`$Id: cmps112-2017q2-final.mm,v 1.173 2017-06-01 17:28:20-07 - - $`

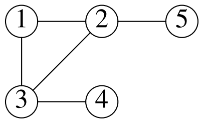
page 1	page 2	page 3	page 4	page 5	Total / 54

Please print clearly :

Name :	
Login :	@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. ***Prolog.*** Define some facts called **`edge`** that will represent the following undirected graph. **[1✔]**
   Define a predicate **`adjacent`** which is true if two nodes are adjacent to each other. **[1✔]**



2. ***Ocaml.*** Given the function at the left, fill in the types of each of the identifiers in the table at the right. **[2✔]**

```
let fib n =
    let rec fib′ m a b =
        if m = 0
        then a
        else fib′ (m - 1) b (a + b)
    in  if n < 0
        then failwith "fac n when n < 0"
        else fib′ n 0 1
;;
```

`fib`	
`fib′`	
`n`	
`m`	
`a`	
`b`	
`-`	
`+`	
`=`	
`<`	
`failwith`	`string -> 'a`

3. ***Ocaml.*** Define the functions **`car`** and **`cdr`**. Use **`failwith`** (see above) for an inappropriate argument. Do not use **`List.hd`** and **`List.tl`**. **[2✔]** State the types of these two functions. **[1✔]**

4. ***Prolog.*** Define **`sum`**. **[1✔]** Define **`length`**. **[1✔]**

```
| ?- sum([],X).            | ?- length([],X).
X = 0                      X = 0
| ?- sum([1,2,3,4],X).     | ?- length([1,2,3,4],X).
X = 10                     X = 4
```

5. ***Scheme.*** Draw a picture of the following list structure. Write a number in the car of the cell if it contains a number. Draw an downward arrow from the car of the cell if it is a pointer. Draw a horizontal arrow pointing right in the cdr of any cell in the list. Write the Greek letter φ to represent a null pointer in the cdr. All cells at the same depth should be on the same horizontal line of your diagram. **[1✔]**
   `(1 2 (3 4) ((5) 6) 7)`

6. For each language described here, fill in the name of the language. Choose from among the following languages : Algol 60, AWK, Bash, Basic, BCPL, C, C++, COBOL, Forth, FORTRAN, Haskell, Intercal, Java, Lisp, ML, OCaml, Pascal, Perl, PL/I, Prolog, Simula 67, Smalltalk. Grading : deduct ½ point for each wrong or missing answer, but do not score less than 0. **[2✓]**

	Bjarne Stroustrup's most noted contribution to language design.
	Business data processing language, designers included Grace Hopper.
	List processing language with **L**ots of **I**diotic **S**illy **P**arentheses.
	Numeric and scientific computation language developed at IBM (1957).
	Simulation language that influenced the design of C++.
	Small language for structured programming designed by Niklaus Wirth.
Intercal ‡	Parody language with the `come from`, `maybe`, and `forget` control structures.

‡ *Lasciate ogni speranza, voi ch'entrate.*

7. ***Smalltalk.*** Extend class `Array` with the message `max` which returns the maximum element in the array. Return `nil` if the array has no elements. **[4✓]**

```
st> a := #(3 1 4 1 5 9 2 6 5 3 5).
(3 1 4 1 5 9 2 6 5 3 5 )
st> a max.
9
st> #(333 9999 88 47) max.
9999
st> z := Array new.
()
st> z size.
0
st> z max.
nil
```

8. ***Scheme.*** Without using a higher order function, define `reverse` to reverse a list. Use tail recursion and a nested helper function. **[2✓]**

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

9. ***Ocaml.*** Define the functions `sum` and `length` by filling in the blanks. **[2✓]**

```
# List.fold_left;;
- : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>


# let sum = List.fold_left ________________________________________;;
val sum : int list -> int = <fun>
# sum [1;2;3;4;5];;
- : int = 15


# let length = List.fold_left ________________________________________;;
val length : '_a list -> int = <fun>
# length [1;2;3;4;5];;
- : int = 5
```

10. ***Ocaml.*** Define the function `twice` which applies a function to its argument twice. **[1✔]**

```
# twice;;
- : ('a -> 'a) -> 'a -> 'a = <fun>
# twice ((+)1) 3;;
- : int = 5
# twice (fun x -> x - 3) 5;;
- : int = -1
```

11. ***Prolog.*** Given facts like the ones presented at the left, define the rules `father` and `mother` where the first argument is the parent and the second argument is the child. For the facts called `parent`, the arguments are, in order `parent( Father, Mother, Child).` **[2✔]**

```
parents( henry_vii, elizabeth_of_york, henry_viii).
parents( henry_viii, catherine_of_aragon, mary_i).
parents( henry_viii, anne_boleyn, elizabeth_i).
parents( henry_viii, jane_seymour, edward_vi).
| ?- father( X, henry_viii).
X = henry_vii
| ?- father( henry_viii, X).
X = mary_i
X = elizabeth_i
X = edward_vi
| ?- mother( X, elizabeth_i).
X = anne_boleyn
| ?- mother( elizabeth_of_york, X).
X = henry_viii
```

12. Define a function `oddlen` which returns true if the length of the list is odd and false if it is even. Do not use any built-in library functions. Remember that 0 is an even number. Do not use any length function. Use tail recursion.

