1. (a) (10 pts.) Prove the following function is \( \Theta(n^4) \).

\[ P(n) = 10n^4 - 50n^3 + 200n^2 - 1000n \]

- Prove \( O(n^4) \):

\[ P(n) \leq 10n^4 + 200n^2 \leq n^4(10 + 200/n^2) \leq n^4(10 + 200/100^2), \quad n \geq 100 \leq 10.02n^4. \]

- Prove \( \Omega(n^4) \):

\[ P(n) \geq 10n^4 - 50n^3 - 1000n \geq n^4(10 - 50/n - 1000/n^3) \geq n^4(10 - 50/100 - 1000/100^3), \quad n \geq 100 \geq n^4(10 - 0.5 - 0.001) \geq 9.499n^4. \]

(b) (10 pts.) Consider the following pseudo-code.

\[ x = 0; \]
\[ \text{for } k = 1 \text{ to } n \]
\[ \quad \text{for } j = k \text{ to } 2n \]
\[ \quad \quad x = x + 1; \]

i. Find the exact number of times the innermost statement (increment \( x \)) is executed. That is, what is the final value of \( x \)?

\[ x = \sum_{k=1}^{n} \sum_{j=k}^{2n} (1) = \sum_{k=1}^{n} (2n - k + 1) = n(2n + 1) - \sum_{k=1}^{n} k = 2n^2 + n - n(n + 1)/2 = 3n^2/2 + n/2. \]

ii. Express the time complexity \( T(n) \) of this code in \( O() \) form.

\[ T(n) = \Theta(n^2) \]
2. Consider the sequence $F_0, F_1, F_2, \cdots$ defined recursively as follows. (This recursive definition is called a recurrence relation.)

$$F_n = \begin{cases} 
0, & n = 0 \\
F_{n-1} + n, & n \geq 1. 
\end{cases}$$

(a) (5pts) Compute and tabulate $F_n$ for $n$ values 0 to 10.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$F_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
</tr>
</tbody>
</table>

(b) (15 pts) Prove by induction that for all $n \geq 0$, the solution is

$$F_n = \frac{n^2 + n}{2}.$$

Proof: We use $n = 0$ for the base. From the recurrence, $F_0 = 0$. And the solution for $n = 0$ becomes $F_0 = \frac{0^2 + 0}{2} = 0$. So the solution is correct for the base value.

Now, we will prove that the solution is correct for any $n \geq 1$, supposing that it is correct for $n - 1$. That is, the hypothesis is:

$$F_{n-1} = \frac{(n - 1)^2 + (n - 1)}{2}.$$

Then,

$$F_n = F_{n-1} + n, \quad \text{(from the recurrence)}$$

$$= \frac{(n - 1)^2 + (n - 1)}{2} + n, \quad \text{(by the hypothesis)}$$

$$= \frac{n^2 - 2n + 1 + n - 1}{2} + n$$

$$= \frac{n^2 - n}{2} + n$$

$$= \frac{n^2 - n + 2n}{2}$$

$$= \frac{n^2 + n}{2}.$$
3. (a) (10 pts.) Find the exact solution of the following recurrence. (Assume \( n = 2^k \).) You may use either the repeated substitution method, or the master-theorem.

\[
T(n) = \begin{cases} 
1, & n = 1 \\
8T(n/2) + n^2, & n > 1 
\end{cases} \tag{1}
\]

i. Master Theorem: \( a = 8, b = 2, \alpha = 2 \).

Since \( h = \log_2 8 = 3 \neq \alpha \), the solution form is:

\[
T(n) = An^h + Bn^\alpha = An^3 + Bn^2.
\]

We may find the constants \( A \) and \( B \) using \( T(1) \) and \( T(2) \). (We don’t need induction.)

\[
\begin{align*}
T(1) &= 1 = A + B \\
T(2) &= 8T(1) + 2^2 = 12 = 8A^3 + 4B.
\end{align*}
\]

This gives \( A = 2 \) and \( B = -1 \). Therefore,

\[
T(n) = 2n^3 - n^2.
\]

ii. Repeated Substitution Method:

\[
\begin{align*}
T(n) &= n^2 + 8T(n/2) \\
&= n^2 + 8((n/2)^2 + 8T(n/4)) \\
&= n^2 + 2n^2 + 8^2T(n/2^2) \\
&= n^2 + 2n^2 + 2^2n^2 + 8^3T(n/2^3) \\
& \vdots \\
&= n^2(1 + 2 + 2^2 + \cdots + 2^{k-1}) + 8^kT(n/2^k) \\
&= n^2(2^k - 1) + (2^k)^3T(1) \\
&= n^2(n - 1) + n^3 \\
&= 2n^3 - n^2.
\end{align*}
\]

(b) (10 pts.) Consider the Fibonacci sequence: \( F_1 = 1, F_2 = 1, \) and \( F_n = F_{n-1} + F_{n-2} \) for \( n \geq 3 \).

