`$Id: cmps112-2009q1-exam3.mm,v 9.34 2009-03-13 23:05:44-07 - - $`

page 1	page 2	page 3	page 4	page 5	Total / 52

Please PRINT using keyboard letters :

Name :	
Login :	`@ucsc.edu`

No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Do your scratch work elsewhere and enter only your final answer into the spaces provided.

1. Give an example of how a function call in normal order might return a result, but the same function call in applicative order might crash or go into an infinite loop. **[2✔]**

2. ***C++ :*** Write code to print out the elements of a vector, one item per line. Assume that **`operator<<`** is defined on type **`foo`**. Use a **`const_iterator`**. **[2✔]**
   ```
   vector<foo> vec;
   ```

3. ***Perl :*** Write a function that will accept a function and an array and return all elements of the array in the same order, for which the function applied to an element of the array returns true. **[2✔]**

4. ***Prolog :*** Draw a picture of the undirected graph represented by the facts given below. Write a function **`adjacent`** which will succeed if two nodes in the graph are adjacent to each other. **[2✔]**
   ```
   edge(a,b).
   edge(b,c).
   edge(c,d).
   edge(d,a).
   edge(a,c).
   ```

5. ***Prolog :*** Write **`drop/3`** such that it returns all elements of the second argument, starting with the one that matches the first argument, or none, if none match. The third argument is the result. **[2✔]**
   ```
   | ?- drop(3,[5,4,3,2,1],X).
   X = [3,2,1]
   | ?- drop(9,[1,2,3],X).
   X = []
   | ?- drop(1,[1,2,3,4],X).
   X = [1,2,3,4]
   ```

6. ***Ocaml :*** Define the function **`merge`** which takes a predicate as an argument and a pair of sorted lists. The result is a single list in sorted order. **[3✔]**

```
# merge;;
- : ('a -> 'a -> bool) -> 'a list -> 'a list -> 'a list
# merge (<) [1;3;5] [2;4;8;9];;
- : int list = [1; 2; 4; 4; 8; 9; 9]
```

7. ***Ocaml :*** Define the function **`sum`** in terms of **`List.fold_left`** in a curried manner to sum a list of integers. Do not write a recursive function. **[1✔]**

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

8. ***Ocaml :*** Define the function **`zip`** whose arguments are a curried pair of lists and whose result is a list of tuples. Raise **`Invalid_argument`** if the lengths are different. **[2✔]**

```
# zip;;
- : 'a list -> 'b list -> ('a * 'b) list
# zip [1;2;3] ["foo";"bar";"baz"];;
- : (int * string) list = [(1, "foo"); (2, "bar"); (3, "baz")]
# zip [1;2;3] [];;
Exception: Invalid_argument "length mismatch".
```

9. ***Ocaml :*** Define the function **`map`**, which a unary function to a list and returns the list of results. Use recursion. Do not use a higher-order function. **[2✔]**

```
# map;;
- : ('a -> 'b) -> 'a list -> 'b list
# map ((+)2) [1;2;3;4];;
- : int list = [3; 4; 5; 6]
```

10. ***Ocaml :*** Write the function **`reverse`** which reverses list. Do not use any higher-order functions. Your function must be tail-recursive or use a local tail-recursive helper. **[2✔]**

```
# reverse;;
- : 'a list -> 'a list
# reverse [1;2;3];;
- : int list = [3; 2; 1]
```

11. ***Ocaml:*** Write a function **iota** which has an integer argument *n* and returns a list of numbers from 1 to *n* inclusive. The empty list is returned for non-positive numbers. Use a local helper function to make your solution tail-recursive. **[2✔]**

```
# iota;;
- : int -> int list = <fun>
# iota 5;;
- : int list = [1; 2; 3; 4; 5]
# iota (-5);;
- : int list = []
```

12. Give an example of one function nested inside another, where the inner function refers to a local variable of the outer function in such a way that the program crashes due to a dangling pointer. **[2✔]**

13. Give an example of how memory leak might occur in Java. **[2✔]**

14. ***Prolog:*** Define the function `product`, which produces the product of all the numbers in a list. **[2✔]**

```
| ?- product(N,[1,2,3,4,5]).
N = 120
| ?- product(N,[]).
N = 1
```

15. ***Scheme:*** Write the function `elim`, which takes a symbol and a list and returns a list consisting of elements of the list starting with the first one that is `eqv?` to its first argument; or the empty list, if none. **[2✔]**

```
> (elim 3 '(5 4 3 2 1))
(3 2 1)
> (elim 9 '(1 2 3))
()
> (elim 1 '(1 2 3 4))
(1 2 3 4)
```

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. **[11✔]**

number of correct answers		× 1 =	= *a*
number of wrong answers		× ½ =	= *b*
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	11		= *c*

1. Haskell is a purely functional language which maintains state using a :
   (A) closure
   (B) daemon
   (C) monad
   (D) thunk

2. Partial parameterization of a currried function keeps arguments in a :
   (A) closure
   (B) daemon
   (C) monad
   (D) thunk

3. In C++, a static variable is bound to a virtual address :
   (A) at compile (`CC`) time.
   (B) at link (`ld`) time.
   (C) at `exec()` time.
   (D) when `main()` is called.

