

ICS H23—Spring 2008  
Honors Introduction to Computer Science III  
Homework 1 Answers

```
1. a)   if (L2 != null) {
        if (L1 == null) L1 = L2;
        else {
            x = L1;
            while (x.next != null) x = x.next;
            x.next = L2;
            L2.prev = x;
        }
    }
    L = L1;
```

b) (Surprisingly, the code below works even if L2 is empty.)

```
L1.prev.next = L2.next;
L2.next.prev = L1.prev;
L2.prev.next = L1;
L1.prev = L2.prev;
L = L1;
```

```
c)   if (L2 != null) {
        if (L1 == null) L1 = L2;
        else {
            temp = L1.next;
            L1.next = L2.next;
            L2.next = temp;
            L1 = L2;
        }
    }
    L = L1;
```

2. a) There are many pairs of values for  $n_0$  and  $c$  that work. Here are two examples of how we might produce a pair of values. One approach is to first realize that letting  $n_0$  be 0 certainly won't work, and therefore try to find a  $c$  that works for  $n \geq 1$ . If  $n \geq 1$ , we know that  $1 \leq n^2$ , so

$$3n^2 - 8n + 12 \leq 3n^2 + 12 \leq 3n^2 + 12n^2 = 15n^2,$$

so letting  $n_0 = 1$  and  $c = 15$  will work.

Alternatively, we could first try choosing a value for  $c$ . Suppose that we try  $c = 4$ . Then we need to find an  $n_0$  such that for  $n \geq n_0$ ,  $3n^2 - 8n + 12 \leq 4n^2$ ; to show this it is sufficient to show that  $3n^2 + 12 \leq 4n^2$ . This is true if  $12 \leq n^2$ , which is true if  $n \geq 4$ . So, the choices  $c = 4$ ,  $n_0 = 4$  work.

In fact, we could even make  $c = 3$ . Note that then we want to have  $3n^2 - 8n + 12 \leq 3n^2$ , which is true if  $-8n + 12 \leq 0$ , which in turn is true if  $12 \leq 8n$ . So, letting  $c = 3$ ,  $n_0 = 2$  will also work.

Actually, since we must show that

$$n \geq n_0 \implies |3n^2 - 8n + 12| \leq cn^2,$$

we must also show that  $3n^2 - 8n + 12$  never becomes too negative. In fact, though, we always have  $3n^2 - 8n + 12 > 0$ . For  $n = 0, 1, \text{ or } 2$ , we can verify this directly. For larger  $n$  we have

$$3n^2 - 8n + 12 = n(3n - 8) + 12 > 0,$$

since then  $3n - 8$  is positive.

- b) One way to satisfy the conditions is to let  $c = \frac{1}{2}$  and  $n_0 = 1$ . Then for  $n \geq n_0$  we have  $\frac{1}{2}n^2 + 8n \geq \frac{1}{2}n^2$ .

*Note:* It might be tempting to say that something like  $n_0 = 4$  and  $c = 2$  would work, because then  $\frac{1}{2}n_0^2 + 8n_0 = 8 + 32 = 40 > 2n_0^2$ . This isn't enough, though, because the inequality has to hold for *all*  $n \geq n_0$ . For example, with this choice of  $n_0$  and  $c$ , for  $n = 100$  we have  $n \geq n_0$ , but  $\frac{1}{2}n^2 + 8n = \frac{1}{2}100^2 + 8 \cdot 100 = 5800$ , which is less than  $2n^2 = 20000$ .

In fact, if we choose any  $c > \frac{1}{2}$ , no  $n_0$  will work.

- c) (To be discussed in class.)
3. a)  $\Theta(n^3)$
- b)  $\Theta(n^2)$ . (For this one it is useful to consider the sum of an arithmetic sequence.)
- c)  $\Theta(n \log n)$ . (The outer **for** loop causes us to do  $\Theta(\log n)$  complete iterations of the inner **for** loop, and each of these takes  $\Theta(n)$  time.)
- d)  $\Theta(n)$ . This was the trickiest one. Note that on the  $k$ th execution of the inner loop, we have  $i = 2^{k-1}$ . Thus for one pass through the outer loop, the number of executions of the inner **for** statement is  $2^{k-1}$ , and the number of executions of the last statement is  $2^{k-1} - 1$ . Let  $k_0$  be the largest integer such that  $2^{k_0-1} < n$ , so we do  $k_0$  complete executions of the inner loop. Note that we must have  $2^{k_0} \geq n$ , so

$$2^{k_0-1} < n \leq 2^{k_0},$$

and therefore  $2^{k_0} = \Theta(n)$ . Then the total number of executions of the inner **for** statement is

$$\sum_{k=1}^{k_0} 2^{k-1} = 2^{k_0} - 1 = \Theta(n).$$

Similarly the total number of executions of the last statement is  $\Theta(n)$ . The total number of executions of the outer **for** statement is  $\Theta(\log n)$ , so the total time is  $\Theta(n)$ .