CS 6371	**Advanced Programming Languages**	SAMPLE
SPRING 2024		FINAL EXAM

This sample final exam is LONGER than a real final exam (to give you more practice problems) and has low-to-medium difficulty problems (to help you identify and practice foundational skills). You may take two, two-sided sheets of notes with you into the exam. All other books or notes must remain closed throughout the exam. You will have 2 hours and 45 minutes to complete the exam.

1 Problem Set

(1) A *metric* is a function $m : \alpha \to \alpha \to \texttt{int}$ that computes some notion of distance between two values. The *path-length* of a list is the sum of the distances between each consecutive pair of elements. For example, if the data has type $\alpha = \texttt{int}$ and the metric is absolute difference, then the path-length of list $[7; 10; 6]$ is $|10 - 7| + |6 - 10| = 7$.

Using only `List.fold_left` for recursion, implement a function (`pathlen` m ℓ) that computes the path-length of ℓ using metric function m. If ℓ has less than 2 elements, the path-length is 0. Do not use any other List library functions in your implementation.

(2) Polish mathematician Jan Łukasiewicz once reduced all of classical propositional logic to an extremely simple language with only two operators, one rule of inference, and three axioms:

$$p ::= v \mid \neg p \mid p_1 \Rightarrow p_2$$

$$\frac{p_1 \Rightarrow p_2 \quad p_1}{p_2}\text{(I1)} \qquad \frac{}{(\neg p_1 \Rightarrow \neg p_2) \Rightarrow (p_2 \Rightarrow p_1)}\text{(A1)}$$

$$\frac{}{p_1 \Rightarrow (p_2 \Rightarrow p_1)}\text{(A2)} \qquad \frac{}{(p_1 \Rightarrow (p_2 \Rightarrow p_3)) \Rightarrow ((p_1 \Rightarrow p_2) \Rightarrow (p_1 \Rightarrow p_3))}\text{(A3)}$$

(a) Implement a Prolog predicate `provable(P,N)` that succeeds if and only if predicate `P` is provable via a derivation consisting of the above derivation rules whose height is at most `N`, where `N` is a LOGICAL encoding of a natural number. When `N`=0, only the axioms (A1–A3) are provable. To model propositional sentences in Prolog, use Prolog atoms `v` for propositional variables v, use structure `neg(`p`)` for $\neg p$, and use structure `imp(`p_1`,`p_2`)` for $p_1 \Rightarrow p_2$.

(b) Using your solution to part a, write a Prolog predicate `proofsearch(P)` that succeeds if `P` is provable with a derivation of any height, but that fails or loops otherwise.

(3) Implement merge-sort in Prolog.

(4) For each of the following System F types, say whether the type is inhabited or not. If the type is inhabited, give an example of a System F term that inhabits it. (Do not prove that your term inhabits the type, just state it.) If the type is not inhabited, just write "uninhabited".

(a) $\forall\alpha\,.\,\forall\beta\,.\,\forall\eta\,.\,((\alpha \times \beta) \to \eta) \to (\alpha \to \beta \to \eta)$

(b) $\forall\alpha.(\alpha + \mathit{unit})$

(c) $\forall\alpha.\forall\beta.(\alpha+\beta)\rightarrow(\alpha\times\beta)$

(d) $\forall\alpha.\forall\beta.((\alpha+\beta)\rightarrow((\alpha\rightarrow\beta)+(\beta\rightarrow\alpha)))$

(5) Encode an *iseven* function in the untyped λ-calculus so that $(\mathit{iseven}\ n_{\mathbb{N}})$ evaluates to *true* whenever n is even and to *false* whenever n is odd.

