Homework Assignment 1 (No Collaborators)

1. **Exercise 2.2−2**
   The loop invariant is that for any iteration i, the smallest $i-1$ elements will be sorted in ascending order. Thus, we only need to run the algorithm on the first $n-1$ elements because the smallest $n-1$ elements will be sorted at that point. In other words, the nth remaining number must be the greatest in our array. In both best and worst cases, the running time of the algorithm is $\theta(n^2)$.

Algorithm 1 Selection Sort Algorithm

Input: Unsorted Array $\mathbf{x}$
Output: Sorted Ascending Order Array $\mathbf{x}$

```
function SelectionSort(x)
    for i ← 0 to length(x)−1 do
        smallest ← x[i]
        for j ← i + 1 to length(x) do
            if x[j] < smallest then
                smallest ← x[j]
            end if
        end for
        x[j] ← x[i]
        x[i] ← smallest
    end for
end function
```

---

2. **Exercise 2.3−3**
   Base case: For $n = 2$, we have $nlogn = 2\log 2 = 2 \cdot 1 = 2$. Therefore, $T(n)$ is true for $n = 2$.

   Inductive Step: Assume that $T(n/2)$ holds for $T(n/2) = (n/2)\log(n/2)$ then, we must show that this holds for $T(n)$.

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2(n/2)\log(n/2) + n \\
&= n(\log(n) - 1) + n \\
&= n\log(n)
\end{aligned}
$$

   Therefore, $T(n)$ is true for all $n \geq 2$ which are exact powers of 2.

3. **Problem 2−3**
   a. The running time of this code fragment is $\Theta(n)$.
   b. The running time of the naive polynomial evaluation algorithm is $\Theta(n^2)$.
   c.
   d. This is trivial to verify because Homer's algorithm returns the correct value for any polynomial.

Algorithm 2 Naive Polynomial-Evaluation Algorthm Part b

```
function NaiveEval(A, x)
    z = 0
    for i ← 1 to A.length do
        coeff = 1
        for j ← 1 to i − 1 do
            coeff *= x
        end for
        z += A[i] · coeff
    end for
return z
end function
```

4. **Prove or disprove** $f(n) + g(n) = \Theta(\textbf{max}(f(n), g(n)))$
   With the assumption that $f(n)$ and $g(n)$ are non-negative functions, $\max(f(n), g(n))$ produces either $f(n)$ or $g(n)$ for all $n \geq 0$. This means that $f(n)+g(n) \leq \max(f(n), g(n))$ for all $n \geq 0$. Therefore, $f(n) + g(n) = \Theta(\text{max}(f(n), g(n)))$.

5. **Problem 3.3a**
   The expressions can be ranked into the following order: $2^{2^{n+1}} \to 2^{2^n} \to (n+1)! \to e^n \to n! \to n \cdot 2^n \to 2^n \to (3/2)^n \to (\log n)! \to n^3 \to n^2 \to 4^{\log n} \to n \log n \to \log(n!) \to n = 2^{\log n} \to (\sqrt{2})^{\log n} \to 2^{\sqrt{2 \log n}} \to \log^2 n \to \ln n \to \ln \ln n \to \sqrt{\log n} \to 2^{\log n} \to \log {}^*n \to \log^*(\log n)$

6. **Exercise 4.1−5**

Algorithm 3 Maximum Subarray Problem

```
Input : A non-empty array A of n numbers.
function MaxSubArray(A)
    maxSoFar ← A[0]
    maxSumHere ← A[0]
    startIndex ← 0
    for i ← 1 to A.length do
        if maxSumHere > 0 then
            maxSumHere ← maxSumHere + A[i]
        else
            maxSumHere ← A[i]
            startIndex ← i
        end if
        if maxSumHere > maxSoFar then
            maxSoFar ← maxSumHere
        end if
    end for
return (maxSoFar)
end function
```

7. **Exercise 4.2–4**
   Assuming that we can multiply 3×3 matrices in k multiplications, then for an $n \cdot n$ matrix, using $n/3$ matrices in $T(n) = kT(n/3) + O(n^2)$. Using the master method, if case 2 applies we have $T(n) = O(n^2 logn)$ and $k = 9$. If case 1 applies, we have $T(n) = O(n^{\log_3 k})$ and $k = 21$. Thus, $k = 21$ is the largest matrix we can multiply in time $O\left(n^{\log 7}\right)$.

8. **Exercise 4.3–7**
   Using the master method in Section 4.5, you can show that the solution to the recurrence $T(n) = 4T(n/3) + n$ is $T(n) = \Theta\left(n^{\log_3 4}\right)$. Show that a substitution proof with the assumption $T(n) \leq cn^{\log_3 4}$ fails. Then show how to subtract off a lower-order term to make a substitution proof work.

9. **Exercise 4.4–9** Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.

10. **Problem 4-3bfhj**