

Sample Questions for Midterm 2 (CS 421 Fall 2025)

Some of these questions may be reused for the exam.

1. Put the following function in full continuation passing style:

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. 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 return **true**, but **all_pos (Snoc(Null, -1))** and **all_pos (Snoc(Snoc(Null, 3), 0))** should both return **false**.

3. Write the definition of an OCAML variant type (algebraic data 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**.

4. Given the following rules for CPS transformation:

$[[x]] K \Rightarrow K x$

$[[c]] K \Rightarrow K c$

$[[\text{let } x = e1 \text{ in } e2]] K \Rightarrow [[e1]] (\text{FN } x \rightarrow [[e2]] K)$

$[[e1 \oplus e2]] K \Rightarrow [[e2]] (\text{FN } a \rightarrow [[e1]] (\text{FN } b \rightarrow K (b \oplus a)))$

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

$[[\text{let } x = 2 + 3 \text{ in } x - 4]] \text{REPORT} K$

5. Review and be able to write any give clause of **cps_exp** from MP5. On the exam, you would be given all the information you were given in MP5.
6. Give a polymorphic type derivation for $\{\} \mid \text{let id} = \text{fun } x \rightarrow x \text{ in id id true} : \text{bool}$