1) Suppose that you have an $\theta(n^2)$ algorithm that required 10 seconds to run on a problem size of 1000. How long would you expect the algorithm to run on a problem size of 10,000?

2) Suppose that you have an $\theta(n^5)$ algorithm that required 10 seconds to run on a problem size of 1000. How long would you expect the algorithm to run on a problem size of 10,000?

3) Analyze the below algorithms to determine its theta notation, $\theta()$.

```c
index binary_search( int n, const keytype S[], keytype x ) {
    index location, low, mid, high;
    low = 1;
    high = n;
    location = 0;
    while (location <= high && location == 0) {
        mid = (low + high) / 2;
        if (x == S[mid])
            location = mid;
        else if ( x < S[mid])
            high = mid - 1;
        else
            low = mid + 1;
    } // end while
    return location
}
```
void exchangesort (int n, keytype S[ ]) {
    index i, j;
    for ( i = 1; i <= n; i++) {
        for ( j = i + 1; j <= n; j++) {
            if ( S[j] < S[i] ) {
                temp = S[j];
                S[j] = S[i];
                S[i] = temp;
            } // end if
        } // end for j
    } // end for i
}

4) Analyze the below algorithm to determine its theta notation, \( \theta( ) \).

\begin{verbatim}
\textbf{i} := \textbf{n} \\
\textbf{while i > 0} do \\
  \textbf{for} \textbf{j = 1} to \textbf{n} do \\
    \textbf{k} := \textbf{1} \\
    \textbf{while} \textbf{k < i} do \\
      \textbf{< something of } \theta(1)\textbf{>} \\
      \textbf{k} := \textbf{k} \times \textbf{2} \\
    \textbf{end while} \\
  \textbf{end for} \\
\textbf{i} := \textbf{i} \div \textbf{2} \\
\textbf{end while}
\end{verbatim}