Sample Questions for Midterm 2 (CS 421 Spring 2023)

Some of these questions may be reused for the exam.

1. Put the following function in full continuation passing style:
   ```ocaml
   let rec sum_odd n = if n <= 0 then 0 else ((2 * n) – 1) + sum_odd (n – 1);
   ```
   Use addk, subk, mulk, leqk, for the CPS forms of the primitive operations (+, -, *, <=).
   All other procedure calls and constructs must be put in CPS.

2. Review and be able to write any give clause of \texttt{cps\_exp} from MP5. On the exam, you would be given all the information you were given in MP5.

3. Given the following rules for CPS transformation:
   \[
   [\text{let} \ x = e1 \ \text{in} \ e2] \ K = \ [\text{FN} \ a \rightarrow [\text{FN} \ b \rightarrow \ K \ (b \oplus \ a)])
   \]
   where \(e1\) and \(e2\) are OCaml expressions, \(K\) is any continuation, \(x\) is a variable and \(c\) is a constant, give the step-by-step transformation of
   \[
   \text{[[\text{let} \ x = 2 + 3 \ \text{in} \ x – 4]] REPORTk}
   \]

4. Write the definition of an OCAML variant type (algebraic data type) \texttt{reg\_exp} to express abstract syntax trees for regular expressions over a base character set of booleans. Thus, a boolean is a \texttt{reg\_exp}, epsilon is a \texttt{reg\_exp}, a parenthesized \texttt{reg\_exp} is a \texttt{reg\_exp}, the concatenation of two \texttt{reg\_exp}'s is a \texttt{reg\_exp}, the “choice” of two \texttt{reg\_exp}'s is a \texttt{reg\_exp}, and the Kleene star of a \texttt{reg\_exp} is a \texttt{reg\_exp}.

5. Given the following OCAML datatype:
   ```ocaml
   type int_seq = Null | Snoc of (int_seq * int)
   ```
   write a tail-recursive function in OCAML \texttt{all\_pos : int\_seq \rightarrow bool} that returns \texttt{true} if every integer in the input \texttt{int\_seq} to which \texttt{all\_pos} is applied is strictly greater than 0 and \texttt{false} otherwise. Thus \texttt{all\_pos (Snoc(Snoc(Snoc(NULL, 3), 5), 7))} should returns \texttt{true}, but \texttt{all\_pos (Snoc(NULL, -1))} and \texttt{all\_pos (Snoc(Snoc(NULL, 3),0))} should both return \texttt{false}.

6. Give a polymorphic type derivation for \{\} |- \texttt{let id = fun x -> x in id id true : bool}

7. Write the clause for \texttt{gather\_exp\_ty\_substitution} for a function expression implementing the rule:
   \[
   \frac{[x : \tau_1] + \Gamma \vdash e : \tau_2 \mid \sigma}{\Gamma \vdash (\text{fun} \ x \rightarrow e) : \tau \mid \text{unify} \{\sigma(\tau), \sigma(\tau_1 \rightarrow \tau_2)\} \circ \sigma}
   \]
   Refer to MP6 for the details of the types. You should assume that all other clauses for \texttt{gather\_exp\_ty\_substitution} have been provided.