1. Consider the following summation, where $k$ is a constant. Prove that $f(n)$ is $\Theta(n^{k+1})$.

$$f(n) = \sum_{i=1}^{n} i^k$$

(a) Prove $f(n)$ is $O(n^{k+1})$.

(b) Prove $f(n)$ is $\Omega(n^{k+1})$. 

<table>
<thead>
<tr>
<th>GRADE</th>
<th>/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td>/100</td>
</tr>
</tbody>
</table>
2. Fibonacci sequence is recursively defined as follows: \( F_1 = 1, \ F_2 = 1 \) and \( F_n = F_{n-1} + F_{n-2}, \ n \geq 3 \).

(a) Compute and tabulate \( F_n \) for \( n = 1 \) to 12.

(b) Prove by induction the following upper bound for \( F_n \).

\[
F_n \leq 2^n, \ n \geq 1.
\]

(c) Prove by induction the following lower bound for \( F_n \).

\[
F_n \geq 2^{n/2}, \ n \geq 6.
\]
3. Consider the following divide-and-conquer algorithm, where the initial call is Find($A, 0, n - 1$). Assume the array size, $n$, is a power of 2.

```java
Boolean Find(keys A[], int left, int right) {
  1. if (left==right) return(FALSE);
  2. if (A[left] != A[right]) return(TRUE);
  3. int m= FLOOR((left+right)/2);
  4. return (Find(A,left,m) OR Find(A,m+1,right))
}
```

(a) Explain what this program does.

(b) Let $f(n)$ be the worst-case number of key comparisons for an array of size $n$. (Note that a key-comparison is the comparison of line 2, but not line 1.) Write a recurrence for $f(n)$.

(c) Find the exact solution for $f(n)$. Then express the order.
4. Given an array $A[0..n-1]$ which consists of an increasing sequence followed by a decreasing sequence. Assume the elements are all distinct. Thus, there is an index $k$ in the range $[0..n-1]$, where $A[k]$ is the maximum and


(a) Describe an efficient algorithm which takes the array $A$ as input and finds $k$. Your algorithm must have time complexity asymptotically better than $n$. Analyze the time complexity of your algorithm.

(b) Write the pseudocode for your algorithm.
5. Consider the following recurrence: \( T(1) = 2 \) and \( T(n) = 2T(n/2) + n^3, \ n > 1. \) (Assume \( n \) is a power of 2.)

(a) Use master theorem to find the exact solution, then express the order of it.

(b) Use repeated substitution method to find the solution.