$Id: cmps112-2015q4-exam3.mm,v 1.108 2015-12-01 20:00:12-08 - - $

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. Define `gcd` which uses Euclid's algorithm to find the greatest common divisor for two integers $x > 0$ and $y > 0$. The C version is given. Example : gcd(111, 259) = gcd(111, 148) = gcd(111, 37) = gcd(74, 37) = gcd(37, 37) = 37.

```
int gcd (int x, int y) {
   while (x != y) if (x > y) x -= y; else y -= x;
   return x;
}
```

(a) ***Scheme.*** Use tail recursion. **[2✔]**
Example call : `(define g (gcd 111 259))`.

(b) ***Ocaml.*** Use tail recursion and curried format. **[2✔]**
Example call : `let d = gcd 111 259;;`.

(c) ***Smalltalk.*** Extend class `Integer` with a keyword method `gcd:`. Use a loop. **[2✔]**
Example call : `g := 111 gcd: 259.`

(d) ***Perl.*** Use a loop or tail recursion. Properly prototype the function. **[2✔]**
Example call : `$g = gcd 111, 259;`

(e) ***Prolog.*** **[2✔]**
Example call : `gcd( 111, 259, G ).`

2. ***λ-calculus.*** Given the expression in the λ-calculus shown at the top of each box, show the derivation order to the number 25 for each of normal order and applicative order evaluation. **[1✔]**

normal order evaluation	applicative order evaluation
$(\lambda x\,.\ *\ x\ x)\ (+\,2\,3)\ =$	$(\lambda x\,.\ *\ x\ x)\ (+\,2\,3)\ =$

3. ***Scheme.*** Using `apply`, `map`, `max`, and `cons`, define the function `depth` for any argument. If it is `null?`, its depth is 1. Otherwise, if it is not a `pair?`, its depth is 0. The depth of anything else (a list) is one more than the maximum depth of the elements of the list. **[2✔]**

```
> (depth '(1 2 (3 4 (5 6)) 88))
3
> (depth '(a b c))
1
> (depth '())
1
> (depth 7)
0
```

4. ***Ocaml.*** Define `drop`, which returns its argument list without the first *n* elements. If *n* is larger than the length of the list, it returns a null list. If *n* is not positive, it just returns the list. Use a tail call. Do not compute the length of the list. **[2✔]**

```
# drop;;
- : int -> 'a list -> 'a list = <fun>
# drop 3 [1;2;3;4;5;6;7];;
- : int list = [4; 5; 6; 7]
# drop 10 [1;2;3;4];;
- : int list = []
# drop (-5) [1;2;3;4];;
- : int list = [1; 2; 3; 4]
# drop 5 [];;
- : 'a list = []
```

5. ***Smalltalk.*** Extend class `Array` with an instance method `find:` whose argument is a value which is searched for in the array. If the value is present in the array, return the index of the first position where it is. If not found, return `nil`. **[2✔]**

```
st> a := #(5 6 7 8 9).
(5 6 7 8 9 )
st> a find: 6
2
st> a find: 99
nil
```

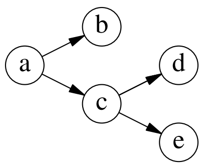
6. ***Perl.*** Write a program which prints out the file size, modification time, and filename for each file mentioned in `@ARGV`. Hints: The result of the `stat` function is an array where `$stat[7]` is the file size and `$stat[9]` is the modification time. Use the `strftime` format `"%b %e %H:%S"` to print out the time. Print a suitable error message if `@stat` has length 0. **[3✔]**

```
-bash-60$ ls.perl *.perl
      84 Nov 12 13:37 count.perl
     240 Nov 16 12:39 euclid.perl
     253 Nov 25 19:03 ls.perl
     110 Dec  5 17:53 range.perl
      91 Mar 14 21:31 wc.perl
```

7. Write the name of a programming language associated with each of the following people. Score 1/4 point for each correct answer, but not more than 2 points total. Choose answers from: AWK, BASIC, C, C++, COBOL, FORTRAN, Java, Lisp, Perl, Python, Scheme, λ-calculus. **[2✓]**

Alfred Aho	John Backus	Alonzo Church
James Gosling	Grace Hopper	John Kemeny
John McCarthy	Dennis Ritchie	Guy Steele
Bjarne Stroustrup	Larry Wall	Guido van Rossum

8. ***Prolog.*** Write facts in Prolog to describe the graph at left. Use the term `arrow` whose first argument is the tail of the arrow and whose second argument is the head of the arrow, i.e., `arrow(X,Y)` means that node `X` points directly at node `Y`. Write a rule `arrow2(X,Y)` which finds out if it is possible to get from `X` to `Y` by following exactly two `arrow`s. **[2✓]**



9. ***Scheme.*** Draw a picture of the following Scheme expression. For each `cons` cell, draw a rectangular box divided into to parts, and draw an arrow from each of the `car` and the `cdr` fields to the cell or object pointed to. **[2✓]**

```
((a b c) d (e) (f (g) h))
```

10. ***Perl.*** Write a program that reads files mentioned on the command line, and reads `STDIN` if none. Do not open files — use the `<>` operator. At the end of the last file, print each word followed by the number of times it appears. Print the words lexicographically. A word is any sequence of characters that matches `m/\w+/`. An example is given. **[2✓]**

example input	example output

`This is a test.`
`test is a This.`
`is this a test?`
`testing this.`
	`This 2`
	`a 3`
	`is 3`
	`test 3`
	`testing 1`
	`this 2`