(6) Consider the untyped λ-calculus expression *foo* defined as follows:

$$\mathit{foo} = Y\big(\lambda f.\lambda x.((\mathit{iszero}\ x)\ ?\ x\ :\ (f\ (\mathit{pred}_{\mathbb{N}}\ x)))\big)$$

Prove by fixed-point induction that $P(\mathit{foo})$ holds, where P is the property defined by

$$P(g) \equiv \forall(x_{\mathbb{N}}, y_{\mathbb{N}}) \in g\ .\ y_{\mathbb{N}} = 0_{\mathbb{N}}$$

In your proof when you claim that an expression e_1 evaluates to another expression e_2, you may do so without a formal proof of $e_1 \rightarrow^* e_2$. That is, it is not necessary to write out all the small-step derivations.

(7) Derive the following typing judgment using the typing rules for the simply-typed λ-calculus:

$$\{\} \vdash (\lambda x{:}\mathit{int}\,.\,x)\,3\ :\ \mathit{int}$$

(8) Derive the following partial correctness assertion using Hoare Logic:

$$\{x=\bar{n}\}\texttt{while }x\texttt{<=}-1\texttt{ do }x\texttt{:=}x+1\{x=\mathit{max}(\bar{n},0)\}$$

2 Solutions

(1)
```
let pathlen m = function [] -> 0 | h::t ->
  fst (List.fold_left (fun (s,p) x -> (s+(m p x),x)) (0,h) t);;
```

(2) **(a)**
```
provable(P2,s(N)) :- provable(imp(P1,P2),N), provable(P1,N).
provable(imp(imp(neg(P1),neg(P2)),imp(P2,P1)),_).
provable(imp(P1,imp(_,P1)),_).
provable(imp(imp(P1,imp(P2,P3)),imp(imp(P1,P2),imp(P1,P3))),_).
```

(b)
```
isnum(0).
isnum(s(N)) :- isnum(N).
proofsearch(P) :- isnum(N), provable(P,N).
```

(3)
```
split([],[],[]).
split([X],[X],[]).
split([X,Y|T],[X|T1],[Y|T2]) :- split(T,T1,T2).

merge(L,[],L).
merge([],L,L).
merge([H1|T1],[H2|T2],[H1|T]) :- H1 @=< H2, merge(T1,[H2|T2],T).
merge([H1|T1],[H2|T2],[H2|T]) :- H1 @> H2, merge([H1|T1],T2,T).
```

```
msort([],[]).
msort([X],[X]).
msort([X,Y|T],Out) :- split([X,Y|T],L1,L2), msort(L1,S1), msort(L2,S2),
                      merge(S1,S2,Out).
```

(4) **(a)** $\Lambda\alpha.\Lambda\beta.\Lambda\eta.\lambda f{:}((\alpha \times \beta) \to \eta).\lambda x{:}\alpha.\lambda y{:}\beta.f(x,y)$

(b) $\Lambda\alpha\,.\,\mathtt{in}_2^{\alpha+unit}()$

(c) uninhabited

(d) The type is inhabited. The following is a term that inhabits it:

$$\begin{aligned}\Lambda\alpha.\Lambda\beta.\lambda x{:}\alpha{+}\beta\,.\,\mathtt{case}\ x\ \mathtt{of}\ \mathtt{in}_1(y) &\to \mathtt{in}_2^{(\alpha\to\beta)+(\beta\to\alpha)}\ (\lambda z{:}\beta.y)\\ \mid\ \mathtt{in}_2(y) &\to \mathtt{in}_1^{(\alpha\to\beta)+(\beta\to\alpha)}\ (\lambda z{:}\alpha.y)\end{aligned}$$

(5) The *iseven* function can be encoded this way:

$$\begin{aligned}iseven = Y(\lambda f.\lambda n.((iszero_{\mathbb{N}}\ n)\ ?\ true\ &:\\ ((iszero_{\mathbb{N}}\ (pred_{\mathbb{N}}\ n))\ ?\ false\ &:\\ (f\ (pred_{\mathbb{N}}\ (pred_{\mathbb{N}}\ n))))))&\end{aligned}$$

(6) *Proof.* Define functional Γ by

$$\Gamma(f) = \lambda x.((iszero_{\mathbb{N}}\ x)\ ?\ x\ :\ (f\ (pred_{\mathbb{N}}\ x)))$$

Since $foo = Y\Gamma = fix(\Gamma)$, we can prove the theorem by fixed-point induction on Γ.

