$Id: cmps112-2017q4-final.mm,v 1.87 2017-12-01 15:30:57-08 - - $

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. For each language described here, fill in the name of the language. Choose from among the following languages : AWK, Ada, Algol 60, BCPL, Bash, Basic, C++, C, COBOL, FORTRAN, Forth, Haskell, Intercal, Java, Lisp, ML, OCaml, PL/I, Pascal, Perl, Prolog, Scheme, Simula 67, Smalltalk. Grading : deduct 0.5 point for each wrong or missing answer, but do not score less than 0. **[2✔]**

C++	Bjarne Stroustrup's most noted contribution to language design.
COBOL	Business data processing language, designed by Grace Hopper (1959).
Java	Designed at Sun Microsystems, James Gosling leading.
Scheme	Mostly-functional statically-scoped language descended from Lisp.
FORTRAN	Numeric and scientific computation language developed at IBM (1957).
Simula 67	Simulation language that influenced the design of C++.
Ada	DOD language named for Lord Byron's daughter, the Countess of Lovelace.

2. Some examples of polymorphism are given in the following table. In each box, write both the general and specific category using the terms : ***universal***, ***ad hoc***, ***conversion***, ***parametric***, ***inclusion***, ***overloading***. **[2✔]**

`class shape { }` universal inclusion `class square extends shape { }`	`class stack<item> { };` `stack<Integer> si;` universal parametric
`void foo (double d);` `void foo (int i);` ad hoc overloading	`double sqrt (double);` `n = sqrt (6);` ad hoc conversion

3. ***Smalltalk.*** Extend the class **`Array`**, that is, add a new method to the existing class array. The keyword method **`merge`** takes a single other array as an argument and produces a new array containing all the elements of both arrays in sorted order using the binary operator **`<`**. Assume both arrays are already in sorted order. Of course, you need to create a new array and cycle through the old arrays copying elements of each of the input arrays. And when one array runs out, copy all elements of the other array. **[6✔]**

```
st> #(1 3 5 7 9) merge: #(2 4 6 8 10).
(1 2 3 4 5 6 7 8 9 10 )
st> #() merge: #(1 2 3).
(1 2 3 )
st> #(4 44 444 999) merge: #()
(4 44 444 999 )
```

```
Array extend [
  merge: other [
    |result si oi ri|
    result := Array new: self size + other size.
    si := 1.  oi := 1.  ri := 1.
    [(si <= self size) & (oi <= other size)]
    whileTrue: [
      (self at: si) < (other at: oi)
      ifTrue: [ result at: ri put: (self at: si).
              si := si + 1.
      ]
      ifFalse: [ result at: ri put: (other at: oi).
               oi := oi + 1.
      ].
      ri := ri + 1.
    ].
    [si <= self size] whileTrue: [
      result at: ri put: (self at: si).
      si := si + 1.
      ri := ri + 1.
    ].
    [oi <= other size] whileTrue: [
      result at: ri put: (other at: oi).
      oi := oi + 1.
      ri := ri + 1.
    ].
    ^ result.
  ]
]
```

4. ***Ocaml.*** Define the function **merge** which merges two sorted lists according to a predicate and produces a resulting list. Assume the argument list are sorted. **[3✔]**

```
# merge (<) [1;3;5;7;9] [2;4;6;8];;
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9]
# merge (>) [9;5;3;1] [77;2;-5];;
- : int list = [77; 9; 5; 3; 2; 1; -5]
# merge (<) [] [1;2;3];;
- : int list = [1; 2; 3]
```

```
let rec merge less ls1 ls2 = match ls1, ls2 with
  | [], ls2 -> ls2
  | ls1, [] -> ls1
  | x::xs, y::ys ->
   if less x y then x :: merge less xs ls2
          else y :: merge less ls1 ys;;
```

