Homework 3

Due 22 October

Handout 5
CS242: Autumn 2002
15 October

Reading

1. Read chapters 5 (ML) and 6 (Types) of the text.

Problems

1. Algol 60 Procedure Types

In Algol 60, the type of each formal parameter of a procedure must be given. However, *proc* is considered a type (the type of procedures). This is much simpler than the ML types of function arguments. However, this is really a type loophole; since calls to procedure parameters are not fully type checked, Algol 60 programs may produce run-time type errors.

Write a procedure declaration for `Q` which causes the following program fragment to produce a run-time type error.

```
proc P (proc Q)
   begin Q(true) end;
P(Q);
```

where `true` is a boolean value. Explain why the procedure is statically type correct, but produces a run-time type error. (You may assume that adding a boolean to an integer is a run-time type error.)

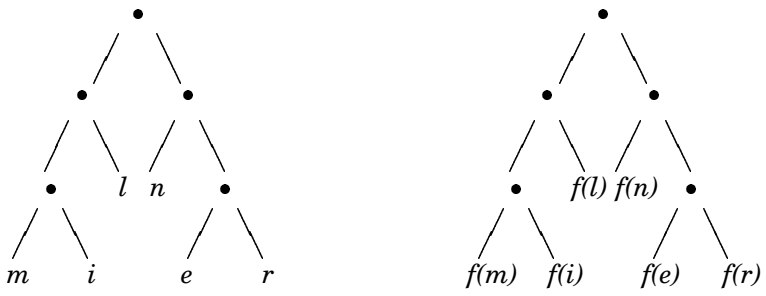
2. ML Map for Trees

(a) The binary tree datatype

```
datatype 'a tree  =  LEAF of 'a |
                     NODE of 'a tree * 'a tree;
```

describes a binary tree for any type, but does not include the empty tree (i.e., each tree of this type must have at least a root node).

Write a function `maptree` that takes a function as an argument and returns a function that maps trees to trees, by mapping the values at the leaves to new values, using the function passed in as a parameter. In more detail, if `f` is a function that can be applied to the leaves of tree `t`, and `t` is the tree on the left, then `maptree f t` should result in the tree on the right.



For example, if `f` is the function `fun f(x)=x+1` then
`maptree f (NODE(NODE(LEAF 1,LEAF 2),LEAF 3))`
should evaluate to `NODE(NODE(LEAF 2,LEAF 3),LEAF 4)`. Explain your definition in one or two sentences.

(b) What is the type ML gives to your function? Why isn't it the expected type `('a → 'a) → 'a tree → 'a tree`?

3. Disjoint Unions

A *union type* is a type that allows the values from two different types to be combined in a single type. For example, an expression of type `union(A, B)` might have a value of type `A` or a value of type `B`. The languages C and ML both have forms of union types.

(a) Here is a C program fragment written using a union type.

```
...
union IntString {
  int i;
  char *s;
} x;
int y;
if ( ... ) x.i = 3 else x.s = "here, fido";
...
y = (x.i) + 5;
...
```

A C compiler will consider this program to be well-typed. Despite the fact that the program type-checks, the addition may not work as intended. Why not? Will the run-time system catch the problem?

(b) In ML, a union type `union(A,B)` would be written in the form `datatype UnionAB = tag_a of A | tag_b of B` and the `if` statement above could be written

```
datatype IntString = tag_int of int | tag_str of string;
...
val x = if ... then tag_int(3) else tag_str("here, fido");
...
let val tag_int (m) = x in m + 5 end;
```

Can the same bug occur in this program? Will the run-time system catch the problem? The use of tags enables the compiler to give a useful warning message to the programmer, thereby helping the programmer to avoid the bug, even before running the program. What message is given and how does it help?