Base Case: $P(\bot)$ holds vacuously.

Inductive Case: We must prove that $P(g)$ implies $P(\Gamma(g))$. Therefore, assume $P(g)$ holds and let $(x_{\mathbb{N}}, y_{\mathbb{N}}) \in \Gamma(g)$ be given. We wish to prove that $y_{\mathbb{N}} = 0_{\mathbb{N}}$.

Case 1: Suppose $x_{\mathbb{N}} = 0_{\mathbb{N}}$. By the definition of Γ, $\Gamma(x_{\mathbb{N}}) = x_{\mathbb{N}} = 0_{\mathbb{N}}$, so $y_{\mathbb{N}} = 0_{\mathbb{N}}$.

Case 2: Suppose $x_{\mathbb{N}} \neq 0_{\mathbb{N}}$. Then by definition of Γ, $y_{\mathbb{N}} = (g\ (pred_{\mathbb{N}}\ x_{\mathbb{N}})) = (g\ (x-1)_{\mathbb{N}})$. This is the same as saying $((x-1)_{\mathbb{N}}, y_{\mathbb{N}}) \in g$. Since we assumed $P(g)$ holds, it follows that $y_{\mathbb{N}} = 0_{\mathbb{N}}$. □

(7) The following typing derivation proves the typing judgment:

$$\dfrac{\dfrac{\dfrac{}{\{(x, int)\} \vdash x : int}(10)}{\{\} \vdash (\lambda x{:}int.x) : int \to int}(11) \qquad \dfrac{}{\{\} \vdash 3 : int}(9)}{\{\} \vdash (\lambda x{:}int.x)3\ :\ int}(12)$$

(8) Choose loop invariant $I = ((x \le 0) \vee (x = \bar{n})) \wedge (x \ge \bar{n})$ and derive the following:

$$\dfrac{\models A \Rightarrow I \qquad \dfrac{\dfrac{\models I \wedge b \Rightarrow C \qquad \dfrac{}{\{C\}x{:=}x+1\{I\}}(4) \qquad \models I \Rightarrow I}{\{I \wedge b\}x{:=}x+1\{I\}}(6)}{\{I\}p\{\neg b \wedge I\}}(5) \qquad \models \neg b \wedge I \Rightarrow B}{\{A\}p\{B\}}(6)$$

where

$$\begin{aligned}
p &= \texttt{while}\ x \texttt{<=} -1\ \texttt{do}\ x \texttt{:=} x \texttt{+} 1 \\
b &\equiv (x \leq -1) \\
I &\equiv ((x \leq 0) \vee (x = \bar{n})) \wedge (x \geq \bar{n}) \\
A &\equiv (x = \bar{n}) \\
B &\equiv (x = max(\bar{n}, 0)) \\
C &\equiv I[x+1/x] \equiv ((x+1 \leq 0) \vee (x+1 = \bar{n})) \wedge (x+1 \geq \bar{n})
\end{aligned}$$

3 Reference

In addition to the material in this reference section, you will also be provided any relevant material from the reference section of the sample midterm exam.

3.1 Syntax of SIMPL

commands	$c ::= \texttt{skip} \mid c_1;c_2 \mid v\texttt{:=}a \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } b \texttt{ do } c$
boolean expressions	$b ::= \texttt{true} \mid \texttt{false} \mid a_1\texttt{<=}a_2 \mid b_1 \,\texttt{\&\&}\, b_2 \mid b_1 \,\texttt{\|\|}\, b_2 \mid \texttt{!}b$
arithmetic expressions	$a ::= n \mid v \mid a_1 \texttt{+} a_2 \mid a_1 \texttt{-} a_2 \mid a_1 \texttt{*} a_2$
variable names	v
integer constants	n

