UC Berkeley – CS164 Midterm 2
Fall 2002

Midterm 2

1. [20 points] Consider the language that consist of possibly empty lists of the identifier x enclosed by parentheses and separated by commas. The language includes `{ () (x) (x,x) (x,x,x) }` etc.

(a) Write a short grammar suitable for use with your LALR parser generator, including augments, so that you accept only these sentences. The sentence `()` should produce an AST that looks like `(ExpList)`. The sentence `(x,x,x)` should look like `(ExpList x x x)`.

(b) Stephen claims this language can be recognized by a DFA with 5 states. Is he right? If so, draw the DFA. If not, explain why this cannot be.

2. [8 points] In assignment 3 you made extensive use of the program

```
(defun aug(&rest r)(if (null (cdr r)) (car r) r))
```

(a) What program calls aug?

(b) When is it called?

(c) What does r get bound to?

(d) What does `(aug '(1 2 3))` return?

3. [16 points] Henry considers implementing the following modifications to the Tiger programming language. For each proposed modification, identify the component(s) of the compiler that Henry will need to change. Select from the lexical analyzer (L), parser (P), type-checker (T), or interpreter (I). It is possible that no change is necessary at all (N). Sometimes more than one may be appropriate or possible. Unless it is utterly clear to you that there is only one possible answer that we are after, we recommend that you write your answer here and also explain on the reverse of this page why you said so.

1. Forbid the type definition let type foo= {}

2. Make the expression () identical to VOID.

3. Make the expression () identical to nil.
4. Allow arithmetic on Boolean results like `(1>0)+(2<3)`.

5. Make the statement `(a<b<c)` mean the same as `(a<b) & (b<c)`.

6. Introduce a type Boolean with two values true and false, such that comparisons result in one of these values rather than 1 or 0.

7. Allow a recursive `x.next:=x` given that x is of `type z= { val=int, next=z }`.

8. Addition of a procedure `shell()` that takes no argument and opens up an interactive shell window, returning an integer "return code" when the `shell` command is done.

4. [10 points] Lee is building an implementation of Tiger (not necessarily built on top of Lisp). Lee needs to consider storage management, and needs your answers to the following questions in complete English sentences.

1. At what point in a Tiger computation would a garbage collection likely be started? Hint: you ordinarily do a garbage collection when you need more memory but don't seem to have enough free. Even bigger hint: What kinds of statements in Tiger allocate heap space?

2. What kinds of statements in Tiger allocate space on a stack?

3. What kinds of statements in Tiger de-allocate space?

4. If you wish to provide explicit heap memory management, what changes would you have to make to Tiger?

5. Any discussion of garbage collection refers to the roots needed to start the operation. In a Tiger implementation using garbage collection, what are the roots?

5. [9 points]

a. How does the Tiger interpreter discussed in class deal with logical "and" and "or" expressions such as `A&B|C` ?

b. Typechecking comparison (such as >) is more elaborate in Tiger than typechecking addition +. Explain why.

c. What do you know about the type of x give than the expression `if x=nil then 1 else 2` passes a semantic check?

6. [6 points] Explain in complete English sentences what steps are necessary in the Tiger interpreter to interpret a Tiger expression "call" such as `F(3,a)`.

7. [16 points] For each of the following Tiger programs, determine where the processing will detect an error. If possible, indicate a location in the program text.

- Mark the location L if it will fail in Lexical Analysis.
- Mark the location P if it will fail in Parsing.
- Mark the location T if it will fail Type Checking.
- If there are no errors, write OK.

Give some explanation as to why the failure occurs. If you think there are several errors such that if you correct the first one, there are still others, mention them all. If you are not sure what will happen, explain that too, and maybe we can give you some partial credit.

```
--01---
let type t1={a:int,b:t1}
type t2={c:t1}
var x:t2:=nil
in x.c.b.a end

--02----
let type t1=int
var x:t1
in x:= 3 end

--03---
let var x:=5 in x:=3+4^5 end

--04---
let type a= {x:int, y:int}
type b= {x:int, y:int}
type c=a
var i:a :=(x=1,y=2}
var j:b :=nil
var k:c :=nil
in b:=j; k:=j end

--05---
let
type intlist ={hd:int, tl:intlist}
type b=c
type c=b
in 1234 end

--06---
let
type intarray = array of intarray
var M:intarray := array[10] of M
in 5676 end


--07---
let function f(a:int):int=3*a*a end
in print(f(2)); "done" end
```

```
--08---
3+4*(("big"<"little")>0)-9)
```

8. [4 points] What is the amortized cost (that is, the cost per reclaimed cell) of a 2-space copying garbage collector process in which, at time *t* a garbage collection is begun, there are *R* live nodes and a total of *H* cells in the heap.

a. Explain any additional symbols you introduce using complete English sentences.

b. What happens if $R = H/2$?

9. [11 points] In the Leopard programming language, a language similar in some respects to Tiger, the following program is legal.

```
let
function q(a,b)= let in a:=3;b:=7 end
type ar=array of int
var m:=ar [6] of 0
var i:=2
in
q(i, m[i+2]);
for i:=0 to 5 do print(chr(m[i]+ ord("0")))
end
```

a. If Leopard has call-by-reference parameter passing, what are the six values in the array m when they are printed?

b. If Leopard has call-by-name parameter passing, what are the six values in the array m?

c. If Leopard has the same parameter passing as Tiger, what are the six values in the array m?

d. Why did we not write `print(m[i])`?

e. What prevents this program from being legal Tiger?