4. In Java, a static variable is allocated :
   (A) at translation time.
   (B) when the class files are put in a jar.
   (C) when the class is loaded.
   (D) when an object is created with `new`.

5. From what segment does a call to `new` in C++ allocate memory ?
   (A) data
   (B) heap
   (C) stack
   (D) text

6. What is the type of `f` ?
   `let f x y = x + y;;`
   (A) `int * int * int`
   (B) `int * int -> int`
   (C) `int -> int * int`
   (D) `int -> int -> int`

7. When arguments to functions are evaluated before the function is called, this is ____ order.
   (A) applicative
   (B) efficient
   (C) normal
   (D) short circuit

8. A garbage collector which is most friendly to the page tables by compacting heap objects into as few pages as possible :
   (A) concurrent reclamation of live objects
   (B) copying collector with semispaces
   (C) mark and sweep collector
   (D) reference counting

9. Of the ones listed here, the attribute most associated with functional programming is :
   (A) dynamic dispatch
   (B) referential transparency
   (C) static type checking
   (D) unification

10. If `a` is a list, which expression produces the same list ?
    (A) `(car (cdr (cons a)))`
    (B) `(car (cons a (cdr a)))`
    (C) `(cons (car a) (cdr a)))`
    (D) `(cons (cdr a) (car a)))`

11. The Java class that permits a process to have multiple things done concurrently is :
    (A) `Daemon`
    (B) `Runnable`
    (C) `Task`
    (D) `Thread`

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. **[11✔]**

number of correct answers		× 1 =	= *a*
number of wrong answers		× ½ =	= *b*
number of missing answers		× 0 =	0
column total $c = \max(a - b, 0)$	11		= *c*

1. What is the type of `swap` ?
   `let swap f x y = f y x;;`
   (A) `('a -> 'b -> 'a) -> 'a -> 'b list -> 'a`
   (B) `('a -> 'b -> 'b) -> 'a list -> 'b -> 'b`
   (C) `('a -> 'b -> 'c) -> 'b -> 'a -> 'c`
   (D) `('a -> 'b) -> ('c -> 'a) -> 'c -> 'b`

2. What is `6` ?
   (A) `((lambda (x)(+ x 3 ))3)`
   (B) `(cdar '(7 6 5 4 3))`
   (C) `(if (2 < 3) 4 6)`
   (D) `(map * '(1 2 3))`

3. The function `List.fold_left` uses up how much stack space on a list of length n ?
   (A) $O(1)$
   (B) $O(2^n)$
   (C) $O(\log_2 n)$
   (D) $O(n)$

4. An object-oriented language such as C++ does dynamic dispatching of method calls by means of a :
   (A) virtual function table
   (B) template declaration
   (C) heap-allocated closure
   (D) friend function

5. What is the stack efficiency of this function ?
```
let rec f n =
    if n <= 1 then n
    else f (n - 1) + f (n - 2);;
```
   (A) $O(1)$
   (B) $O(n)$
   (C) $O(n^2)$
   (D) $O(2^n)$

6. If `guess` is a predicate that searches a database to return one of its elements, and `verify` checks to see if the thing found is good, then we may define the predicate `find`, which returns a valid entry from the database as :
   (A) `find(X) :- guess(X), verify(X).`
   (B) `find(X) :- guess(X).`
   `find(X) :- verify(X).`
   (C) `find(X) :- verify(X), guess(X).`
   (D) `guess(X) :- find(X), verify(X).`

7. What is the stack efficiency of this function ?
```
let rec f n =
    if n <= 1 then n
    else f (n - 1) + f (n - 2);;
```
   (A) $O(1)$
   (B) $O(n)$
   (C) $O(n^2)$
   (D) $O(2^n)$

8. Which of the following data structures violates the spirit of functional programming ?
   (A) array
   (B) list
   (C) stack
   (D) tree

9. Passing a parameter by ____ means that it is passed in unevaluated and then evaluated only if needed.
   (A) name
   (B) reference
   (C) value
   (D) value-result

10. Which of these is not part of the local stack frame in ANSI C ?
    (A) access (static) link
    (B) control (dynamic) link
    (C) register save area
    (D) return address

11. The most recently released version of the Scheme language (not in draft format) is :
    (A) R5RS
    (B) R6RS
    (C) R7RS
    (D) R8RS

Total : 52 ✔