22C:44 Homework 1

Due by 5 pm on Tuesday, 2/13

---

1. Consider the following "strange" function:

```
Strange(n) {
      for i ← to n do
            if (n mod i == 0) then
                  for j ← 1 to n do
                        print(i, j);
}
```

   (a) [5 points] Your friend who took 22C:44 last semester tells you that the running time of `Strange(n)` is $\Theta(n^2)$. Disprove her claim.

   (b) [5 points] Your elder brother who took 22C:44 about 3 years ago claims that the running time of `Strange(n)` is $\Theta(n)$. Disprove his claim.

   (c) [3 points] Use the Big-Oh and Big-Omega notation respectively, to express asymptotic upper and lower bounds on the running time of `Strange(n)`. Make these bounds as tight as possible.

2. Consider the following "stranger" function:

```
Stranger(A[1..n], i){
      Define j;
      if (i == (n+1)) then
            print(A)
      else
            for j ← i to n do {
                  swap(A, i, j);
                  Stranger(A, i + 1);
                  swap(A, i, j);
            }
}
```

   Here the calls to `swap(A, i, j)` swap the elements `A[i]` and `A[j]`.

   (a) [5 points] Assuming that `A` is an array of n elements and i is an integer satisfying $1 \le i \le n+1$, let $T(n-i+1)$ denote the running time of a call to the function `Stranger(A, i)`. Set up the recurrence relation for the running time of the function call `Stranger(A, 1)` for an n-element array `A`.

   (b) [5 points] Solve the above recurrence and determine the running time of `Stranger(A, 1)`, asymptotically.

   (c) [2 points] Despite appearances to the contrary, `Stranger` does something reasonable, especially when called as `Stranger(A, 1)` where `A` contains the sequence $1, 2, \dots, n$. Explain in a sentence what the function `Stranger` does.

3. Let us investigate the problem of sorting arrays $A[1 \dots n]$ that are known to be almost ordered initially in the sense that only some elements close to each other may be in the wrong order. More precisely, there exists a constant c (independent of n) such that whenever two element $A[i]$ and $A[j]$ are in the wrong order then $|j - i| \le c$. Suppose that such an almost sorted array is given as input to the `MergeSort` function.

(a) [5 points] Modify the `Merge` function so that it runs in $\Theta(1)$ time.

(b) [5 points] Let $T(n)$ be the running time of `MergeSort` on an array of size n. Rewrite the recurrence relation for $T(n)$ and solve it to determine the new running time of `MergeSort` in Θ-notation.

4. Solve the following recurrence relations using the *iteration method*. For each problem, assume that $T(n) = \Theta(1)$ for $n \leq 1$ and $T(n)$ for $n > 1$ is given below.

   (a) $T(n) = aT(n/b) + \Theta(n)$. Here a and b are positive integers.

   (b) $T(n) = 5T(n/5) + n^2$.

   (c) $T(n) = T(n/2) + T(n/3) + n$.

   (d) $T(n) = T(n-2) + 7$.

   (e) $T(n) = nT(n-1) + 1$.

   For Part (a), you may have to consider the cases $a > b$, $a = b$, and $a < b$ separately.

   Each part is worth 3 points.