(a) ***Scheme.*** **[1✔]**

```
> (oddlen '(1 2 3))
#t
> (oddlen '(1 2 3 4))
#f
```

(b) ***Ocaml.*** **[1✔]**

```
# oddlen [1;2;3];;
- : bool = true
# oddlen [1;2;3;4];;
- : bool = false
```

(c) ***Prolog.*** **[1✔]**

```
| ?- oddlen([1,2,3]).
true
| ?- oddlen([1,2,3,4]).
no
```

13. Define the function `map`.

(a) ***Ocaml.*** **[2✔]**

```
# List.map;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
# List.map ((+)6) [1;2;3;4];;
- : int list = [7; 8; 9; 10]
```

(b) ***Scheme.*** **[2✔]**

```
> (map (lambda (x) (+ x 6)) '(1 2 3 4))
(7 8 9 10)
```

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. How much stack space is taken up by the following function?
```
let rec fib n
    if n <= 1 then n
    else fib (n - 1) + fib (n - 2)
```
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(2^n)$

2. How much stack space is taken up by the function `fib` on the first page of this exam?
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(2^n)$

3. How will `Ocaml` respond to the following statement?
   `(+);;`
   (A) `- : int * int * int = <fun>`
   (B) `- : int * int -> int = <fun>`
   (C) `- : int -> int * int = <fun>`
   (D) `- : int -> int -> int = <fun>`

4. Which of the following data structures is inconsistent with functional programming style?
   (A) list
   (B) stack
   (C) tree
   (D) vector

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. How much stack space is used by:
```
(define (fib n)
    (if (<= n 1) n
        (+ (fib (- n 1)) (fib (- n 2)))))
```
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(2^n)$

7. 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))`

8. What is `5` in Smalltalk?
   (A) `(1 + 4) value.`
   (B) `<1 + 4> value.`
   (C) `[1 + 4] value.`
   (D) `{1 + 4} value.`

9. In Smalltalk `2*3+4*5` is:
   (A) `((2*3)+4)*5`
   (B) `(2*3)+(4*5)`
   (C) `2*((3+4)*5)`
   (D) `2*(3+(4*5))`

10. In Prolog and Scheme, type checking is:
   (A) strong and dynamic
   (B) strong and static
   (C) weak and dynamic
   (D) weak and static

11. What is 24?
   (A) `(apply * '(1 2 3 4))`
   (B) `(map * '(1 2 3 4))`
   (C) `(cons * '(1 2 3 4))`
   (D) `(list * '(1 2 3 4))`

12. "Go To Statement Considered Harmful"
   (A) Corrado Böhm & Giuseppe Jacopini
   (B) Donald E. Knuth
   (C) Edsger W. Dijkstra
   (D) Niklaus Wirth

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.
APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.
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.

— EWD498: "How do we tell truths that might hurt?"
prof. dr. Edsger W. Dijkstra, 1975.
http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/

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. What does the pure functional language Haskell use to maintain state ?
   (A) closure
   (B) dæmon
   (C) monad
   (D) thunk

2. Which language permits extension, addition of new methods to a class, at runtime ?
   (A) C
   (B) C++
   (C) Java
   (D) Smalltalk

3. Perl. What regex will match one or more occurrences of the letter `a` ?
   (A) `[a]`
   (B) `a*`
   (C) `a+`
   (D) `a?`

4. Perl. What will substitute the value of the variable `$foo` into the string assigned to `$x` ?
   (A) `$x = '$foo\n';`
   (B) `$x = <$foo\n>;`
   (C) `$x = "$foo\n";`
   (D) `` $x = `$foo\n`; ``

5. What function is called immediately after `d()` if `d()` is true ?

```
for (a(); b(); c()){
   if (d()) continue;
   e();
   if (f()) break;
   g();
}
h();
```

   (A) `b()`
   (B) `c()`
   (C) `e()`
   (D) `h()`

6. Ocaml. `List.hd`
   (A) `'a -> 'a`
   (B) `'a -> 'a list`
   (C) `'a list -> 'a`
   (D) `'a list -> 'a list`

7. What kind of polymorphism is exhibited by :

```
class foo { ... }
class bar extends foo { ... }
```

   (A) ad hoc conversion
   (B) ad hoc overloading
   (C) universal inclusion
   (D) universal parametric

8. What kind of polymorphism is exhibited by :

```
void foo (int);
void foo (string);
```

   (A) ad hoc conversion
   (B) ad hoc overloading
   (C) universal inclusion
   (D) universal parametric

9. What kind of polymorphism is exhibited by C++ templates and Java generics ?
   (A) ad hoc conversion
   (B) ad hoc overloading
   (C) universal inclusion
   (D) universal parametric

10. What kind of polymorphism is exhibited by :

```
void foo (double);
foo (3);
```

   (A) ad hoc conversion
   (B) ad hoc overloading
   (C) universal inclusion
   (D) universal parametric

11. In Java or C++, which statement can cause control to pass from the current function to the calling function, or perhaps the caller of the caller, or perhaps even all the way back to the main function ?
   (A) `break`
   (B) `continue`
   (C) `return`
   (D) `throw`

12. The syntax of what language was the first to be defined by Backus-Naur Format (BNF) ?
   (A) Algol 60
   (B) Basic
   (C) Cobol
   (D) Fortran