# CSCE 330 Fall 2000

Final Exam

Assigned Saturday 00/12/16

This exam is "open book." Please show your work for partial credit. There are eight problems on six pages in this exam. The scores for the problems sum up to 180, but the highest grade that can be obtained in 145. The score for each question is given so that you can budget your time. You will be graded on clarity and conciseness as well as correctness. Good luck!

**Problem 1** [5 points–Syntax and Semantics].
Match

**a** Syntax

**b** Semantics

to

$\alpha$ Function

$\beta$ Form

and

**1** The set of rules that specify the composition of programs from letters, digits, and other characters

**2** The set of rules that specify what the result of a program is.

**Answer:** a, beta, 1; b, alpha, 2.

**Problem 2** [5 points–Denotational Semantics].
In semantical terms, what is the difference between an expression and a statement? (Hint: Contrast the denotational semantics of expression and assignment statements.)
**Answer:** An expression does not change the state; a statement does.

**Problem 3** [5 points–Lists in ML]. This question deals with the difference between the list constructor (`::`) and the list concatenation operator `@` in ML. One of the following expressions is different from the others. Which one? Is it syntactically incorrect? If so, why?

1. `[2] @ [3]`
2. `2 :: 3 :: nil`
3. `2 :: [3]`
4. `2 @ [3]`
5. `(2 :: []) @ (3 :: nil)`
6. `2 :: 3 :: []`

**Answer:** The fourth one; yes, because concatenation requires two lists, and the first argument is not a list.

**Problem 4** [25 points–Prolog]. Assume that the following relations are defined: `interval(Name, Start, End)`, `lte(Small, Large)`, `gte(Large, Small)`.

`interval(N,S,E)` holds when the interval called N starts at S and ends at E. `lte(S,L)` holds when S is smaller than or equal to L. `gte(L,S)` holds when L is larger than or equal to S.

Define a relation *overlaps* that holds when two intervals overlap.

**Answer:** `overlaps(B1,D1, B2,D2) :- lte (B1, B2), lte (B2, D1).`
`overlaps(B1,D1, B2,D2) :- lte (B2,B1), lte(B1,D2).`

**Problem 5** [25 points–Recursive Functions in ML]

Assume that you represent sets as lists. Write a function `subset (S, T)` that returns `true` if S is a subset of T and false otherwise. You do not need to check whether S and T are sets. (Hint: recall that S is a subset of T if every element of S is a member of T and use `member`.)

**Answer**

```
fun member (x, []) = false
|   member (x, y::ys) = x = y orelse subset (x, ys);

fun subset ([], T) = true
|   subset (x::xs, T) = member (x, T) andalso subset (xs, T);
```

**Problem 6** [40 points–Recursive Functions in ML]

Write a recursive function `makechange (coins, amount)` for the change-making problem. Your function will return a list of values from `coins` that sums up to `amount`. For example, `makechange ([25,10,5,1], 43)` will return `[25,10,5,1,1,1]` and `makechange ([25,10,5,2,1], 44)` will return `[25,10,5,2,2]`. (Note that the available coins in the first example are the common U.S. coins, i.e., quarter, dime, nickel, and penny.) Be greedy: use the largest coin at each step.

**Answer:**

```
fun makechange (coins, 0) = []
|   makechange (c::cs, amt) = if c > amt then makechange (cs, amt)
 else c :: makechange (c::cs, amt - c);

val us = [25, 10, 5, 1];
```

**Problem 7** [30 points–Higher-order Functions in ML].

- Write a (higher-order) function **exists** that takes as arguments a predicate `pred` and a list `L`, and returns true if there is an element `e` in `L` for which `pred(e)` is true, and false otherwise.
- Write a (higher-order) function **all** that takes as arguments a predicate `pred` and a list `L`, and returns true if `pred(e)` is true for all elements `e` in `L`, and false otherwise.

**Answer:**

```
(* Paulson, 185 *)
fun exists pred [] = false
|   exists pred (x::xs) = pred (x) orelse exists pred xs;

fun all pred [] = true
|   all pred (x::xs) = pred (x) andalso all pred xs;

(* Example:
- exists (fn x => x > 2) [1];
val it = false : bool
- exists (fn x => x > 2) [1,2,3,4];
val it = true : bool
-
*)
```

**Problem 8** [45 points–Association Lists in ML]. A *symbol table* is an Abstract Data Type (ADT) consisting of a set of entries, each containing a key and a value. Each key uniquely identifies its entry, and therefore the keys also form a set.

For this exercise, implement the symbol table ADT using an association list. (To refresh your memory, `[(330, "Programming Language Structures"), (350, "Data Structures and Algorithms")]` is an example of association list.)

You need to write ML functions that:

1. Insert an entry into the symbol table. Call this function `insert (key, value, list)`. Your function will insert the pair `(key, value)` in the association list `list`. Check for duplicate keys and raise an exception if that is the case.

2. Check whether the symbol table contains an entry with a given key. Call this function `isPresent (key, list)`. This function will return a Boolean value.

3. Delete an entry with a given key from a symbol table. Call your function `delete (key, list)`. If the entry is not in the symbol table, leave the symbol table unchanged.

**Answer:**

```
(* Operations for the Symbol Table ADT *)

(* Exception Duplicate *)

exception Duplicate;

fun isPresent (k, nil) = false
|   isPresent (k, (x,y)::zs) = k = x orelse isP (k, zs);

fun insert (key, value, list) = if isP (key, list) then raise Duplicate
       else (key, value) :: list;

fun delete (key, []) = []
|   delete (key, (x,y)::zs) = key = x then zs orelse (x,y)::delete(key, zs);
```