Sample Questions for Midterm 2 (CS 421 Fall 2019)

Some of these questions may be reused for the exam.

0. Review and be able to write any give clause of **cps_exp** form MP5. On the exam, you would be given all the information you were given in MP5.

1. Write the definition of an OCAML variant type **reg_exp** to express abstract syntax trees for regular expressions over a base character set of booleans. Thus, a boolean is a **reg_exp,** epsilon is a **reg_exp**, a parenthesized **reg_exp** is a **reg_exp,** the concatenation of two **reg_exp**'s is a **reg_exp,** the "choice" of two **reg_exp**'s is a **reg_exp**, and the Kleene star of a **reg_exp** is a **reg_exp**.
2. Given the following OCAML datatype:

   type int_seq = Null | Snoc of (int_seq * int)

   write a tail-recursive function in OCAML **all_pos : int_seq -> bool** that returns **true** if every integer in the input **int_seq** to which **all_pos** is applied is strictly greater than 0 and **false** otherwise. Thus **all_pos (Snoc(Snoc(Snoc(Null, 3), 5), 7))** should returns **true,** but) **all_pos (Snoc(Null, -1))** and **all_pos (Snoc(Snoc(Null, 3),0))** should both return **false.**
3. Given a polymorphic type derivation for {} |- let id = fun x -> x in id id true : bool
4. Write the clause for **gather_exp_ty_substitution** for a function expression implementing the rule:

$$\frac{[x : \tau_1] + \Gamma \vdash e : \tau_2 \mid \sigma}{\Gamma \vdash (\textbf{fun}\ x \rightarrow e) : \tau \mid \underline{\text{unify}}\{(\sigma(\tau), \sigma(\tau_1 \rightarrow \tau_2))\} \circ \sigma}$$

   Refer to MP6 for the details of the types. You should assume that all other clauses for **gather_exp_ty_substitution** have been provided.
5. Give a (most general) unifier for the following unification instance. Capital letters denote variables of unification. Show your work by listing the operation performed in each step of the unification and the result of that step.

   {X = f(g(x),W); h(y) = Y; f(Z,x) = f(Y,W)}
6. For each of the following descriptions, give a regular expression over the alphabet {a,b,c}, and a regular grammar that generates the language described.
   a. The set of all strings over {**a**, **b**, **c**}, where each string has at most one **a**

   b. The set of all strings over {**a**, **b**, **c**}, where, in each string, every **b** is immediately followed by at least one **c**.

   c. The set of all strings over {**a**, **b**, **c**}, where every string has length a multiple of four.