Use either repeated substitution or induction to prove that for all \( n \geq 2 \),

\[
F_n \geq 2^{(n-2)/2}. \tag{2}
\]

i. Proof by Induction:

Induction Base: We use \( n = 2 \) and \( n = 3 \) for two base cases. For \( n = 2 \), \( F_2 = 1 \) which is equal to \( 2^{(2-2)/2} = 1 \). For \( n = 3 \), \( F_3 = F_2 + F_1 = 2 > 2^{1/2} \). So the two base cases are correct.

Next, for the induction step, we will prove (2) is correct for any \( n \geq 4 \), supposing that it is correct for \( n - 1 \) and \( n - 2 \). That is, the hypothesis is:

\[
F_{n-1} \geq 2^{(n-3)/2}, \text{ and } F_{n-2} \geq 2^{(n-4)/2}.
\]

Then,

\[
F_n = F_{n-1} + F_{n-2} \geq 2^{(n-3)/2} + 2^{(n-4)/2} = 2 \cdot 2^{(n-4)/2} = 2^{(n-2)/2}.
\]
ii. Repeated Substitution Method:

For all $n \geq 3$, observe that $F_{n-1} \geq F_{n-2}$. So,

$$F_n = F_{n-1} + F_{n-2} \geq 2F_{n-2}. $$

Now, we can easily apply repeated substitution to this simplified recurrence.

$$F_n \geq 2F_{n-2} \geq 2^2F_{n-4} \geq 2^3F_{n-6} \geq \cdots \geq 2^kF_{n-2k}. $$

We consider two cases when $n$ is even or odd.

- $n = 2k + 2$:
  $$F_n \geq 2^kF_{n-2k} = 2^{(n-2)/2}F_2 = 2^{(n-2)/2}. $$

- $n = 2k + 1$:
  $$F_n \geq 2^kF_{n-2k} = 2^{(n-1)/2}F_1 = 2^{(n-1)/2} > 2^{(n-2)/2}. $$
4. (a) Given a sorted array of \( n \) elements. Write a recursive version of the binary search algorithm modified so that it locates the **leftmost** occurrence of the search key \( X \) found in the array (in the event that there are several occurrences). For example, for the following array, a search on key 5 should return index 2.

\[
\begin{array}{cccccccc}
i & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

Note: The algorithm must be asymptotically efficient. It is very easy to modify the binary-search algorithm for this problem with time complexity \( O(\log n) \). Therefore, if you give an answer that involves sequential search and has worst-case time \( O(n) \), it would be unacceptable.

```c
int BS (dtype A[], int i, int j, dtype X) {
    //Search A[i..j] for key X
    if (i == j) { //case n = 1 terminates recursion
        if (X == A[i]) return i else return -1
    }
    m = \lfloor (i + j) / 2 \rfloor
    if (X <= A[m])
        return BS(A, i, m, X)
    else return BS(A, m + 1, j, X)
}
```

(b) Illustrate your algorithm on the above example array.

\[
\begin{array}{cccccccc}
m & 2 & 4 & 5 & 5 & 5 & 5 & 8 & 10 & 20 & 25 \\
\end{array}
\]

\[
\begin{array}{cccc}
m & 2 & 4 & 5 & 5 & 5 \\
\end{array}
\]

\[
\begin{array}{ccc}
m & 2 & 4 & 5 \\
\end{array}
\]

5
5. (a) (8 pts.) Insert the following sequence of elements into a Binary-Search-Tree (BST), starting with an empty tree.

Insert: 70, 35, 40, 90, 20, 50, 45, 60, 80, 95, 85

(b) (4 pts) A binary tree is called **height-balanced** if for every node in the tree, the heights of its left and right subtrees differ by at most 1. Is the final BST produced in part (a) height-balanced? Explain.

No, the tree is not height-balanced. The node 35 is unbalanced. So is the node 40.

(c) (8 pts.) Delete element 80 in the following BST. Mark the operations on the given BST. Then, redraw the final BST after deletion.