1. Loop Invariant (5 points)

Use the loop invariant (I) to show that the code below correctly computes the product of all elements in an array \( a[0..n - 1] \) of \( n \) integers for any \( n \geq 1 \). First use induction to show that (I) is indeed a loop invariant, and then draw conclusions for the termination of the while loop.

\[
\begin{align*}
\text{Algorithm 1} & \text{ computeProduct}(\text{int}[ ] a[0..n - 1]) \\
p &= a[0] \\
i &= 0 \\
\text{while } i < n - 1 \text{ do} & \\
& \quad // (I) \ p = a[0] \cdot a[1] \cdots a[i] \\
& \quad i ++ \\
& \quad p = p \cdot a[i] \\
\text{end while} & \\
\text{return } p
\end{align*}
\]

2. Recursive Definitions (12 points)

(1) (4 points) Section 5.3 problem 8 a,b,d. Write down the first 6 elements of the sequence, then give a recursive definition for \( a_n \). Do not forget the base case. (You do not need to prove it is correct).

(2) (4 points) Section 5.3 problem 18. Use strong induction. See the bottom of page 180 for a definition of \( A^n \) for a square matrix \( A \). Note that \( f_k \) is the \( k \)th Fibonacci number.

(3) (4 points) Give a recursive algorithm to compute the maximum element of an array of \( n \) integers. Also give the initial call to your recursive algorithm.

3. Structural Induction (5 points)

(1) (5 points) Section 5.3, problem 26 a,c.

3. Applications of Recurrence Relations (4 points)

(1) (4 points) Section 8.1, problem 2. Note that iteration is the same as the expansion method we did in class to generate a guess for the number of moves in the Towers of Hanoi problem.