11. ***Ocaml.*** Write a function `eval` which takes an `expr` as an argument and returns a `float` result. An `expr` is either a `Number` or an `Expr` with a `char` operator and two `expr`s. The only operators recognized are `'+'` and `'*'`. **[2✓]**

Definitions:

```
type expr = Number of float
          | Expr of char * expr * expr;;
let a = Expr ('+',
        Expr ('*', Number 6.0, Number 7.2),
        Expr ('*', Number 1.5, Number 2.7));;
```

Interaction:

```
# eval;;
- : expr -> float = <fun>
# eval a;;
- : float = 47.25
```

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. If `a` is a valid list, what is equal to `a` itself ?
   (A) `(car (cdr (cons a)))`
   (B) `(cons (car (cdr a)))`
   (C) `(cons (car a) (cdr a))`
   (D) `(cons (cdr a) (car a))`

2. What is the Perl equivalent to `strerror(errno)` ?
   (A) `"$!"`
   (B) `"$0"`
   (C) `"$?"`
   (D) `"$_"`

3. In Perl, how can `$p` be made to be a reference to an array containing some integers ?
   (A) `$p = (1, 2, 3, 4);`
   (B) `$p = <1, 2, 3, 4>;`
   (C) `$p = [1, 2, 3, 4];`
   (D) `$p = {1, 2, 3, 4};`

4. What is the Ocaml type signature for the definition: `let f x = x;;`
   (A) `val f : 'a -> 'a = <fun>`
   (B) `val f : 'a -> 'b -> 'b * 'a = <fun>`
   (C) `val f : 'a -> 'b -> 'b = <fun>`
   (D) `val f : int -> int = <fun>`

5. 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

6. An object-oriented language like C++ does dynamic dispatching of method calls using a :
   (A) friend function
   (B) heap-allocated closure
   (C) template declaration
   (D) virtual function table

7. The Perl pattern equivalent to `[a-zA-Z0-9_]` is :
   (A) `\d+`
   (B) `\s+`
   (C) `\t+`
   (D) `\w+`

8. If we have a function `not` (`bool -> bool`) and a function `even` (`int -> bool`), how might the function `odd` be defined ?
   (A) `let odd = compose not even`
   (B) `let odd = map not even`
   (C) `let odd = not even`
   (D) `let odd x = not even x`

9. 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.

10. In Perl, what command will put the names of files in the current directory in the variable `@files` ?
    (A) `@files = <ls>;`
    (B) ``@files = `ls`;``
    (C) `@files = glob "ls";`
    (D) `@files = system 'ls';`

11. What is the type of `car` in the following ?
    `let car s = match s with | x::xs -> x`
    (A) `val car : 'a -> 'a = <fun>`
    (B) `val car : 'a -> 'a list = <fun>`
    (C) `val car : 'a list -> 'a = <fun>`
    (D) `val car : 'a list -> 'a list = <fun>`

12. What is the type of `tail` in the following ?
    `let cdr s = match s with | x::xs -> xs`
    (A) `val cdr : 'a -> 'a = <fun>`
    (B) `val cdr : 'a -> 'a list = <fun>`
    (C) `val cdr : 'a list -> 'a = <fun>`
    (D) `val cdr : 'a list -> 'a list = <fun>`

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. The basic algorithm used in type inference is:
   (A) code replication
   (B) interpretation
   (C) overloading
   (D) unification

2. If `guess` finds something in a sequence of facts, and `verify` checks to see if it is a good one, then `find` can be defined in Prolog as:
   (A) `find(X) :- guess(X), verify(X).`
   (B) `find(X) :- guess(X).`
       `find(X) :- verify(X).`
   (C) `find(X) :- guess(X), !, verify(X).`
   (D) `guess(X), verify(X) := find(X).`

3. What is 6?
   (A) `(apply + '(1 2 3))`
   (B) `(cons + '(1 2 3))`
   (C) `(list + '(1 2 3))`
   (D) `(map + '(1 2 3))`

4. Which will unexpectedly start a comment?
   (A) `let f = (*);;`
   (B) `let f = (+);;`
   (C) `let f = (-);;`
   (D) `let f = (/);;`

5. What Perl statement will open a pipe to a subprocess and allow writing to its standard input?
   (A) `open my $file, "$name|"`
   (B) `open my $file, "<$name"`
   (C) `open my $file, ">$name"`
   (D) `open my $file, "|$name"`

6. Which language uses lazy evaluation by default?
   (A) Haskell
   (B) Lisp
   (C) Ocaml
   (D) Scheme

7. In Ocaml, what is 7?
   (A) `(+) (3, 4);;`
   (B) `(+) 3 4;;`
   (C) `(+) 3, 4;;`
   (D) `3 (+) 4;;`

8. 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()`

9. The following interaction indicates what kind of polymorphism?

```
# List.length;;
- : 'a list -> int = <fun>
```

   (A) conversion
   (B) inclusion
   (C) overloading
   (D) parametric

10. If `$key` is a key, what is the value associated with it in a hash? \$hash{\$key} %hash{\$key} &hash{\$key} @hash{\$key}

11. What kind of function is
    `let f x y z = x + y + z;;`
    (A) curried
    (B) thunked
    (C) tupled
    (D) unified

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