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

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

Answer all questions to the best of your ability.

1. (30 pts) Here's a problem from Brassard and Bratley. Consider the `wastePaper()` algorithm below. Let $W(n)$ stand for the number of lines of output generated by a call to `wastePaper(n)`.

   a. (10 pts) What is the value of $W(0)$?

   b. (10 pts) Use summation notation to compute the complexity lines 1 – 5.

   c. (10 pts) Write a recurrence equation for $W(n)$

⟨ *Waste Paper* ⟩≡

```
0:  public void wastePaper(int n) {
1:    for (int i = 0; i < n; i++) {
2:      for (int j = 0; j <= i; j++) {
3:        System.out.println("i = "+i+", j= "+j+", n = "+n);
4:      }
5:    }
6:    if (n > 0) {
7:      for (int i = 0; i < 4; i++) {
8:        wastePaper(n/2);
9:      }
10:   }
11: }
```

2. (25 pts) Find a simple formula for the recurrence equation:

$$T(n) = 2T(n-2) + 1 \quad T(0) = 0,\, T(1) = 1$$

3. (30 pts) This problem asks questions about the `heap` data structure.

   a. (5 pts) Define: *left-complete binary tree.*

   b. (5 pts) Explain how a left-complete binary tree can be stored in an array.

   c. (5 pts) Define: *the heap property of a left-complete binary tree.*

   d. (15 pts) The following algorithm, from Sedgewick, is used in building a heap. What is the worst case time complexity of `heapify`?

3 ⟨*Heapify an array* 3⟩≡

```
public void  heapify (int[] A, int k, int n) {
  while (2*k <= n) {
    j = 2*k;
    if ((j < n) && (A[j] < A[j+1])) j++;
    if (A[k] >=  A[j]) break;
    swap(A[k], A[j]);
    k = j;
  }
}
```

4. (15 pts) You are do design an algorithm for the following task.

> Given two positive integers M and N, with $M < N$. Output a *sorted* list of M random integers in the range 1—N with no integer occurring *more than once*.

You may use a function `RandInt(I,J)` that returns a random integers chosen uniformly from the range I—J or a function `RandReal(0,1)` that returns a random real number chosen uniformly from the range $[0, 1)$. Analyze the complexity of your algorithm.