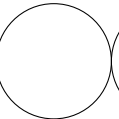
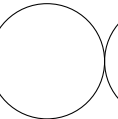
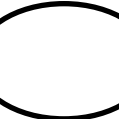


\$Id: cmpls112-2018q4-final.mm,v 1.117 2018-11-30 14:20:58-08 - - \$

page 1	page 2	page 3	page 4	page 5	Total / 54	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. Define **gcd** which uses Euclid's algorithm to find the greatest common divisor for two integers $x > 0$ and $y > 0$.

Mathematically: $\text{gcd}(111, 259) = \text{gcd}(111, 148) = \text{gcd}(111, 37) = \text{gcd}(74, 37) = \text{gcd}(37, 37) = 37$.

Translate the C version and be consistent with the examples.

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

- (a) *Scheme*. Use **cond** for selection. [2✓]

```
> (gcd 111 259) (define (gcd x y)
37              (cond ((> x y) (gcd (- x y) y))
> (gcd 11 9)    ((< x y) (gcd x (- y x)))
1              (else x)))
```

- (b) *Ocaml*. [2✓]

```
# gcd;;
- : int -> int -> int = <fun>
# gcd 111 259;; let rec gcd x y =
- : int = 37      if x > y then gcd (x - y) y
# gcd 11 9;;      else if x < y then gcd x (y - x)
- : int = 1       else x;;
```

- (c) *Prolog*. [2✓]

```
| ?- gcd( 111, 259, E).
E = 37 ?
yes
| ?- gcd( 11, 9, E).
E = 1 ?
yes
gcd( X, Y, Z ) :- X > Y, T is X - Y, gcd( T, Y, Z ).
gcd( X, Y, Z ) :- X < Y, T is Y - X, gcd( X, T, Z ).
gcd( X, X, X ).
```

- (d) *Perl*. Prototype the function. [2✓]

```
print gcd (111, 259), "\n";
37
print gcd (11, 9), "\n";
1
sub gcd ($$) {
    my ($x, $y) = @_;
    while ($x != $y) {
        if ($x > $y) {$x -= $y} else {$y -= $x}
    }
    return $x
}
```

- (e) *Smalltalk*. Extend class **Integer** with a keyword method **gcd: .** [2✓]

```
st> 111 gcd: 259.
37
st> 11 gcd: 9.
1
Integer extend [
gcd: other [
|x y|
x := self.
y := other.
[x ~= y] whileTrue: [
x > y ifTrue: [x := x - y]
ifFalse: [y := y - x]
].
^ x
]
```

2. **Bash.** Write a script which will run the file `hzip.st` on a sequence of files in the current directory all of which match `*.in` and put the compressed files in files ending in `*.in.zipped`. Example: `foo.in` will be zipped into `foo.in.zipped`. (Reminder: the `-c` option is used for compression.) [2✓]

```
for file in *.in
do
  hzip.st -c $file $file.zipped
done
```

3. Code a linear search through a data structure, given a comparison function and a key. The data structure contains paired keys and values. Return the value associated with the first key that matches. The examples show what is expected in various cases.

(a) **Ocaml.** [2✓]

```
# find (=) 3 [1,2;3,4;5,6];;
- : int option = Some 4
# find (=) 9 [0,1;2,3;4,5];;
- : int option = None
# find (<) 8 [9,2;7,3;5,9];;
- : int option = Some 2
```

```
let rec find cmp key list = match list with
| [] -> None
| (k,v)::xs -> if cmp key k then Some v
               else find cmp key xs;;
```

(b) **Scheme.** [2✓]

```
> (find = 3 ' ((1 2) (3 4) (5 6)))
4
> (find = 9 ' ((0 1) (2 3) (4 5)))
#f
> (find < 8 ' ((2 2) (9 3) (9 9)))
3
```

```
(define (find cmp key list)
  (cond ((null? list) #f)
        ((cmp key (caar list)) (cadar list))
        (else (find cmp key (cdr list)))))
```

(c) **Prolog.** Use unification to determine equality. [2✓]

```
| ?- find( 3, [pair(1,2),pair(3,4),pair(5,6)], V) .
V = 4 ?
yes
| ?- find( 9, [pair(0,1),pair(2,3),pair(4,5)], V) .
no
```

```
find( K, [pair(K,V)]_ , V).
find( K, [_Tail], V) :- find( K, Tail, V).
```

