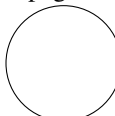
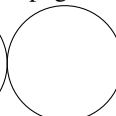
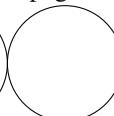
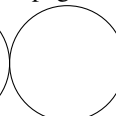



\$Id: cmps112-2017q4-midterm.mm,v 1.71 2017-10-27 15:54:00-07 - - \$

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  $\beta$ -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
;;
```

<b>fac</b>	
<b>n</b>	
<b>fac'</b>	
<b>n'</b>	
<b>a'</b>	
<b>&lt;=</b>	
<b>1</b>	
<b>-</b>	
<b>*</b>	

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** if 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  $\beta$ -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		$\times 1 =$	$= a$
number of wrong answers		$\times \frac{1}{2} =$	$= b$
number of missing answers		$\times 0 =$	0
column total $c = \max(a - b, 0)$	12		$= c$

- Mathematical system defined by Alonzo Church which was later used by John McCarthy in the design of Lisp.
  - $\alpha$ -calculus
  - $\beta$ -calculus
  - $\lambda$ -calculus
  - $\eta$ -calculus
- The type system in Scheme is :
  - strong and dynamic
  - strong and static
  - weak and dynamic
  - weak and static
- The type system in Ocaml is :
  - strong and dynamic
  - strong and static
  - weak and dynamic
  - weak and static
- Backus-Naur form (BNF) was first used in the specification of which language ?
  - ALGOL 60
  - BASIC
  - COBOL
  - FORTRAN
- What is the running time of :
 

```
let rec fib n =
  if n < 2 then n
  else fib (n - 1) + fib (n - 2);;
```

  - $O(n)$
  - $O(\log_2 n)$
  - $O(n^2)$
  - $O(2^n)$
- How much stack space is used by **fib** ?
  - $O(n)$
  - $O(\log_2 n)$
  - $O(n^2)$
  - $O(2^n)$

- What is 10 ?
  - (**apply** + ' (1 2 3 4))
  - (**cons** + ' (1 2 3 4))
  - (**filter** + ' (1 2 3 4))
  - (**foldl** + ' (1 2 3 4))
- "Go To Statement Considered Harmful"
  - John Backus
  - Edsger Dijkstra
  - Grace Hopper
  - Donald Knuth
- 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 ?
  - all of them.
  - none of them.
  - only D, but neither M nor U.
  - only M, but neither D nor U.
- Type of (+) ?
  - int \* int \* int
  - int \* int -> int
  - int -> int \* int
  - int -> int -> int
- What is (3 4) ?
  - (**caar** ' (1 2 3 4))
  - (**cadr** ' (1 2 3 4))
  - (**cdar** ' (1 2 3 4))
  - (**cddr** ' (1 2 3 4))
- In the expression  $(\lambda x. (+x)y)$ 
  - $x$  is bound and  $y$  is bound.
  - $x$  is bound and  $y$  is free.
  - $x$  is free and  $y$  is bound.
  - $x$  is free and  $y$  is free.

