PL Midterm

Answer each problem briefly.

1 What are the pro and con of specifying the sizes of primitive types? (4%)

2 Give a non-orthogonal feature of C++. (4%)

3 Give a context-sensitive feature of C++. (4%)

4 What is the usual implementation method for Javascript? for Java? (4%)

5 Consider the task of deleting the ***last*** element of a singly linked list.
What are the time and space complexities of an imperative-style deletion procedure? of a functional-style deletion procedure? (4%)
Note: For the space complexity, count only the auxiliary space and exclude the list itself.

6 The following three Perl statements have similar syntax. But, their semantics are distinct.
What is the semantics of each statement? (4%)

```
$a=(1,2,3,4,5,6);
@a=(1,2,3,4,5,6);
%a=(1,2,3,4,5,6);
```

7 Recall the grammar for C++ expressions:

conditional-expression:
  logical-or-expression
  logical-or-expression ? *expression* : *assignment-expression*
assignment-expression:
  conditional-expression
  logical-or-expression assignment-operator assignment-expression
expression:
  assignment-expression
  expression , *assignment-expression*

a) According to the grammar, what is the associativity of the conditional operator?
What is the associativity of the comma operator? (4%)

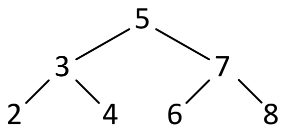
b) Most C++ books *imprecisely* state that the conditional operator has a higher precedence than the assignment operators. Give examples to show that their precedence levels depend on how they are used. Note: Derivations aren't required. (4%)

c)** How would you modify the grammar so that the conditional operator is guaranteed to have a higher precedence than the assignment operators? (4%)

8 The following CFG generates the language of binary trees

T → D | (T D T)
D → 0 | 1 | 2 | … | 9 (i.e. the 10 digits)

Note that a leaf contains only a datum **D**, and an internal node contains a left subtree **T**, a datum **D**, and a right subtree T.
For example, ((2 3 4) 5 (6 7 8)) represents the binary tree



a) Explain why the language of binary trees is not a regular language. (4%)
Hint: Think of examples given in the lecture

b) Suppose that the grammar rule for **T** is rewritten as
T → D | T (D) T
Explain why the resulting grammar is ambiguous. (4%)

c)* Augment this grammar with attributes to accept only binary search trees: (6%)

"A binary search tree is a binary tree with the property that all the data in the left subtree of any node N are less than the datum at node N, and all the data in the right subtree of N are greater than or equal to the datum at node N."

For example, the tree depicted above is a binary search tree.
Hint: Use synthesized attributes
T.min = the minimum datum in the tree generated by **T**
T.max = the maximum datum in the tree generated by **T**
D.value = the datum in the leaf generated by **D**

9 Given

```
(define x '(a b))
(define y (append x (list x)))
```

a) Draw a diagram showing the internal list structures bound to **x** and **y**. (4%)

b) What is the value of each expression below? (4%)

1 `y`

2 `` `(cons ,x ,@x) ``

10 a) Compile the following λ-expression to code consisting of S, K, and I (4%)
λx.λy.x

b) Reduce the following expression with lazy evaluation (4%)
S (S (K +) I) I (* 2 3)

11 a) Let Y = λf.(λx.f(xx))(λx.f(xx)) and G = λy.λf.f(yf)
Show that YG is a fixed point combinator with lazy evaluation. (4%)

b) Write down the corresponding fixed point combinator with eager evaluation. (4%)
Note: Proof isn't required.

12 a) Consider (4%)

```
(define f
    (lambda (n)
        (if (= n 0) 1 (* n (f (- n 1)))))))
```

(The expression `(- n 1)` is boxed.)

Write down a lambda function that represents the continuation of the boxed expression.

b) As discussed in class, by writing a function in continuation-passing style we may control the computation. Give an example to show that such a control over the computation by continuation is useful. (4%)
Note: Code isn't required.

13 Write each function below iteratively ***or*** recursively.

a) Write a function **sum** in Fortran 95 that takes an assumed-shape integer array and sums up the integers of the array. (6%)

Sample run

```
integer, dimension(6) :: a = (/1,2,3,4,5,6/)    ! initialize the array
print *, sum(a)                                 ! output 21
print *, sum(a(2:5))                            ! output 14
```

b) Repeat a), but this time writes the function in Perl. (6%)

Sample run

```
@a=(1..6)          ! initialize the array
print sum @a;      ! output 21
```

c) Repeat a), but this time writes the function in Scheme to sum up a list of integers. (6%)

Sample run

(sum '(1 2 3 4 5 6)) $\Rightarrow$ 21