4. ML Types

Explain the ML type for each of the following declarations:

(a) `fun a(x) = 3*x;`

(b) `fun b(x,y) = x/2.0 * y/3.0;`

(c) `fun c(f,g) = fn x => f(g(x));`

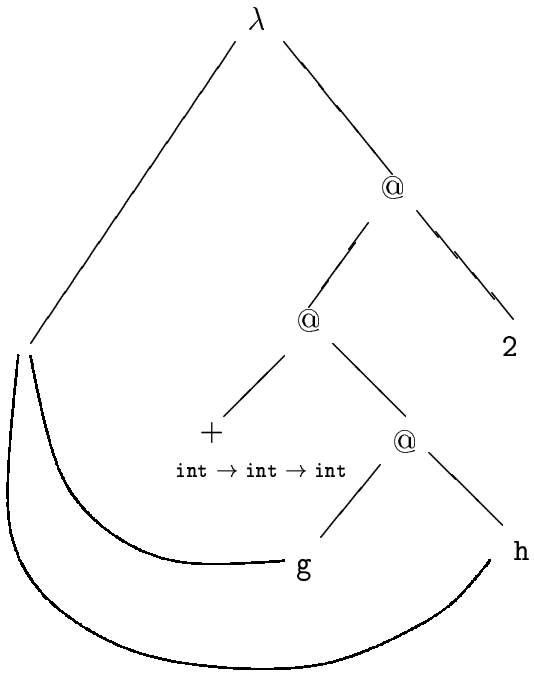
(d) `fun d(b,f,x,y) = if b(y) then f(x) else y;`

Since you can simply type these expressions into an ML interpreter to determine the type, be sure to write a short *explanation* to show that you understand why the function has the type you give.

5. ..... Parse Graph

Use the parse graph below to calculate the ML type for the function

```
fun f(g,h) = g(h) + 2;
```

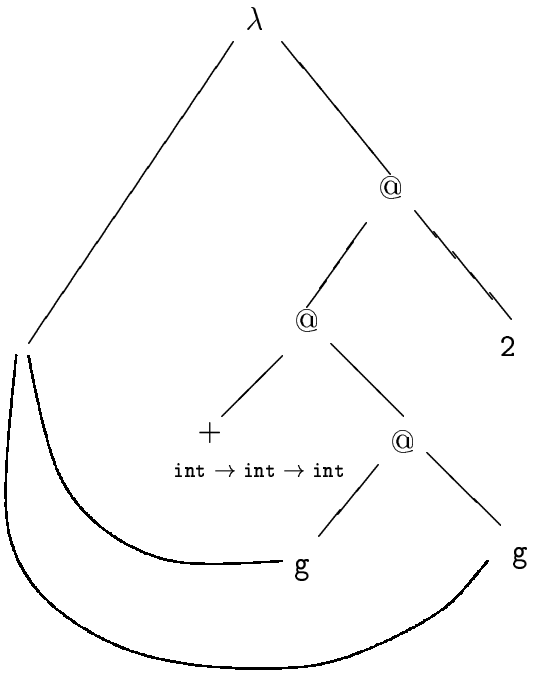


6. ..... Parse Graph

Use the parse graph below to follow the steps of the ML type inference algorithm on the function declaration

```
fun f(g) = g(g) + 2;
```

What is the output of the type checker?



7. ..... Type Inference and Debugging

The `reduce` function takes a binary operation, in the form of a function `f`, and a list, and produces the result of combining all elements in the list using the binary operation. For example:

$$\texttt{reduce plus } [1,2,3] \quad = \quad 1+2+3 \quad = \quad 6$$

if `plus` is defined by

$$\texttt{fun plus } (\texttt{x},\texttt{y} : \texttt{int}) \quad = \texttt{x} + \texttt{y}$$

A friend of yours is trying to learn ML and tries to write a `reduce` function. Here is his incorrect definition.

```
fun reduce(f, x) = x
|   reduce(f, (x::y)) = f(x, reduce(f,y));
```

He tells you that he doesn't know what to return for an empty list, but this should work for a nonempty list: if the list has one element, then the first clause returns it. If the list has more than one element, then the second clause of the definition uses the function `f`. This sounds like a reasonable explanation, but the type checker gives you the following output:

```
val reduce = fn : ((('a * 'a list) -> 'a list) * 'a list) -> 'a list
```

How can you use this type to explain to your friend that his code is wrong?