3.2 Axiomatic Semantics of SIMPL

$$\{A\}\texttt{skip}\{A\} \tag{1}$$

$$\frac{\{A\}c_1\{C\} \qquad \{C\}c\{B\}}{\{A\}c_1;c_2\{B\}} \tag{2}$$

$$\frac{\{A \wedge b\}c_1\{B\} \qquad \{A \wedge \neg b\}c_2\{B\}}{\{A\}\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2\{B\}} \tag{3}$$

$$\{B[a/v]\}v\texttt{:=}a\{B\} \tag{4}$$

$$\frac{\{I \wedge b\}c\{I\}}{\{I\}\texttt{while } b \texttt{ do } c\{\neg b \wedge I\}} \tag{5}$$

$$\frac{\models A \Rightarrow A' \qquad \{A'\}c\{B'\} \qquad \models B' \Rightarrow B}{\{A\}c\{B\}} \tag{6}$$

3.3 Untyped Lambda Calculus

3.3.1 Syntax and Semantics of Untyped λ-calculus

$$e ::= v \mid \lambda v.e \mid e_1 e_2$$

$$\frac{e_1 \rightarrow_1 e_1'}{e_1 e_2 \rightarrow_1 e_1' e_2} \tag{7}$$

$$(\lambda v.e_1)e_2 \rightarrow_1 e_1[e_2/v] \tag{8}$$

3.3.2 Abbreviations in Untyped λ-calculus

$$\begin{aligned}
\mathit{true} &= (\lambda x.\lambda y.x) \\
\mathit{false} &= (\lambda x.\lambda y.y) \\
e_1?e_2{:}e_3 &= (e_1e_2e_3) \\
\mathit{not} &= (\lambda b.(b?\mathit{false}{:}\mathit{true})) \\
\mathit{and} &= (\lambda a.\lambda b.(a?b{:}\mathit{false})) \\
\mathit{or} &= (\lambda a.\lambda b.(a?\mathit{true}{:}b)) \\
Y &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))
\end{aligned}$$

$$\begin{aligned}
\mathit{pair} &= (\lambda x.\lambda y.\lambda b.(b?e_1{:}e_2)) \\
\pi_1 &= (\lambda x\,.\,x\ \mathit{true}) \\
\pi_2 &= (\lambda x\,.\,x\ \mathit{false}) \\
0_{\mathbb{N}} &= (\lambda x.x) \\
\mathit{succ}_{\mathbb{N}} &= (\mathit{pair}\ \mathit{false}) \\
\mathit{pred}_{\mathbb{N}} &= \pi_2 \\
\mathit{iszero}_{\mathbb{N}} &= \pi_1
\end{aligned}$$

$$\begin{aligned}
\mathit{add}_{\mathbb{N}} &= (Y(\lambda f.\lambda m.\lambda n.((\mathit{iszero}_{\mathbb{N}}\ m)\,?\,n : (f(\mathit{pred}_{\mathbb{N}}\ m)(\mathit{succ}_{\mathbb{N}}\ n))))) \\
\mathit{sub}_{\mathbb{N}} &= (Y(\lambda f.\lambda m.\lambda n.((\mathit{iszero}_{\mathbb{N}}\ n)\,?\,m : (f(\mathit{pred}_{\mathbb{N}}\ m)(\mathit{pred}_{\mathbb{N}}\ n)))) \\
\mathit{mult}_{\mathbb{N}} &= (Y(\lambda f.\lambda m.\lambda n.((\mathit{iszero}_{\mathbb{N}}\ m)\,?\,0_{\mathbb{N}} : (\mathit{add}_{\mathbb{N}}\ (f\ (\mathit{pred}_{\mathbb{N}}\ m)\ n)\ n))))
\end{aligned}$$

