Problem 1 a

We can prove that $10n^3 - 3n^2 + 5n \in \Theta\left(n^3\right)$ by taking the limit.

$$\begin{aligned}
&\lim_{n\to\infty} \frac{10n^3 - 3n^2 + 5n}{n^3} \\
=&\lim_{n\to\infty} 10 - \frac{3}{n} + \frac{5}{n^2} \\
=&10
\end{aligned}$$

The ratio $\frac{f(n)}{g(g)}$ where $f(n) = 10n^3 - 3n^2 + 5n$ and $g(n) = n^3$ approaches a constant, this is sufficient to prove that $10n^3 - 3n^2 + 5n \in \Theta\left(n^3\right)$.

Problem 1 b

We will disprove using limits

$$\begin{aligned}
&\lim_{n\to\infty} \frac{n^n}{2^{2n\log n}} \\
=&0
\end{aligned}$$

Because the limit is zero, we can determine that $2^{2n\log n}$ grows faster than n^n. Therefore, $n^n \notin \Theta\left(2^{2n\log n}\right)$.

Problem 1 c

We will prove using limits

$$\begin{aligned}
&\lim_{n\to\infty} \frac{9\log n^3 + \log\frac{n^2}{4}}{\log n} \\
=&\lim_{n\to\infty} \frac{9\log n^3}{\log n} + 2 \\
=&\lim_{n\to\infty} 9\cdot 3 + 2 \\
=&29
\end{aligned}$$

Because the limit is a constant, we can say that $\log n$ and $9\log n^3 + \log\frac{n^2}{4}$ have the same order of growth. Therefore, $9\log n^3 + \log\frac{n^2}{4} \in \Theta\left(\log n\right)$.

Problem 2

We can solve running-time of this program by solving the Sigma-expression.

$$\sum_{i=1}^{2n}\sum_{j=1}^{i+1}\sum_{k=1}^{2j} 1$$

$$\sum_{i=1}^{2n} \sum_{j=1}^{i+1} 2j$$

$$\sum_{i=1}^{2n} i^2 + 3i + 2$$

$$\frac{8n^3 + 24n^2 + 22n}{3}$$

The highest-order term, $\frac{8n^3}{3}$ belongs to $\mathcal{O}\left(n^3\right)$. Therefore, the running-time of the presented algorithm is $\mathcal{O}\left(n^3\right)$.

Problem 3

Each recursive call to $isPalindrome$ performs $\mathcal{O}(1)$ work and calls itself exactly once with the input string's length reduced by two. We can establish the following recurrence relation.

$$T(n) = T(n-2) + 1$$

Using the condition $T(0) = 1$, we get a running-time of $n + 1$. The order of growth is therefore $\mathcal{O}(n)$.

Problem 4 a

By substituting values for n, we get the following

$$T(1) = 0$$
$$T(2) = 2$$
$$T(3) = 6$$
$$T(4) = 14$$
$$T(5) = 30$$

The solution to the recurrence relation is $2^n - 2$, which is in $\mathcal{O}(2^n)$.

Problem 4 b

We can use the master theorem to solve for the order of growth. $log_4 4 = 1$ which is equal to d, which is also 1. This is therefore case two of the master theorem. Case two says that the complexity of the algorithm will be $\mathcal{O}\left(n^d \log n\right)$, substituting our d, we get $\mathcal{O}(n \log n)$.

Problem 5

This can be done by using a mechanism similar to a shift register. we begin with $a = 0, i = 0$. We then add the next m numbers in the list to a. If $a = s$, we output i (which is zero). If not, we subtract $X[i]$ from a, increment i, and then add $X[i + m]$ to a. This is now the sum from $X[1]$ to $X[m + 1]$. We continue to "slide" this window across the array until we find an i which makes $a = s$ or $i + m > n$, at which point no i exists.

The summation for this loop would be:

$$\sum_{i=0}^{n-m} 1 = -m + n + 1$$

Since $m < n$, this algorithm is $\mathcal{O}(n)$.

Problem 6

We can use the "Median of Medians" process to determine the k'th smallest element of an array. We will implement quick select, but use the Median of Medians algorithm to select the pivot. This will ensure $\mathcal{O}(n)$ time. Start by splitting the array into two distinct sets. The will have sizes of m and $m - n$ respectively. The two sets must obey the following property: The largest element of the first set is smaller than the smallest element of the second. This will mean that the solution to the problem will be the k'th smallest element of the first list. To separate the two lists, we can simply select an element and put everything smaller than it in the first set, leaving the remaining items in the second set. We select the pivot element by using MID. We start by splitting the array into partitions of 5 elements each. This gives us $\frac{n}{5}$ partitions. Compute the median on each of these partitions. We can then compute the median of these medians. So far, this has taken $n + \frac{n}{5}$. This median of medians is our pivot. Since it is a median of medians, and we started with groups of five, each median is greater than or equal to three other elements in it's own partition. Further, the median we selected is greater than or equal to the medians of at least $\frac{3n}{10}$ groups. If we repeat this process recursively on the smaller set, we get the following recurrence relation.

$$T(n) = T(n/5) + T(7n/10) + n$$

In order to solve this by back-substitution, we should pick a large base case

$$T(n) \leq n \iff n \geq 1000$$

$$T(n) \leq T(n/5) + T(7n/10) + c \cdot n$$

$$\begin{aligned} T(n) &= T(n/5) + T(7n/10) + c \cdot n \\ &\leq T(9n/10) + c \cdot n \end{aligned}$$

Because the algorithm does linear work at each step, and the input size decrease by at least ten percent at each step, we can determine that this algorithm is $\mathcal{O}(n)$.