5. ***Scheme.*** Define the function **merge** which merges two sorted lists according to a predicate and produces a resulting list. Assume the argument list are sorted. **[3✔]**

```
> (merge < '(1 3 5 7 9) '(2 4 6 8))
(1 2 3 4 5 6 7 8 9)
> (merge > '(88 77 66) '(99 70 55))
(99 88 77 70 66 55)
> (merge < '(1 3 5 7) '())
(1 3 5 7)
```

```
(define (merge less ls1 ls2)
    (cond ((null? ls1) ls2)
        ((null? ls2) ls1)
        (else (if (less (car ls1) (car ls2))
              (cons (car ls1) (merge less (cdr ls1) ls2))
              (cons (car ls2) (merge less ls1 (cdr ls2)))))))
```

6. ***Smalltalk.*** Define a class **Max** which has a keyword class method **max** whose argument is an array. It returns the maximum value in the array. If the array is empty, it returns **nil**. **[4✔]**

```
st> Max max: #(3.14159265358979 1.4142135623730951 2.718281828459045).
3.14159265358979
st> Max max: #(3 1 4 1 5 9 2 6 5 3 5).
9
st> Max max: #('hello' 'world' 'foo' 'bar' 'baz').
'world'
st> Max max: #().
nil
```

```
Object subclass: Max [
  Max class >> max: array [
    array size = 0
    ifTrue: [ ^ nil ]
    ifFalse: [
      |max|
      max := array at: 1.
      2 to: array size do: [:i|
        (array at: i) > max ifTrue: [ max := array at: i ].
      ].
      ^ max.
    ].
  ]
].
```

7. ***Prolog.***

   (a) Define `length`. **[1✔]**

```
| ?- length([],N).
N = 0
| ?- length([1,2,3,4,5],N).
N = 5
```

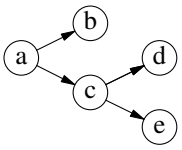
```
length([],0).
length([_|T],N) :- length(T,M), N is M + 1.
```

   (b) Define `sum`. **[1✔]**

```
| ?- sum([],N).
N = 0
| ?- sum([1,2,3,4,5],N).
N = 15
```

```
sum([],0).
sum([H|T],N) :- sum(T,M), N is M + H.
```

8. ***Prolog.*** Write a list of facts such that `arrow(X,Y)` indicates that there is an arrow `X` → `Y`. Write a predicate `path(X,Y)` which says "yes" if there is a path from `X` to `Y`. There is always a path from a node to itself. Assume that the graph is acyclic (no cycles). **[2✔]**




```
arrow(a,b).
arrow(a,c).
arrow(c,d).
arrow(c,e).
path(X,X).
path(X,Y) :- arrow(X,U), path(U,Y).
```

9. If `mother(M,X)` and `father(F,X)` means that `M` and `F` are the parents of `X` (respectively), and that the database contains facts `female(X)` and `male(X)`, define the following predicates:

   (a) `parent(X,Y)` if `X` is the parent of `Y`. **[1✔]**

```
parent(X,Y) :- mother(X,Y).
parent(X,Y) :- father(X,Y).
```

   (b) `sister(X,Y)` if `X` is the sister (including half-sister) of `Y`. **[1✔]**

```
sister(X,Y) :- parent(P,X), parent(P,Y), female(X).
```

10. Define the function `map2` which takes a function and a pair of lists and returns a list by applying the function pairwise to the elements of the list and returns new list in the same order as the input lists. If the lists are of different length, ignore excessive elements of the longer list.

    (a) ***Scheme.*** **[2✔]**

```
> (map2 + '(1 2 3 4) '(5 6 7))
(6 8 10)
> (map2 * '(1 2 3 4 5) '(6 7 8 9))
(6 14 24 36)
> (map2 / '(1 2 3 4) '())
()
```

```
(define (map2 f l1 l2)
    (if (or (null? l2) (null? l2)) '()
        (cons (f (car l1) (car l2))
              (map2 f (cdr l1) (cdr l2)))))
```

