CIS425 - Midterm Exam

Your Name : ____________________________________

Your Student ID : ______________________________

(10 points) Consider the grammar $G = (T, NT, R, P)$ where $T = \{e, f, g\}$, $NT = \{S, T, U\}$, $R = S$, and P is given as follows:

$$\begin{array}{l} S ::= e\,S|T \\ T ::= f\,T|U \\ U ::= g\,U|\epsilon \end{array}$$

1. What is $L(G)$?

2. Is the grammar regular, context free or context sensitive? Briefly explain your answer

3. Give a parse tree of a string belonging to the language

(10 points) Consider the following grammar G:

$$E ::= E + E \mid E * E \mid (E) \mid a \mid b$$

1. Show that the grammar is ambiguous

2. Rewrite the grammar into a non ambiguous grammar reflecting the fact that $+$ and $*$ are right associative and $*$ has lower precedence than $+$

3. Briefly explain with an example the difference between a parse tree and an abstract syntax tree

(4 points) Draw the abstract syntax tree of the lambda expression $\lambda x.(y\ z\ w)$

(6 points) Lambda-calculus expressions are represented by the following datatype

```
datatype M = Var of string | App of M*M | lam of string*M
```

Represent the lambda-expression $\lambda x.(y\ z\ w)$ as a value of type M.

(9 points) Give all the free and bound variables in the following lambda-expressions. If a variable is bound draw a line between the variable and its definition.

1. $\lambda x.x\ z\ \lambda y.x\ y$

2. $(\lambda x.x\ z)\ \lambda y.w\ \lambda w.w\ y\ z\ x$

3. $\lambda x.x\ y\ \lambda x.y\ x$

(6 points) Define the function FV which returns the set of free variables occurring in a lambda-expression. You do not need to use ML code, define it as we did in class.

$$FV(x) \quad =$$

$$FV(\lambda x.M) \quad =$$

$$FV(M\ N) \quad =$$

(5 points) Explain using an example what is the variable capture problem

(6 points) Apply β-reduction to the following expressions as much as possible :

1. $(\lambda y.y)(\lambda x.\lambda z.z)w$

2. $(\lambda x.x\ \lambda y.y\ x)\ y$

3. $\lambda x.(\lambda y.y\ y)\ w\ z$

(2 points) Can I α-convert the expression $(\lambda x.\lambda y.x\ y)\ y$ to $(\lambda x.\lambda y.x\ y)\ z$? Explain briefly why or why not.

(2 points) In the expression $(\lambda x.\lambda y.x\ y)\ y$, can I rename variable x to y ? Explain briefly why or why not.

(10 points) Consider the following simple language E of arithmetic:

```
E ::= num | E + E | E * E
```

where num is any integer. We represent strings of E using a corresponding datatype:

```
datatype E = NUM of int | PLUS of E * E | TIMES of E * E
```

1. Write $(5 * 3) + 9$ as a value of type E.

2. Write a function `interp` that accepts as input a program of type E, interprets it, and returns the integer result. For example:

```
- interp (PLUS (NUM 1, NUM 2));
val it = 3 : int

fun interp
|   interp
|   interp
```

(10 points) Write a function `foldl` (also called `reduce`) with type:

```
foldl: ('a * 'b -> 'b)  -> 'b ->  'a list -> 'b
```

The type `('a * 'b -> 'a)` is the type of the input function (say `f`), `'b` is the type of an initial value, and `'a list` is the type of the input list.

```
foldl(f,init,[x1,...xn])
```

returns

```
f(xn,...,f(x2, f(x1,init))...)
```

For example, we have:

```
- fun plus (x,y) = x + y + 0  ;
- foldl (plus, 0, [1,2,3,4]) ;
  val it = 10 : int

  fun foldl (f: 'a*'b->'b) (init: 'b) (l: 'a list): 'b =
  case l of
```

(5 points) Briefly explain what the following function is doing:

```
 fun foo l1 x = foldl (fn (hd, acc) => if hd = x then acc else hd::acc) [] l1
 foo :  'a list -> 'a -> 'a list
```

For example, what is the result of `foo ["a","b","c"] "b"`?

(5 points) Briefly explain what the following function is doing:

```
fun foo l1 l2 = foldl (fn (hd, acc) => if (exists l2 hd) then acc else hd::acc) [] l1
foo: 'a list -> 'a list -> 'a list
```

assuming that function `exists` is defined as

```
fun exists [] n = false
|   exists (x :: xs) n = if n = x then true else exists xs n
```

For example, what is the result of `foo ["a","b","c"] ["b","c"]`?