CS 61A, Summer 2006 July 31, 2006

HOMEWORK ASSIGNMENT 6A

DUE WEDNESDAY AUGUST 9, 2006 AT 11:59 PM

These exercises use the Instant Message program, found in the following files:

```
~cs61a/lib/im-client.scm
~cs61a/lib/im-server.scm
~cs61a/lib/im-common.scm
```

1. Invent the capability to send a message to a list of clients as well as to a single client. Do this entirely in the client program, so what actually goes to the server is multiple requests.

2. Invent the capability to broadcast a message to every client. Do this by inventing a BROADCAST command that the server understands.

3. Could #1 have been done with the server doing part of the work? Could #2 have been done entirely in the client code? Compare the virtues of the two approaches.

4. Invent the capability of refusing messages from specific people. The sender of a refused message should be notified of the refusal. Decide whether to do it entirely in the client or with the server's cooperation, and explain why.

5. Why is the 3-way handshake necessary when connecting to the server?

• Textbook exercises 3.38, 3.39, 3.40, 3.41, 3.42, 3.44, 3.46, 3.48

Vector questions: In all these exercises, don't use a list as an intermediate value. (That is, don't convert the vectors to lists!)

1. Write `vector-append`, which takes two vectors as arguments and returns a new vector containing the elements of both arguments, analogous to `append` for lists.

2. Write `vector-filter`, which takes a predicate function and a vector as arguments, and returns a new vector containing only those elements of the argument vector for which the predicate returns true. The new vector should be exactly big enough for the chosen elements. Compare the running time of your program to this version:

```
(define (vector-filter pred vec)
(list->vector (filter pred (vector->list vec))))
```

3. Sorting a vector.

(a) Write `bubble-sort!`, which takes a vector of numbers and rearranges them to be in increasing order. (You'll modify the argument vector; don't create a new one.) It uses the following algorithm:

[1] Go through the array, looking at two adjacent elements at a time, starting with elements 0 and 1. If the earlier element is larger than the later element, swap them. Then look at the next overlapping pair (0 and 1, then 1 and 2, etc.).

[2] Recursively bubble-sort all but the last element (which is now the largest element).

[3] Stop when you have only one element to sort.

(b) Prove that this algorithm really does sort the vector. Hint: Prove the parenthetical claim in step [2].

(c) What is the order of growth of the running time of this algorithm?