r} \leq c_1(x^k)$ for some c_1 then $x^k + c \cdot x^{k-r} \in \mathbf{O}(x^k)$
$1 + c \cdot x^{-r} \leq c_1$
$\frac{1}{x^r} \leq 1$ will always be true.

Problem 5

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

If $f(n) \in \mathbf{O}(g(n))$ then there exists a c_1 such that $f(n) \leq c_1 \cdot g(n)$
and if $g(n) \in \mathbf{O}(h(n))$ then there exists a c_2 such that $g(n) \leq c_2 \cdot h(n)$
Let $c_3 = c_1 \cdot c_2$. Then $f(n) \leq c_1 g(n) \leq c_3 h(n)$
Due to the transitivity of $\leq$, it must be that $f(n) \in \mathbf{O}(h(n))$

Problem 6

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
```

For the first element you have to compare it with $n-1$ items.
For the second element you have to compare with $n-2$ items.
The total number of compares is $\sum_{i=1}^{n} n - i$ or $\frac{n^2-n}{2} \in \Theta(n^2)$

Problem 7

```
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]]
```

(a) Give the recurrence relation for badSort
remember `l[a:b]` copies the elements from `l[a]` to `l[b]`

Assuming no copies are made on the return (I'm sure there are but I have no idea how the python list is implemented) each recursive call only takes 2 items off the length and 3 calls are made.
$T(n) = 3T(n-2)$ or $T(n) = 3^{n-1}2$

(b) Give the Θ for badSort

$T(n) \in \Theta(3^n)$

Problem 8

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 our list is 0, or 1 element, the its sorted
    if n <= 1:
        return lst
```

Problem 8 continued on next page...

```
    # recursive case: split the list in half
    # sort the halves
    # merge the lists together
    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?

Merge does one compare and one move for each of the items in the two inputs.
This means merge is linear time $\in \Theta(n)$

(b) Use part 1 to give a recurrence relation for the running time of mergeSort

$T(n) = 2T(\frac{n}{2}) + n$

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

$$\begin{aligned}
a &= 2 \\
b &= 2 \\
f(n) &= n \\
k &= \log_2 n \\
T(n) &= 2^{\log_2 n} \cdot 1 + \sum_{i=0}^{\log_2 n - 1} 2^i \cdot \frac{n}{2^i} \\
&= n + n \cdot \log_2 n \\
&\text{master theorem:} \\
f(n) &\in \Theta(n^{\log_2 2}) \\
T(n) &\in \Theta(n^{\log_2 2} \cdot \log n) \\
T(n) &\in \Theta(n \log n)
\end{aligned}$$