

Homework 4

COSE212

Hakjoo Oh

Problem 1 Consider the language ML^- from HW3:

$P \rightarrow$	E	
$E \rightarrow$	$()$	unit
	$ \text{ true } \text{ false }$	booleans
	$ n$	integers
	$ x$	variables
	$ E + E \mid E - E \mid E * E \mid E / E$	arithmetic
	$ E = E \mid E < E$	comparison
	$ \text{ not } E$	negation
	$ \text{ nil }$	empty list
	$ E :: E$	list cons
	$ E @ E$	list append
	$ \text{ head } E$	list head
	$ \text{ tail } E$	list tail
	$ \text{ isnil } E$	checking empty list
	$ \text{ if } E \text{ then } E \text{ else } E$	if
	$ \text{ let } x = E \text{ in } E$	let
	$ \text{ letrec } f(x) = E \text{ in } E$	recursion
	$ \text{ letrec } f(x_1) = E_1 \text{ and } g(x_2) = E_2 \text{ in } E$	mutual recursion
	$ \text{ proc } x E$	function definition
	$ E E$	function application
	$ \text{ print } E$	print
	$ E; E$	sequence

In OCaml datatype:

```

type program = exp
and exp =
  | UNIT
  | TRUE
  | FALSE
  | CONST of int
  | VAR of var

```

```

| ADD of exp * exp
| SUB of exp * exp
| MUL of exp * exp
| DIV of exp * exp
| EQUAL of exp * exp
| LESS of exp * exp
| NOT of exp
| NIL
| CONS of exp * exp
| APPEND of exp * exp
| HEAD of exp
| TAIL of exp
| ISNIL of exp
| IF of exp * exp * exp
| LET of var * exp * exp
| LETREC of var * var * exp * exp
| LETMREC of (var * var * exp) * (var * var * exp) * exp
| PROC of var * exp
| CALL of exp * exp
| PRINT of exp
| SEQ of exp * exp
and var = string

```

Types for the language are defined as follows:

```

type typ =
  TyUnit
| TyInt
| TyBool
| TyFun of typ * typ
| TyList of typ
| TyVar of tyvar
and tyvar = string

```

Implement a sound type checker, `typeof`, for the language (the notion of soundness is defined with respect to the dynamic semantics of the language defined in HW3):

`typeof : exp -> typ`

which takes a program and returns its type if the program is well-typed. When the program is ill-typed, `typeof` should raise an exception `TypeError`.