

CIS-280
Homework 6: Abstraction; Trees
Due Tuesday, April 12, 2005
80 points

The last two problems are fairly easy procedures. Be sure to save your code from them (`FindJoy`, `FindSad`, `FindJoyEmp`, and `FindSadEmp`) since you will need it for a future homework on data-directed programming.

1. (10 points) `map` takes a procedure and a list as arguments and returns a list of the results of applying the procedure to each item in the list. Write a procedure (`apply-all item lst`) that instead takes as arguments an element `item` (which could be a list) and a list `lst` of procedures, and returns a list of the results of applying each procedure in `lst` to `item`. For example,

```
(apply-all 4 (list (lambda(x)(* x x)) sqrt)) returns (16 2)
(apply-all '(4 5 8) (list length car (lambda(x)(cons 7 x)))) returns (3 4 (7 4 5 8))
```

2. (10 points) Recall that we defined `accumulate` on sequences as follows:

```
(define accumulate (lambda (op init-val lst)
  (if (null? lst)
      init-val
      (op (car lst) (accumulate op init-val (cdr lst))))))
```

In the following, you are to define `append` and `length` in terms of `accumulate`. (Although the TAs and I will not help you do the following, we will review and explain examples from class. The following two problems are a straightforward application of the concepts discussed in class, but it may at first appear a bit challenging.)

- (a) Complete the following definition of `append` by replacing `?A` and `?B` with the arguments that should be passed to `init-val` and `lst` respectively.

```
(define append1 (lambda (seq1 seq2)
  (accumulate cons ?A ?B)))
```

- (b) Complete the following definition of `length` by replacing `?C` with the argument that should be passed to `op`.

```
(define length1 (lambda (seq)
  (accumulate ?C 0 seq)))
```

3. (15 points) Procedure (`CountSubtrees tree`) takes `tree`, a set represented as an ordered binary tree as described in class, as its argument and returns a count of the number of subtrees (including the tree itself as a subtree) whose root has a left or a right branch but not both. Design and test procedure `CountSubtrees`. Your procedure must traverse `tree` using the procedures `entry`, `left-subtree`, and `right-subtree` as described in class, not just process a list using list operations.
4. (15 points) Procedure (`Smaller tree num`) takes as arguments a set of integers represented as an ordered binary tree `tree` and an integer `num` and returns an ordered list (ordered from smallest to largest) of the integers in `tree` that are smaller than `num`. You will want to use procedures `entry`, `left-subtree`, and `right-subtree` as described in class. Design and test procedure `Smaller`. Your procedure should be efficiently designed so that it does not look at parts of the tree that can be eliminated from consideration.

5. (15 points) The **Joy** Chemical Company has a file of personnel records, one record for each employee. Each record is represented as an unordered list of entries, where each entry consists of an attribute and a value for that attribute. For example, a record for the employee whose `employee#` is 489235 and containing his age (25) and his salary (50000) would be represented as:

```
((empnum 489235)(age 25)(salary 50000))
```

The **Sad** Chemical Company has a file of personnel records. Each record is represented as a list of two lists, one containing the names of attributes that are stored for that employee and the second giving the values of those attributes in the same order as the attribute names appeared in the first list. For example, the above employee might be represented as

```
((empnum age salary) (489235 25 50000))
```

Design and test two procedures, (`FindJoyAtt record att-name`) and (`FindSadAtt record att-name`) (one for each chemical company), that take as argument an employee record `record` and an attribute name `att-name` and return the attribute's value for that employee record or `()` if the attribute does not appear in the record.

6. (15 points) Employee records are stored in files. For the **Joy** Chemical Company, a file of employee records is represented as a list of employee records ordered according to employee number. For the **Sad** Chemical Company, a file of employee records is represented as an ordered binary tree of employee records ordered according to employee number. Suppose we have the following employees and attributes:

```
Employee-1: empnum: 119911
             age: 24
Employee-2: empnum: 489235
             age: 25
             salary: 50000
Employee-3: empnum: 556647
             salary: 20000
```

Then the Joy Chemical Company might represent the file as:

```
((age 24)(empnum 119911))
((empnum 489235)(age 25)(salary 50000))
((salary 20000)(empnum 556647)))
```

and the Sad Chemical Company might represent the file as:

```
((age empnum salary) (25 489235 50000))
(((empnum age)(119911 24)) () ())
(((empnum salary)(556647 20000)) () () ))
```

Design and test two procedures, (`FindJoyEmp file number`) and (`FindSadEmp file number`) (one for each chemical company), that take as arguments a file of employee records `file` and an `employee# number` and return the employee record if it exists and `()` otherwise. `FindJoyEmp` and `FindSadEmp` should make use of procedures `FindJoyAtt` and `FindSadAtt`.

Submit your code for procedures `apply-all`, `append1`, `length1`, `CountSubtrees`, `Smaller`, `FindJoyAtt`, `FindSadAtt`, `FindJoyEmp`, and `FindSadEmp` on the electronic submission system for Homework-6.