Analysis of Algorithms **Comprehensive Examination, Fall 2002**

Sign the exam with your student number - not your name ____________________

Answer the following questions to the best of your ability.

1. (30 pts) Give big-O estimates on the size of the following sums.

   - $\sum_{i=1}^{n} 1$
   - $\sum_{i=1}^{n} i$
   - $\sum_{i=0}^{n} 2^i$
   - $\sum_{i=1}^{n} \frac{1}{i}$
   - $\sum_{i=1}^{n} \lg i$
   - $\sum_{i=1}^{n} i \lg i$

2. (20 pts) What is the time and space complexity of the algorithm given below.

```
longestCommonSubsequence(String X, String Y) {
  int m = X.length();
  int n = Y.length();
  int C[m][n];
  int B[m][n];
  for (int i = 0; i < m; i++) { C[i][0] = 0; }
  for (int j = 0; j < n; j++) { C[0][j] = 0; }
  for (int i = 1; i < m; i++) {
    for (int j = 1; j < n; j++) {
      if (X.charAt(i) == Y.charAt(j)) {
         C[i][j] = C[i-1][j-1]+1;
         B[i][j] = 0;
      }
      else if (c[i-1][j] >= c[i][j-1]) {
         C[i][j] = C[i-1][j];
         B[i][j] = 1;
      }
      else {
         C[i][j] = C[i][j-1];
         B[i][j] = 2;
      }
    }
  }
}
```

3. (20 pts) The algorithm given below solves the matrix chain multiplication problem: given a multiplication chain $A_1 A_2 \cdots A_n$ of matrices specify the order of multiplications to minimize the scalar multiplications. This minimum number is returned as `m[1][n]`, and the array `p[]` holds the orders of the matrices (A_1 is $p_0 \times p_1$, A_2 is $p_1 \times p_2$, $\ldots A_n$ is $p_{n-1} \times p_n$).

```
#define INFINITY MAX_VALUE // 2147483647
matrix-chain(int[] p, int} i, int j) {
  if (i == j) { return 0;}
  m[i][j] = INFINITY;
  for (k=i; k < j; k++) {
    q = matrix-chain(p, i, k) + matrix-chain(p, k+1, j) + p[i-1]*p[k]*[j];
    if (q < m[i][j]) { m[i][j]=q;}
  }
  return m[i][j];
}
```

Let $T(n)$ denote the time to compute `m[1][n]` by the call `matrixChain(p, 1, n)`. Pretend $T(1) = 1$.

- Write a recurrence relation that expresses $T(n)$ in terms of a sum of $T(1), \ldots, T(n-1)$ (and any other needed terms or factors).
- Use your formula with mathematical induction to prove that $T(n) \geq 2^{n-1}$.

4. (30 pts) This question asks you to analyze 2 algorithms that compute Fibonacci numbers. Pretend the cost of adding, subtracting, or multiplying two numbers is $O(1)$, independent of the size of the numbers.

   a) One algorithm to evaluate Fibonacci numbers is given in the code below.

```
int Fibonacci(int n) {
  if ((0==n) || (1==n)) { return n; }
  else { return Fibonacci(n-1) + Fibonacci(n-2); }
}
```

      i) What is the time complexity of the algorithm?

      ii) Is the algorithm efficient or not? Give reasons for you answer.

   2) Another algorithm for evaluating Fibonacci numbers is:

```
int Fibonacci(int n) {
  int F1=1, F0=0;
  for (int i=1; i <= n; i++) {
    F0=F1 + F0;
    F1=F0-F1;
  }
  return F0;
}
```

      i) What is the time complexity of the algorithm?

      ii) Is the algorithm efficient or not? Give reasons for you answer.