4. **Smalltalk.** Extend class `Array` with a keyword method `innerprod` which computes the inner product of two arrays. If the sizes of the two arrays are the same, return the inner product. If not, indicate an error with the state-ment: `^ self error: 'innerprod different sizes'`. [2✓]

Mathematically, the inner product p of two arrays a and b is: $p = \sum_i a_i b_i$

```
Array extend [
  innerprod: other [
    self size = other size
    ifFalse: [
      ^ self error: 'innerprod different sizes'
    ]
    ifTrue: [
      |sum|
      sum := 0.
      1 to: self size do: [:i|
        sum := sum + (self at: i) * (other at: i)
      ].
      ^ sum
    ]
  ]
]
```

5. **Perl.** Write a program that reads in numbers from the files specified on the command line, or from the standard input if no files are specified. Scan each line for numbers, and at end of file, print the sum of all the numbers on the line. A number is any sequence of decimal digits, with no decimal point or exponents. Any non-digit characters are ignored. [2✓]

```
-bash-92$ cat /tmp/num
33 44 55
838
100 3
-bash-93$ ./count.perl /tmp/num /tmp/num
2146
-bash-94$ echo 44 55 | ./count.perl
99
```

```
#!/usr/bin/perl
while ($line = <>) {
    $count += $& while $line =~ s/\d+//;
}
print $count, "\n";
```

6. **Prolog.** Reverse a list. [1✓]

```
| ?- reverse([1,2,3,4],L).
L = [4,3,2,1]
yes
```

```
rev([],R,R).
rev([H|T],R,S) :- rev(T,[H|R],S).
```

7. **Bash.** One command using a pipe that lists all of the files in the current directory in long mode, but prints only those files which were modified in the year 2016. [1✓]

```
ls -la | grep 2016
```

8. **Prolog.** Split a list of numbers into the positive list and the negative list. [2✓]

```
| ?- posneg([1,3,-6,0,32,-44,8,0,77],P,N).
N = [-6,-44]
P = [1,3,32,8,77]
| ?- posneg([-3,-5,7,10,0,0,0,44,-2],P,N).
N = [-3,-5,-2]
P = [7,10,44]
```

```
posneg([],[],[]).
posneg([H|T],[H|P],N) :- H > 0, posneg(T,P,N).
posneg([H|T],P,[H|N]) :- H < 0, posneg(T,P,N).
posneg([0|T],P,N) :- posneg(T,P,N).
```

9. **Scheme.** Define `eval` whose argument is either a number or an arithmetic expression. An arithmetic expression is an operator followed by two expressions. This definition is recursive and expressions may be nested arbitrarily deeply. Assume all expressions are well-formed, so do not do any error checking. [2✓]

```
> (eval 3)
3
> (eval (+ 3 4))
7
> (eval (+ (* 3 4) (* 5 6)))
42
> (eval (+ (* 3 4) (/ (+ 8 2) (- 7 6))))
22
```

```
(define (eval e)
  (if (number? e) e
      (apply (car e) (map eval (cdr e)))))
```

10. **Smalltalk.** Extend class `Array` with the keyword message `contains:` which accepts a block with one argument as a predicate. Return `true` if any element of the array, when sent to the predicate via the `value:` message return true. [2✓]

```
st> #(1 3 4 5) contains: [:i|i=3].
true
st> #(1 3 4 5) contains: [:i|i=0].
false
st> #() contains: [:i|i=3].
false
```

```
Array extend [
  contains: predicate [
    1 to: self size do: [:i|
      (predicate value: (self at: i)) ifTrue: [^ true].
    ].
    ^ false
  ]
]
```

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		$= c$

- What makes the script `hzip.st` executable ?
 (A) `chmod +x hzip.st`
 (B) `chmod -x hzip.st`
 (C) `chmod hzip.st +x`
 (D) `chmod hzip.st -x`
- Which system call is used to find the modification time of a file ?
 (A) `ls(1)`
 (B) `readlink(2)`
 (C) `stat(2)`
 (D) `time(2)`
- Which of these languages uses normal order evaluation by default, rather than applicative order evaluation ?
 (A) Haskell
 (B) Ocaml
 (C) Prolog
 (D) Scheme
- Which C, C++, Java operator uses applicative (not normal) order evaluation ?
 (A) `&&`
 (B) `<<`
 (C) `? :`
 (D) `||`
- What is the type of `f` in the declaration :
`let f (x, y) = x +. y;;`
 (A) `float * float -> float`
 (B) `float -> float -> float`
 (C) `int * int -> int`
 (D) `int -> int -> int`
- Lisp, Scheme, and similar languages use ideas derived from :
 (A) α -calculus
 (B) β -calculus
 (C) λ -calculus
 (D) η -calculus

