CSCE 330 Fall 2012
FINAL EXAM
Closed Book and Notes
Friday 2012-12-14
120 points, excluding 16 extra credit points

1 Syntax and Semantics—20 points

1. (Robert Sebesta–6 points) Consider the following grammar.

```
<S> ::= <A><B><C>
<A> ::= a<A> | a
<B> ::= b<B> | b
<C> ::= c<C> | c
```

   (a) Give a leftmost derivation of `aabcc`

   (b) Give a parse tree of `aabcc`

   (c) Describe, using a single English sentence, the language defined by the grammar.

2. (2 points) What does it mean for a context-free grammar to be *ambiguous*?

3. (8 points) The grammar of the original definition of Algol 60 contained the following production rules:
   `<statement> -> <conditional-statement> | begin <statement> end`
   `<conditional-statement> -> if <condition> then <statement>`
   `| if <condition> then <statement> else <statement>`
   Show that any grammar containing these production rules is ambiguous.

4. (4 points) Match:

   (a) Command

   (b) Declaration

   (c) Expression

   with

   (a) Is evaluated to yield a value.

   (b) Is executed to change the value of a variable or to change the input or output streams.

   (c) Is elaborated to produce a binding, usually to allocate memory, and sometimes to initialize variables.

2 FP–28 points

1. (3 points) Match the FP combining forms to their examples:

   (a) composition

   (b) construction

   (c) apply-to-all (map)

   (a) `& %1`

   (b) `[id, %5]`

   (c) `tl @ [%1, id]`

2. (1 point) Composition and construction (in FP) are examples of

   (a) primitive functions

   (b) control structures

   (c) combining forms

   (Choose one)

3. (1 point) Combining forms are also called higher-order functions, because

   (a) they are closer to the way programmers think than normal functions

   (b) they take other functions as arguments

   (c) their domains and ranges have high dimension

   (Choose one)

4. (4 points) Write a function that multiplies by three the value of its argument plus two. Call it `functionone`. So, for example, `functionone:5` is `21.0`. (The "`.0`" appears if you use Carter Bays's FP interpreter.)

5. (4 points) Write a function that applies `functionone` to all elements of a sequence and give an example of its application to a sequence of three numbers. Do not give a name to the function.

6. (3 points) What is `!+: <1 2 3>`? What do you call ! in FP?

7. (7 points) Write a function that computes the length of a sequence. Do not use recursion. Do not use `while`. Use composition. (Hint: What is `& %1 : <1 2 3>`?)

8. (5 points) Call the function you wrote in the previous exercise `length`. (So, for example, `length: <2 3 4>` is 3.) Write a function that computes the average of a sequence of numbers. Call the function `avg`. For example, `avg: <1 4 4>` is 3.0. (The "`.0`" appears if you use Carter Bays's FP interpreter.)

3 Haskell—88 points

1. (1 point) In Haskell, `[1,2,3]` is an abbreviation for `1 : (2 :(3 : []))`. True or false?

2. (1 point) Here are signatures for two Haskell functions. Which one is curried?

   (a) `add_a :: (Int, Int) -> Int`

   (b) `add_b :: Int -> Int -> Int`

3. (2 point) What is the domain of the type `([a], [a])` in Haskell?

4. (4 points) A recursive function has two parts, the *basis* and the *inductive step*.

   (a) The basis computes the result for sufficiently small arguments, without making any recursive call.

   (b) The inductive step calls the function recursively, with smaller arguments.

   The following recursive function (which is intended to compute factorials) breaks one of these two rules. Which one? In which way?

```
fact  ::  [a] -> [a]
fact n = if n = 0  then 1 else n * fact(n);
```

5. (25 points total) Define functions `length` of one argument that compute the length of a list in five different ways:

   (a) (5 points) a non-recursive function using list comprehension. Do not use a loop. (Name this len1.)

   (b) (5 points) a recursive function with a conditional expression. (Name this len2.)

   (c) (5 points) guarded equations. (Name this len3.)

   (d) (5 points) pattern matching. (Name this len4.)

   (e) (5 points) a non-recursive function using combining forms in FP-style. (Name this len5.)

   For each case, write the type of the function. (You do not need to be most general.)

6. (10 points) Define a function `rev1` of one argument that reverses a list. Use patterns.

7. (25 points) A supermarket sells items as recorded in a database, where each entry in the database is a triple, (BarCode, Name, Price). You have to write two function: the first one takes a list of bar codes into a list of pairs (Name, Price); the second one takes a list of pairs as constructed by the first function and returns the total price of the items in the pairs. Here are the necessary declarations and definitions:

```
type BarCode = Int
type Name = String
type Price = Int

type Database = [(BarCode, Name, Price)]

codeIndex :: Database
codeIndex = [ (4719, "Fish Sticks", 121),
                        (5643, "Diapers", 1010),
                        (3814, "Orange Jelly", 56),
                        (1111, "Hula Hoops", 21),
                        (1112, "Hula Hoops (Giant)", 133),
                        (1234, "Dry Sherry, 1lt", 540)]
```

   (a) (15 points) Define a function `makeBill` that replaces each BarCode in a list of BarCodes with a (Name, Price) pair using the codeIndex database. Also give the type of this function.

   (b) (10 points) Define a function `makeTotal` that adds the prices of each item in a list of (Name, Price) pairs. Also give the type of this function.

8. (20 points total)

   (a) (8 points) Define a function `count` of two arguments that counts the number of times that the first argument occurs in the second argument, which is a list of elements of the type of the first argument, using a list comprehension.

   (b) (2 points) What is the type of this function? (Hint: a class constraint is needed because of equality testing.)

   (c) (8 points) Define a function that counts the number of occurences of the character 'a' in a string. Name your function countA. Here is an example of use:

   ```
   Main> countA "CSCE330 is a great course!"
   2 :: Int
   ```

   `countA` must be defined as a partial application of `count`.

   (d) (2 points) What is the type of `countA`?