YOUR NAME: ________________________________

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (e.g., submitting another person’s work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Science Department Code of Conduct can be found at http://www.cs.utexas.edu/academics/conduct/.

Late submission policy

This homework is due at the beginning of class on November 4. All late submissions will be subject to the following policy.

You start the semester with a credit of 3 late days. For the purpose of counting late days, a “day” is 24 hours starting at 2pm on the assignment’s due date. Partial days are rounded up to the next full day. You are free to divide your late days among the take-home assignments any way you want: submit four assignments 1 day late, submit one assignment 3 days late, etc. After your 3 days are used up, no late submissions will be accepted and you will automatically receive 0 points for each late assignment.

You may submit late assignments to Vitaly Shmatikov (CSA 1.114—slide under the door if the office is locked). If you are submitting late, please indicate how many late days you are using.

Write the number of late days you are using: ______
Homework #5 (35 points)

Problem 1

Consider the following ML function that constructs a new association list by calling a function on each of the values in the association list (in this function, _hd_ returns the head of a list, _tl_ returns the tail):

```ml
fun key(k,v) = k;
fun value(k,v) = v;
fun mapValues f alist = 
  if null alist then nil 
  else (key(hd alist),f(value (hd alist))) :: mapValues f (tl alist);
```

Here is how _mapValues_ might be used:

```ml
mapValues double [("bob",4), ("betty",7), ("jane",6)] => 
[("bob",8), ("betty",14), ("jane",12)]
```

Problem 1a (3 points)

What types will be inferred for _key_ and _value_?

Problem 1b (4 points)

Explain how the ML type inference algorithm would compute the type of the _mapValues_ function. What is the resulting type?
Problem 2 (4 points)
Solve Problem 6.6 from the Mitchell textbook (p. 158).

Problem 3
Consider the following Java implementation of a simple publish-subscribe system. When new data are published, all subscribers should be notified and updated automatically.

```java
class Publisher {
    private List subscribers = new LinkedList();
    private String data;

    public interface Subscriber {
        public void updateData(String newData);
    }

    public void addSubscriber(Subscriber subscriber) {
        subscribers.add(subscriber);
    }

    public void publishData(String newData) {
        data = newData;
        Iterator i = subscribers.iterator();
        while(i.hasNext()) {
            ((Subscriber)i.next()).updateData(newData);
        }
    }
}
```
Problem 3a (2 points)

Is the above implementation thread-safe? Explain.

Problem 3b (4 points)

Suppose we make `publishData` and `addSubscriber` methods synchronized. Does this solve the problem? If not, give a specific example and explain what goes wrong.

Problem 3c (4 points)

Suppose the `addSubscriber` method is synchronized, and the `publishData` method is as follows:

```java
public void publishData(String newData) {
    List copyOfSubscribers;
    synchronized(this) {
        data = newData;
        copyOfSubscribers = new LinkedList(subscribers);
    }
    Iterator i = copyOfSubscribers.iterator();
    while(i.hasNext()) {
        ((Subscriber)i.next()).updateData(newData);
    }
}
```
Is the new implementation thread-safe? If not, give a specific example and explain what goes wrong.

Problem 4
Consider the following imperative program:

function f(x) { return x+1; }
function g(y) { return 2*y; }
f(g(1));

Problem 4a (3 points)
Write the above program fragment as a $\lambda$-expression.

Problem 4b (2 points)
Evaluate your expression by choosing, at each step, the reduction that eliminates the leftmost $\lambda$ that can be reduced.
Problem 4c (2 points)
Evaluate your expression by choosing, at each step, the reduction that eliminates the right-most \( \lambda \) that can be reduced.

Problem 5

Problem 5a (5 points)
Translate the following ML program into \( \lambda \)-calculus:

\[
\text{(let fun subtract(x) =}
\quad \text{fn(y) => let fun unaryminus(z)=(~z) in}
\quad \quad \text{x+unaryminus(y)}
\quad \text{end}
\quad \text{end) 10}
\]

Problem 5b (2 points)
Reduce the resulting \( \lambda \)-expression.