$Id: cmps112-2012q1-exam2.mm,v 1.75 2012-02-24 18:40:26-08 - - $

page 1	page 2	page 3	Total / 31

Please print clearly :

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. ***Haskell :*** Define the function `map` whose first argument is a unary function, and second argument list. The result is a list of the function applied to each element of the argument list. The definition is a one-liner using a list comprehension. **[1✔]**

```
Prelude> map (+2) [1,3,5,6]
[3,5,7,8]
Prelude> map (3-) [1,2,5,6]
[2,1,-2,-3]
```

2. ***Ocaml :*** Define the function `mapf` whose first argument is a unary function, and second argument list. The result is a list of the function applied to each element of the argument list. Write a one-liner using `fold_right`, and not a recursive function. **[2✔]** 1 point.

```
# fold_right;;
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
# mapf (fun x -> x + 2) [1;3;5;6];;
- : int list = [3; 5; 7; 8]
# mapf ((-)3) [1;2;5;6];;
- : int list = [2; 1; -2; -3]
```

3. ***Ocaml :*** Define the function `mapr` having exactly the functionality in the previous questions, except do not use any higher order functions. Use recursion. **[2✔]**

4. ***Ocaml :*** Define the functions `car` and `cdr` in Ocaml so that they work in the expected way. Use pattern matching. Neither function may call any other. Use `failwith` for a `[]` argument. **[2✔]**

```
# car;;
- : 'a list -> 'a = <fun>
# cdr;;
- : 'a list -> 'a list = <fun>
# car [1;2;3;4];;
- : int = 1
# cdr [1;2;3;4];;
- : int list = [2; 3; 4]
```

5. ***Perl :*** Write a program in Perl which reads words and keeps track of their lengths. At end of file, print out a table of two columns, with each line consisting of the length of a word and the number of words of that length. Use `<>` to read lines. A word is any sequence of characters that matches the regex `m/\w+/`. The example output shows that for this input, there is 1 word of length 1, 4 words of length 3, and 3 words of length 5. **[3✔]**

example input	example output
`foo bar baz x`	`1    1`
`hello world`	`3    4`
`qux quuux`	`5    3`

6. ***Ocaml:*** Define a function `zipwith` whose first argument is a curried function of two arguments, second argument is a single value, and whose third and fourth arguments are lists. It merges the lists into a single list by applying the function in a pairwise manner, using the single value if one list runs out first. **[3✔]**

```
# zipwith;;
- : ('a -> 'a -> 'b) -> 'a -> 'a list -> 'a list -> 'b list = <fun>
# (+);;
- : int -> int -> int = <fun>
# zipwith (+) 0;;
- : int list -> int list -> int list = <fun>
# zipwith (+) 0 [1; 2; 3] [4; 5; 6; 7; 8];;
- : int list = [5; 7; 9; 7; 8]
# zipwith (^) "**" ["foo"] ["bar"; "baz"];;
- : string list = ["foobar"; "**baz"]
# zipwith (^) "**" ["foo"; "bar"] ["baz"];;
- : string list = ["foobaz"; "bar**"]
```

7. ***Ocaml:*** Define a function `max` such that given a function of two arguments giving a `bool` and a list, it returns `Some` maximum element of the list, and `None` otherwise. **[3✔]**

```
# type 'a opt = None | Some of 'a;;
type 'a opt = None | Some of 'a
# max;;
- : ('a -> 'a -> bool) -> 'a list -> 'a opt = <fun>
# (>);;
- : 'a -> 'a -> bool = <fun>
# max (>) [3; 1; 4; 1; 5; 9];;
- : int opt = Some 9
# max (<) [3; 1; 4; 1; 5; 9];;
- : int opt = Some 1
# max (>) [];;
- : 'a opt = None
```

8. ***Scheme:*** Define the function `zipwith` in Scheme. Note that for both of these programs, if the first list is shorter, the value is used as the first argument, and if the second list is shorter, the value is the second argument to the function. **[4✔]**

```
> (zipwith - 5 '(9 8 7) '(5 4 3 2 1))
(4 4 4 3 4)
> (zipwith string-append "**"
  '("hello" "foo" "qux" "goto") '(" world" " bar"))
("hello world" "foo bar" "qux**" "goto**")
```

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. The type signature of `(/.)` is :
   (A) `float * float * float`
   (B) `float * float -> float`
   (C) `float -> float * float`
   (D) `float -> float -> float`

2. Assuming a competent implemention in Ocaml, which function takes up the most stack space ?
   (A) `List.find`
   (B) `List.fold_left`
   (C) `List.fold_right`
   (D) `List.length`

3. In Smalltalk, the expression `3-4/5-6` is equivalent to :
   (A) `((3-4)/5)-6`
   (B) `(3-(4/5))-6`
   (C) `(3-4)/(5-6)`
   (D) `3-((4/5)-6)`

4. If you have a function `not` (`bool->bool`) and a function `even` (`int->bool`), which higher-order function would be useful in combining them to make a function called `odd` ?
   (A) `compose`
   (B) `filter`
   (C) `fold_left`
   (D) `map`

5. A closure is :
   (A) A special field of a structure or class used to point at a base class when implementing shared multiple inheritance.
   (B) A special type declaration in Ocaml used to distinguish sum types from product types.
   (C) A structure on the heap, used to hold variables of an outer function when referenced by an inner function.
   (D) A table used to dynamically dispatch virtual functions in an object-oriented environment.

6. After the following open statement, what can be used to read one line from the file ?
   ```
   open my $file, "<$filename"
   ```
   (A) `$line = '$file';`
   (B) `$line = <$file>;`
   (C) `$line = "$file";`
   (D) ``$line = `$file`;``

7. In a lazy language, unevaluated arguments are passed into functions by means of a :
   (A) closure
   (B) curry
   (C) thunk
   (D) tuple

8. The type of `[1; 2; 3; 4]` is :
   (A) `'a list`
   (B) `int list`
   (C) `list<Object>`
   (D) `list<int>`

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

10. The following prints the number 6 in which language ?
    ```
    stdout << 6 << Character nl.
    ```
    (A) Ocaml
    (B) Perl
    (C) Scheme
    (D) Smalltalk

11. The classic paper† "Go To Statement Considered Harmful", CACM, 1968, was written by :
    (A) John Backus
    (B) Edsger Dijkstra
    (C) Grace Hopper
    (D) Donald Knuth

† He also said :
- "FORTRAN, the infantile disorder, by now nearly 20 years old, is hopelessly inadequate for whatever computer application you have in mind today : it is now too clumsy, too risky, and too expensive to use."
- "PL/I, the fatal disease, belongs more to the problem set than to the solution set."
- "It is practically impossible to teach good programming to students that have had a prior exposure to BASIC : as potential programmers they are mentally mutilated beyond hope of regeneration."
- "The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence."
- "In the good old days physicists repeated each other's experiments, just to be sure. Today they stick to FORTRAN, so that they can share each other's programs, bugs included."