3.4 Simply-typed Lambda Calculus

3.4.1 Syntax of $\lambda^{\rightarrow}$

expressions	$e ::= n \mid v \mid \lambda v{:}\tau.e \mid e_1e_2 \mid \texttt{true} \mid \texttt{false} \mid e_1\ \mathit{aop}\ e_2 \mid e_1\ \mathit{bop}\ e_2$
	$\mid e_1\ \mathit{cmp}\ e_2 \mid (e_1, e_2) \mid \pi_1 e \mid \pi_2 e \mid () \mid \texttt{in}_1^{\tau_1+\tau_2} e \mid \texttt{in}_2^{\tau_1+\tau_2} e$
	$\mid (\texttt{case}\ e\ \texttt{of}\ \texttt{in}_1(v_1) \to e_1\ \texttt{|}\ \texttt{in}_2(v_2) \to e_2)$
types	$\tau ::= \mathit{int} \mid \mathit{bool} \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2 \mid \mathit{unit} \mid \tau_1 + \tau_2 \mid \mathit{void}$
arithmetic ops	$\mathit{aop} ::= \texttt{+} \mid \texttt{-} \mid \texttt{*}$
boolean ops	$\mathit{bop} ::= \wedge \mid \vee$
comparisons	$\mathit{cmp} ::= \leq \mid \geq \mid < \mid > \mid =$

3.4.2 Static Semantics of $\lambda^{\rightarrow}$

$$\Gamma \vdash n : \mathit{int} \tag{9}$$

$$\Gamma \vdash v : \Gamma(v) \tag{10}$$

$$\frac{\Gamma[v \mapsto \tau_1] \vdash e : \tau_2}{\Gamma \vdash (\lambda v\!:\!\tau_1\,.\,e) : \tau_1 \to \tau_2} \tag{11}$$

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} \tag{12}$$

$$\Gamma \vdash \texttt{true} : \mathit{bool} \tag{13}$$

$$\Gamma \vdash \texttt{false} : \mathit{bool} \tag{14}$$

$$\frac{\Gamma \vdash e_1 : \mathit{int} \qquad \Gamma \vdash e_2 : \mathit{int}}{\Gamma \vdash e_1 \ \mathit{aop}\ e_2 : \mathit{int}} \tag{15}$$

$$\frac{\Gamma \vdash e_1 : \mathit{bool} \qquad \Gamma \vdash e_2 : \mathit{bool}}{\Gamma \vdash e_1 \ \mathit{bop}\ e_2 : \mathit{bool}} \tag{16}$$

$$\frac{\Gamma \vdash e_1 : \mathit{int} \qquad \Gamma \vdash e_2 : \mathit{int}}{\Gamma \vdash e_1 \ \mathit{cmp}\ e_2 : \mathit{bool}} \tag{17}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} \tag{18}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \qquad i \in \{1,2\}}{\Gamma \vdash \pi_i e : \tau_i} \tag{19}$$

$$\Gamma \vdash () : \mathit{unit} \tag{20}$$

$$\frac{\Gamma \vdash e : \tau_i \qquad i \in \{1,2\}}{\Gamma \vdash \texttt{in}_i^{\tau_1+\tau_2} e : \tau_1 + \tau_2} \tag{21}$$

$$\frac{\Gamma \vdash e : \tau_1 + \tau_2 \qquad \Gamma[v_1 \mapsto \tau_1] \vdash e_1 : \tau \qquad \Gamma[v_2 \mapsto \tau_2] \vdash e_2 : \tau}{\Gamma \vdash (\texttt{case}\ e\ \texttt{of}\ \texttt{in}_1(v_1) \to e_1\ \texttt{|}\ \texttt{in}_2(v_2) \to e_2) : \tau} \tag{22}$$

3.5 System F

expressions	$e ::= \cdots \mid \Lambda\alpha.e \mid e[\tau]$
types	$\tau ::= \cdots \mid \alpha \mid \forall\alpha.\tau$

$$(\Lambda\alpha.e)[\tau] \to_1 e[\tau/\alpha]$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau} \tag{23}$$

$$\frac{\Gamma \vdash e : \forall\alpha.\tau'}{\Gamma \vdash e[\tau] : \tau'[\tau/\alpha]} \tag{24}$$