Homework #5 Solutions:

5.17 Use Horner's rule to evaluate:
   a) \( 8x^5 + 2x^4 + 4x^3 + x^2 + 2x + 5 \)
   b) \( 2x^7 + 3x^5 + 2x^3 + 3x^2 + 3x + 7 \)

(8) \( t_1 = 3x + 2 \)
    \( t_2 = t_1x + 4 \)
    \( t_3 = t_2x + 1 \)
    \( t_4 = t_3x + 2 \)
    \( P = t_4x + 5 \)

(9) \( t_1 = 2x + 0 \)
    \( t_2 = t_1x + 3 \)
    \( t_3 = t_2x + 0 \)
    \( t_4 = t_3x + 2 \)
    \( t_5 = t_4x + 5 \)
    \( t_6 = t_5x + 3 \)

Note: the problem does not give specific values to evaluate...

5.32 Prove/Disprove: “If in Step 7 of ‘Procedure’ candidate in ‘Majority’ is \( n \) and \( c > 0 \) then \( c \) is the majority element.”

False: Consider \( 3,1,1,3,3,3,3,3,3,3 \)
For \( AR = 5 \) the \( c = 2 \) when \( j = 10 \) but the majority is in fact 3 and not 5.
Problem 5.24

Modify Algorithm Permutations2 so that it generates all \( k \)-subsets of the set \( \{1, 2, \ldots, n\} \) with \( 1 \leq k \leq n \).

Assuming subset can be unordered. Looking for combinations of length \( k \) for set of \( n \) elements.

Input: \( \text{Combo}(n, k) \)
- \( n \): # of elements
- \( k \): subset length

Output: All \( k \)-combos of elements \( (1, 2, \ldots, n) \).

\[
\text{for } j = 1 \text{ to } k \\
\quad \text{\( C[j] = j \)} \\
\quad \text{\( \text{Combo}(1, 1) \)}
\]

\[
\text{\( \text{Combo}(m, s) \)} \\
\text{if } (m-1) = s \\
\quad \text{\( \text{output } C[1..m] \)} \\
\text{else} \\
\quad \text{for } j = s \text{ to } n \\
\quad \quad \text{\( C[m] = j \)} \\
\quad \quad \text{\( \text{Combo}(m+1, j+1) \)}
\]
Problem 5.25

Analyze the time complexity of the modified algorithm from 5.24.

Time Complexity: \( \Theta(kn! / (n! (n-k)!!)) \)

K slots to fill with n slots \( kn! \) makes sense

k slots

\[
\begin{array}{c|c|c}
1 & 2 & \ldots \text{extra slots} \\
1 & 2 & 3 \quad \text{or} \quad 4 \\
1 & 3 & 4 \quad \text{or} \quad 2 \\
2 & 3 & 4 \\
3 & 4 \\
\end{array}
\]

Divide by the "extra" permutation slots which is

\[
n! (n-k)! = n! (n-k)!\]

the elements = k slots
5.28 Give an iterative version of "Majority"

Input: \( \mathbb{Z}^+ \) array \( A[1, \ldots, n] \)

Output: Return the majority element or zero if no found

```c
int Majority(int * A, int n) {
    int Count = 1;
    int Element = A[1];
    for (int j = 1; j < n; j++) {
        if (A[j] == Element)
            Count++;
        else
            Count--;
    }
    if (Count > n/2)
        return Element;
    else
        return 0;
}
```
5.33 Let $A[1, ..., n]$ be a sorted array of $n \in \mathbb{Z}$ and $x \in \mathbb{Z}$. Design an $O(n)$ time algorithm to determine whether there are two elements $i, j$, if any, whose sum is exactly $x$.

```java
boolean doesSum(int[] A, int x) {
    int left = 1;
    int right = n;
    while (left < right) {
        int sum = A[left] + A[right];
        if (sum == x) {
            return true;
        } else if (sum < x) {
            left += 1;
        } else {
            right -= 1;
        }
    }
    return false;
}
```
1.6 Illustrate the operation of "Insertion Sort" on
30 12 13 13 44 12 25 13
How many comparisons are performed by
the algorithm?

+ (Insert 12)
30 > 12
30 > 13
(Insert 13)
(Insert 44)
30 < 44
(Insert 12)
(Insert 25)
(Insert B)


18 comparisons were performed and

\[18 + 3 = 25\] element assignments

1. I illustrate "Bottom Set (up)" on:

\[A[1...10] = 2, 17, 19 \leq B\]

How many comparisons are performed?

\[
\begin{align*}
2, & 17, 19, 5, 13, 11, 4, 8, 15, 12, 7 \\
2, & 17, 5, 19, 11, 13, 4, 8, 12, 15, 7 \\
2, & 5, 17, 19, 4, 8, 11, 13, 7, 12, 15 \\
2, & 4, 5, 8, 11, 13, 17, 19, 7, 12, 15 \\
2, & 4, 5, 3, 8, 11, 12, 13, 15, 17, 19 \\
\end{align*}
\]

26 comparisons are done
Give an array $A[1, \ldots, 8]$ of integers on which Bottom-Up sort performs:

(a) the minimum # of comparisons

1 2 3 4 5 6 7 8 (16 comparisons)

(b) the maximum # of comparisons

8 1 7 3 6 2 4 (117 comparisons)

Note: The worst case & best case are both $\Theta(n \log n) = \Theta(8 \log 8) = \Theta(16.64)$

$\Rightarrow$ we have the desired array $A$. 