Analysis of Algorithms **Comprehensive Examination, Spring 2001**

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

Answer all three questions to the best of your ability.

1. (40 pts) Provide a *time and space* analysis of the Floyd-Warshall algorithm below that finds the shortest path between all nodes in a graph with n nodes given an $n \times n$ matrix of edge `weights`. Explain your reasoning.

1 ⟨*Floyd-Warshall* 1⟩≡

```
 public int[][] FloydWarshall(int[][] weights) {
   int n = weights.length;
   int[][] D = new int[n][n];
   D = weights;
   for (int k = 0; k < n; k++) {
      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
             D[i][j] = min(D[i][j], D[i][k]+D[k][j]);
         }
      }
   }
   return D;
}
public int min(a, b) {
  if (a < b) return a;
  else return b;
}
```

2. (40 pts) The recursive algorithm below computes x^n.

2 $\langle$*Exponentiation* 2$\rangle\equiv$

```
public double exp(double x, int n) {
   if (0 == n) return 1;
   else if (0 == n % 2) { // n is even
      double y = exp(x, n/2);
      return y*y;
   else { // n is odd
      return x*exp(x, n-1);
   }
}
```

1. Define a reasonable *initial condition* for the time complexity when $n = 0$.
2. Define a recurrence relation that describes the recursive nature of the algorithm. Note that the recurrence relation will have two cases: for even and odd values of the exponent n.
3. Provide the best analysis you can to describe the time complexity of the algorithm. Note that when n is odd in one call it is even in the next call.

3. (10 pts) Solve the recurrence relation

$$T(n) = 2T(n/2) + 1$$

with initial condition T(1) = 1 (you may assume n is a power of 2, e.g., $n = 2^p$ for some integer p).

4. (10 pts) Suppose an array $X[0..n-1]$ has been sprinkled with random real numbers chosen uniformly over the range [0, 1], and consider the code fragment:

4 ⟨*average case analysis* 4⟩≡

```
float max = X[1];
for (int i = 2; i < n; i++) {
    if (max < X[i]) {
        max = X[i];
    }
 }
```

What is the expected number of times that the variable `max` will be re-set? That is, what is the average time complexity of the statement `max = X[i]` that is inside the `for` loop?