CSE 130 Final Solution, Winter 2019

Nadia Polikarpova

March 22, 2019

Q1: Lambda Calculus [25 pts]

```
let F1 = \n   -> n NOT TRUE

let F2 = \n   -> n (MUL TWO) ONE

let F3 = \x y -> ISZ (SUB y x) x y

let H1 = \f p -> PAIR (NOT (FST p)) (ITE (FST p) (f (SND p)) (SND p))
let F4 = \n   -> SND (n (H1 INC) (PAIR FALSE ZERO))

let H2 = \p   -> PAIR (INC (FST p)) (ADD (FST p) (SND p))
let F5 = \n   -> SND (n H2 (PAIR ONE ZERO))
```

```
                         F1    F2    F3    F4    F5

(A) max of x and y       [ ]   [ ]   [X]   [ ]   [ ]   (A)
(B) x < y                [ ]   [ ]   [ ]   [ ]   [ ]   (B)
(C) x > y                [ ]   [ ]   [ ]   [ ]   [ ]   (C)
(D) n squared            [ ]   [ ]   [ ]   [ ]   [ ]   (D)
(E) 2 to the power of n  [ ]   [X]   [ ]   [ ]   [ ]   (E)
(F) n divided by 2       [ ]   [ ]   [ ]   [X]   [ ]   (F)
(G) is n even?           [X]   [ ]   [ ]   [ ]   [ ]   (G)
(H) constant false       [ ]   [ ]   [ ]   [ ]   [ ]   (H)
(I) n-th fibonacci       [ ]   [ ]   [ ]   [ ]   [ ]   (I)
(J) sum from 0 to n      [ ]   [ ]   [ ]   [ ]   [X]   (J)
```

Q2: Haskell: Files and Directories [35 pts]

2.1 Tail-Recursive Size [15 pts]

```
size :: Entry -> Int
size e = loop 0 [e]
  where
    loop :: Int -> [Entry] -> Int
    loop acc []                 = acc
    loop acc (File _ s : es) = loop (acc + s) es
    loop acc (Dir _ cs : es) = loop acc (cs ++ es)
```

2.2 Remove [15 pts]

With pre-filtering and HO functions:

```
remove :: (Entry -> Bool) -> Entry -> Entry
remove _ f@(File _ _) = f
remove p (Dir name es) = Dir (map (remove p) (filter (not . p) es))
```

With post-filtering and HO functions:

```
remove :: (Entry -> Bool) -> Entry -> Entry
remove _ f@(File _ _) = f
remove p (Dir name es) = Dir (filter (not . p) (map (remove p) es))
```

With pre-filtering, no HO functions:

```
remove :: (Entry -> Bool) -> Entry -> Entry
remove _ f@(File _ _)      = f
remove _ (Dir name [])     = Dir name []
remove p (Dir name (e:es)) = Dir name (heads ++ rest)
  where
    heads = if p e then [] else [remove p e]
    (Dir _ rest) = remove p (Dir name es)
```

2.3 Clean up [5 pts]

```
cleanup :: Entry -> Entry
cleanup = remove isEmpty
  where
    isEmpty (Dir _ []) = True
    isEmpty _          = False
```

Q3: Semantics and Type Systems [20 pts]

3.1 Reduction 1 [5 points]

(A) `5                       =>  5                 [ ]`

(B) `(\x -> x) (1 + 2)       =>  (\x -> x) 3       [X]`

(C) `(\x -> x) (1 + 2)       =>  1 + 2             [ ]`

(D) `(\x -> x) (1 + 2)       =>  3                 [ ]`

(E) `(1 + 2) + (\x -> x)     =>  3 + (\x -> x)     [X]`

3.2 Reduction 2 [5 points]

```
(\x y -> (x + y) + (1 + 2)) (3 + 4) 5  =>  ???
```

(A) `Add-L` [ ]

(B) `Add-R` [ ]

(C) `Add` [X]

(D) `App-L` [X]

(E) `App-R` [X]

3.3 Typing 1 [5 points]

(A)	`[] |- \x -> x :: Int -> Int`	[X]
(B)	`[] |- \x -> x :: a -> a`	[X]
(C)	`[] |- \x -> x :: forall a . a -> a`	[X]
(D)	`[] |- x :: Int`	[ ]
(E)	`[x: a] |- x :: forall a . a`	[ ]

3.4 Typing 2 [5 points]

```
[] |- \x y -> x y :: forall a . forall b . (a -> b) -> a -> b
```

(A)	`T-Var`	[X]
(B)	`T-Abs`	[X]
(C)	`T-App`	[X]
(D)	`T-Inst`	[ ]
(E)	`T-Gen`	[X]

Q4: Prolog: Regular expressions [30 pts]

4.1 One of [10 points]

```
match(oneOf([C|_]), [C]).
match(oneOf([_|T]), [C]) :- match(oneOf(T), [C]).
```

4.2 Sequential Composition [10 points]

```
match(seq(R1, R2), S) :-
  append(S1, S2, S),
  match(R1, S1),
  match(R2, S2).
```

4.3 Kleene Star [10 points]

```
match(star(_), []).
match(star(R), S) :-
  append(S1, S2, S),
  match(R, S1),
  match(star(R), S2).
```