- Smalltalk*. Operator precedence with the highest at the left and lowest at the right is :
 (A) binary, keyword, unary
 (B) binary, unary, keyword
 (C) keyword, binary, unary
 (D) keyword, unary, binary
 (E) unary, binary, keyword
 (F) unary, keyword, binary
- Ocaml*. The function `List.fold_left` on a list of length n takes :
 (A) $O(1)$ stack and $O(1)$ time
 (B) $O(1)$ stack and $O(n)$ time
 (C) $O(n)$ stack and $O(1)$ time
 (D) $O(n)$ stack and $O(n)$ time
- Ocaml*. The function `List.fold_right` on a list of length n takes :
 (A) $O(1)$ stack and $O(1)$ time
 (B) $O(1)$ stack and $O(n)$ time
 (C) $O(n)$ stack and $O(1)$ time
 (D) $O(n)$ stack and $O(n)$ time
- What is used to allow multiple concurrent sequences of instructions in a program such that shared variables require synchronization ?
 (A) kernel
 (B) process
 (C) thread
 (D) vector
- Perl*. What is a comment ?
 (A) `# comment`
 (B) `/* comment */`
 (C) `// comment`
 (D) `(* comment *)`
- Smalltalk*. What is used to dynamically add new methods to class `Integer` ? (Assume that \cdots represents some valid code.)
 (A) `Integer extend [...].`
 (B) `class Integer: Object [...].`
 (C) `extend class Integer with [...].`
 (D) `interface extends Integer [...].`



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		$= c$

1. Which language is purely object-oriented ?

- (A) C++
- (B) Java
- (C) Ocaml
- (D) Smalltalk

2. What regular expression will match one or more white-space characters ?

- (A) `\S+`
- (B) `\W+`
- (C) `\s+`
- (D) `\w+`

3. What is the first line of a Perl script ?

- (A) `#!/usr/bin/perl`
- (B) `#!/bin/bash perl`
- (C) `#!/usr/bin/perl`
- (D) `#!/usr/bin/perl`

4. *Perl*. Which expression will find the value associated with the key i in the hash a ?

- (A) `$a($i)`
- (B) `$a<$i>`
- (C) `$a[$i]`
- (D) `$a{$i}`

5. Which of these languages is compiled (not interpreted) ?

- (A) Ocaml
- (B) Perl
- (C) Prolog
- (D) Smalltalk

6. *Perl*. Which of the following is an array ?

- (A) `$a`
- (B) `%a`
- (C) `*a`
- (D) `@a`

7. Which of the following **Makefiles** will compile an arbitrary C program when given the name of the executable binary as an operand to **make** ?

Example: `make foo`

- (A) `%.c: %
gcc $@ -o $$`
- (B) `%.c: %.c
gcc $< -o $@`
- (C) `%.c: %.c
gcc %.c`
- (D) `*: *.c
gcc $< >$@`

8. What will redirect both the **stdout** and **stderr** of program **foo** into the file **bar** ?

- (A) `foo 1>&2 >bar`
- (B) `foo 2>&1 >bar`
- (C) `foo >bar 1>&2`
- (D) `foo >bar 2>&1`

9. *Haskell*. Define **filter** so that

Prelude> `filter (>0) [1,3,-2,-8,7]`
[1,3,7]

- (A) `let filter p z = match z with [] -> []
| x::xs -> filter p xs;;`
- (B) `let filter p z = [x <- z | x > 0]`
- (C) `let filter p z = [x | x <- z, p x]`
- (D) `let filter p z = if null p x then []
else p cdr z`

10. Perl. What will read the next line from the standard input if no files are specified in `@ARGV`, but if files are specified in `@ARGV`, will read the next line from the current file, sequencing through these files in the order given ?

- (A) `while ($line = <*>)`
- (B) `while ($line = <>)`
- (C) `while ($line = <@ARGV>)`
- (D) `while ($line = <STDIN>)`

11. *Ocaml*. `# (=);;`

- (A) `- : 'a * 'a -> bool = <fun>`
- (B) `- : 'a -> 'a -> bool = <fun>`
- (C) `- : int * int -> bool = <fun>`
- (D) `- : int -> int -> bool = <fun>`

12. Templates in C++ and generics in Java are an example of ____ polymorphism.

- (A) ad-hoc parametric
- (B) universal ad-hoc
- (C) universal conversion
- (D) universal parametric