$Id: cmps112-2017q4-midterm.mm,v 1.71 2017-10-27 15:54:00-07 - - $

page 1	page 2	page 3	page 4	Total / 42

Please print clearly :

Name :

CruzID : @ucsc.edu

No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Do your scratch work elsewhere and enter only your final answer into the spaces provided. Points will be deducted for messy answers. Unreadable answers will be presumed incorrect.

1. ***Ocaml.***

   (a) Define `sum` without using any higher-order functions. **[2✔]**

```
sum : int list -> int = <fun>
# sum [1;2;3;4;5];;
- : int = 15
```

   (b) Define `fold_left`. **[2✔]**

```
val fold_left : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>
```

   (c) Define `sumf` which uses a β-reduced form of calling `fold_left`. **[1✔]**

```
let sumf = fold_left ____________________________________________;;
val sumf : int list -> int = <fun>
```

2. ***Scheme.*** Write a function to reverse a list. **[3✔]**

```
> (reverse '(1 2 3 4 5))
(5 4 3 2 1)
> (reverse '())
()
```

3. ***Scheme.*** Define `map`. **[2✔]**

```
> (map (lambda (x) (+ 5 x)) '(1 2 3 4))
(6 7 8 9)
> (map (lambda (x) (cons 5 x)) '(1 2 3 4))
((5 . 1) (5 . 2) (5 . 3) (5 . 4))
```

4. ***Ocaml.*** Given the definition of `fac`, fill in the type signatures of each of the entries in the table. **[2✔]**

```
let fac n =
    let rec fac′ n′ a′ =
        if n′ <= 1
        then a′
        else fac′ (n′ - 1) (n′ * a′)
    in fac′ n 1
;;
```

`fac`	
`n`	
`fac′`	
`n′`	
`a′`	
`<=`	
`1`	
`-`	
`*`	

5. ***Scheme.*** Using the same definitions as for Ocaml on the previous page:

   (a) Define `sum` without using any higher-order functions. **[2✔]**

   (b) Define `fold_left`. **[2✔]**

   (c) Define `sumf` which uses `fold_left`. **[1✔]**

6. ***Ocaml.***

   (a) Without using a higher-order function, define `evenlen` which returns `true` of the length of the list is even and `false` otherwise **[2✔]**

```
val evenlen : 'a list -> bool = <fun>
```

   (b) Define `evenlen` which uses `fold_left` with the same result. Use a β-reduced version. **[1✔]**

```
let evenlen = List.fold_left ____________________________________________;;
val evenlen : '_a list -> bool = <fun>
```

7. Name the two general types of polymorphism, and for each of them, name the specific kinds that represents each of them. **[2✔]**

general	specific

8. ***Ocaml.*** Write a function to reverse a list. **[2✔]**

9. ***Java.*** Write a function to reverse a list. Do not allocate or free any nodes. Do not use auxiliary functions. **[2✔]**

```
class node {
  int value;
  node link;
}
```

```
node reverse (node head) {

}
```

10. ***Ocaml.*** The Collatz conjectures states that for any positive integer n, if it is repeatedly replaced by $n/2$ when even and $3n+1$ when odd, it eventually converges on the integer 1. Write a function that uses a tail-recursive inner function to return a list of all integers starting from the argument and ending with 1. The inner function produces the list in the reverse order, then the outer function reverses the list. Use `List.rev` from the library to reverse the list. **[4✔]**

```
# collatz 4;;
- : int list = [4; 2; 1]
# collatz 10;;
- : int list = [10; 5; 16; 8; 4; 2; 1]
# collatz 20;;
- : int list = [20; 10; 5; 16; 8; 4; 2; 1]
# collatz 16;;
- : int list = [16; 8; 4; 2; 1]
```

Multiple choice. To the ***left*** of each question, write the letter that indicates your answer. Write ***Z*** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[12✔]**

number of correct answers		× 1 =	$= a$
number of wrong answers		× ½ =	$= b$
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	12		$= c$

1. Mathematical system defined by Alonzo Church which was later used by John McCarthy in the design of Lisp.
   (A) α-calculus
   (B) β-calculus
   (C) λ-calculus
   (D) η-calculus

2. The type system in Scheme is :
   (A) strong and dynamic
   (B) strong and static
   (C) weak and dynamic
   (D) weak and static

3. The type system in Ocaml is :
   (A) strong and dynamic
   (B) strong and static
   (C) weak and dynamic
   (D) weak and static

4. Backus-Naur form (BNF) was first used in the specification of which language ?
   (A) ALGOL 60
   (B) BASIC
   (C) COBOL
   (D) FORTRAN

5. What is the running time of :
```
let rec fib n =
    if n < 2 then n
    else fib (n - 1) + fib (n - 2);;
```
   (A) $O(n)$
   (B) $O(\log_2 n)$
   (C) $O(n^2)$
   (D) $O(2^n)$

6. How much stack space is used by `fib` ?
   (A) $O(n)$
   (B) $O(\log_2 n)$
   (C) $O(n^2)$
   (D) $O(2^n)$

7. What is 10 ?
   (A) `(apply + '(1 2 3 4))`
   (B) `(cons + '(1 2 3 4))`
   (C) `(filter + '(1 2 3 4))`
   (D) `(foldl + '(1 2 3 4))`

8. "Go To Statement Considered Harmful"
   (A) John Backus
   (B) Edsger Dijkstra
   (C) Grace Hopper
   (D) Donald Knuth

9. Assuming only pure Java code with no sneaky tricks written in C, If M = memory leaks, D = dangling references, and U = unsafe type conversions or casting, which of the following are possible in Java ?
   (A) all of them.
   (B) none of them.
   (C) only D, but neither M nor U.
   (D) only M, but neither D nor U.

10. Type of `(+)` ?
   (A) `int * int * int`
   (B) `int * int -> int`
   (C) `int -> int * int`
   (D) `int -> int -> int`

11. What is `(3 4)` ?
   (A) `(caar '(1 2 3 4))`
   (B) `(cadr '(1 2 3 4))`
   (C) `(cdar '(1 2 3 4))`
   (D) `(cddr '(1 2 3 4))`

12. In the expression $(\lambda x.\,(+x)\,y)$
   (A) x is bound and y is bound.
   (B) x is bound and y is free.
   (C) x is free and y is bound.
   (D) x is free and y is free.