1. Consider the following algorithm for sorting a list of $n$ numbers $A(1:n)$:

```plaintext
for $i := n$ down to 2 do
    for $j := 1$ to $i - 1$ do
        if $A(j) < A(j + 1)$ then
            swap $A(j)$ and $A(j + 1)$;
        end if
    end for
end for
```

Assume that it takes three operations to swap two elements $A(j)$ and $A(j + 1)$. You do not need to compute the operations it takes to update the loop indices $i$ and $j$. Find the worst-case running time complexity of this algorithm.

2. Consider the following algorithm for computing a polynomial

$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n.$

```plaintext
p := a_0; power := 1;
for $i := 1$ to $n$ do
    power := power * x;
    p := p + a_i * power;
end for
```

Write down a loop invariant for this program, and prove that it correctly computes the value of the polynomial $p(x)$.

3. Consider the recurrence relation $x_n = 2 * x_{n-1} - x_{n-2}$, with $x_0 = 1$, and $x_1 = 3$.

(a) Compute the values of $x_n$, for a few small values of $n$. Prove by induction that $x_n = 2n + 1$, for $n \geq 0$.

(b) Write down an iterative algorithm to compute the value of $x_n$ using the recurrence relation. Your algorithm should not store all of $x_0, x_1, \ldots, x_n$ at the end of the program. Instead, it should store only the two values $x_{i-1}$ and $x_{i-2}$ to calculate the value $x_i$. 