1. (25 points) Please show the result of Red-Black Tree after inserting each of the following numbers, 7, 3, 4, 9, 8, in the given order into an empty tree.

Answer: presented in class.

2. (25 points) The function F(n) will compute the n\textsuperscript{th} Fibonacci number for n >= 0:
   \[
   \text{int } F(\text{int } n) \{ \text{if } (n == 0 || n == 1) \text{ return } n; \text{ else return } F(n-1)+F(n-2); \} 
   \]
   Please define the number of “+” used to compute F(n) as a function A(n) and prove formally that A(n) >= F(n) for n > 1.

Answer: A(n) can be recursively defined as:
   \[
   A(0) = A(1) = 0 \quad \text{(because F(0) and F(1) doesn’t use “+”).} \\
   A(n) = A(n–1)+A(n–2) + 1 \text{ for } n > 1 \quad \text{(A(n–1) is the cost of F(n–1), A(n–2) is the cost of F(n–2), and “1” is for “+” in “F(n–1)+F(n–2)”.)}
   \]
   To show A(n) >= F(n) , we use mathematical induction.
   Base cases: n = 2: A(2) = A(1)+A(0)+1 = 1 >= F(2) = F(1)+F(0) = 1+0 = 1.
   n = 3: A(3) = A(2)+A(1)+1 = 2 >= F(3) = F(2)+F(1) = 1+1 = 2.
   Induction hypotheses: A(n–1) >= F(n–1) and A(n–2) >= F(n–2) for n > 3.
   Inductive case: For n > 3, A(n) = A(n–1) + A(n–2) + 1 \quad \text{// definition of A(n)}
   \[
   >= F(n–1) + F(n–2) + 1 \quad \text{// induction hypotheses.} \\
   > F(n–1) + F(n–2) \\
   = F(n) \quad \text{// definition of F(n)}
   \]
   Note: Using the same proving process, we can prove A(n) = F(n+1) – 1.

3. (25 points) Given a sequence of n integers S = [a\textsubscript{1}, a\textsubscript{2}, \ldots, a\textsubscript{n}], a subsequence of S is said to be monotonic if this subsequence is sorted. For example, if S = [6, 3, 4, 7, 9, 8], then subsequences [6, 7, 9], [3, 4, 7, 9], [3, 4, 7, 8], \ldots, are monotonic. Please design an efficient algorithm to find the length of longest monotonic subsequences of S, and analyze its time and space complexity. Please analyze the complexity of your algorithm.

Answer: There are at least two solutions.
   (a) S’ = copy(S); sort(S’); return LCS(S, S’);
where LCS is the Longest Common Subsequence discussed in class (omitted here). The time complexity is $O(n \log n + n^2) = O(n^2)$. The space is $O(n)$.

(b) Directly apply dynamic programming.
Let $\text{LMS}(i)$ be the length of the longest monotonic subsequence in the sequence $= [a_1, a_2, \ldots, a_i]$ and the last symbol of the subsequence is $a_i$. Then $\text{LMS}(0) = 0$. The solution we are looking is $\max_{0 \leq i \leq n} \{\text{LMS}(i)\}$. For $i>1$, $\text{LMS}(i)$ is obtained from $\text{LMC}(j)$ where $0<j<i$ and $a_j \leq a_i$. That is, we can add $a_i$ to the solution of $\text{LMS}(k)$.

$$\text{LMS}(i) = 1 + \max_{0 \leq j < i} \{\text{LMS}(j) : a_j \leq a_i\}.$$ 

The order we compute $\text{LMS}(i)$ will be for $i$ from 0 to $n$. Here is the pseudo code:

```plaintext
LMS[0] = 0;
for (int i=1; i<=n; i++) {
    LMS[i] = 1;  // $a_i$ is a monotonic subsequence
    for (int j=i-1; j>0; j--) if (a_j <= a_i && LMS[j]>=LMS[i]) LMS[i] = LMS[j]+1;
}
int result = 0;
for (int i=1; i<=n; i++) if (result < LMS[i]) result = LMS[i];
return result;
```

The time complexity is $O(n^2)$ and the space complexity is $O(n)$.

4. (25 points) We like to sort $n$ student records by their grades for a course. If the grades are $A$, $B$ or $C$, what is the most efficient sorting algorithm for this job? If the students are stored in an array, what is the most efficient in-place sorting algorithm for this job? Please provide the algorithms in details.

Answer: Suppose the students records are stored in the array $S[0...n-1]$. The solution below takes $O(n)$ time and works for both cases. $S[0..a]$ are ‘A’; $S[c..n-1]$ are ‘C’.

```plaintext
Sortbyscore(S, n) {
    int a = -1, b=0, c = n;
    while (b<c)
        switch(score(S[b])) {
            case ‘A’: a++; swap(S, a, b); b++; break;
            case ‘B’: b++; break;
            case ‘C’: c--; swap(S, b, c); break;
        }
}
```

This problem is called the Dutch National Flag problem in computer science, where $A$, $B$, and $C$ represent three colors of the Dutch National Flag.