    (b) ***Ocaml.*** Use pattern matching. Do not use `List.hd` or `List.tl`. **[2✔]**

```
# map2;;
- : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list = <fun>
# map2 (+) [1;2;3;4] [5;6;7];;
- : int list = [6; 8; 10]
# map2 ( * ) [1;2;3;4;5] [6;7;8;9];;
- : int list = [6; 14; 24; 36]
# map2 (/) [1;2;3;4] [];;
- : int list = []
```

```
let rec map2 f l1 l2 = match l1, l2 with
  | [], _ -> []
  | _, [] -> []
  | x::xs, y::ys -> f x y :: map2 f xs ys;;
```

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 is the type of `f`:
```
let f (x, y) = x + y;;
```
(A) `f : int * int * int`
(B) `f : int * int -> int`
(C) `f : int -> int * int`
(D) `f : int -> int -> int`

2. What is the type of `g`:
```
let g x y = x * y;;
```
(A) `f : int * int * int`
(B) `f : int * int -> int`
(C) `f : int -> int * int`
(D) `f : int -> int -> int`

3. What is the type of `inc`?
```
let inc = (+) 1;;
```
(A) `inc : 'a int`
(B) `inc : int * int`
(C) `inc : int -> int`
(D) `inc : int <'a>`

4. Smalltalk determines if an object can respond to a message by:
(A) duck typing
(B) its interface
(C) its template declaration
(D) parametric polymorphism

5. Prolog:
```
| ?- X is cos(pi).
```
(A) `X = -1.0`
(B) `X = 1.2246467991473532e-16`
(C) `X = 3.1415926535897931`
(D) `no`

6. What is equivalent to just `a` itself?
(A) `(cons (car a) (cdr a))`
(B) `(car (cons a) (cdr a))`
(C) `(cons (cdr a) (car a))`
(D) `(cdr (cdr a) (cons a))`

7. Prolog determines the values of expressions by:
(A) lazy evaluation
(B) template parameters
(C) type inference
(D) unification

8. What is `(1 2 3 4)`
(A) `(apply * '(1 2 3 4))`
(B) `(cons * '(1 2 3 4))`
(C) `(foldl * '(1 2 3 4))`
(D) `(map * '(1 2 3 4))`

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

10. What has type `int list` in Ocaml?
(A) `(1,2,3,4);;`
(B) `(1;2;3;4);;`
(C) `[1,2,3,4];;`
(D) `[1;2;3;4];;`

11. A function like `fold_left` is tail recursive. How much stack space does it use?
(A) $O(1)$
(B) $O(\log_2 n)$
(C) $O(n)$
(D) $O(n \log_2 n)$

12. If `fold_left` takes a function whose execution time is $O(1)$ and is applied to a list of length $O(n)$, how much time will it take?
(A) $O(1)$
(B) $O(\log_2 n)$
(C) $O(n)$
(D) $O(n \log_2 n)$

> 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/EWD498.html`

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

2. C++ templates are implemented by :
   (A) compiling all template parameters as type Object.
   (B) recompiling each template separately for each different parameter.
   (C) tagging all non-pointer fields in an object.
   (D) using a virtual function table.

3. Parametric polymorphism is implemented in Java by :
   (A) compiling all generic parameters as Objects.
   (B) recompiling each generic parameter separately for each different parameter.
   (C) tagging all non-pointer fields in an object.
   (D) using a virtual function table.

4. "Lazy" evaluation is also known as ____ order evaluation.
   (A) applicative
   (B) functional
   (C) normal
   (D) unified

5. PL/I's non-local goto (unwinding the function call stack several levels) can be evaluated in C++ and Java using :
   (A) `break`
   (B) `continue`
   (C) `return`
   (D) `throw`

6. In Smalltalk, the value of an uninitialized variable is :
   (A) `0`
   (B) `nan`
   (C) `nil`
   (D) `nullptr`

7. In Smalltalk, what is 5 ?
   (A) `(1 + 4) value.`
   (B) `<1 + 4> value.`
   (C) `[1 + 4] value.`
   (D) `{1 + 4} value.`

8. What is a Scheme comment ?
   (A) `(*...*)`
   (B) `/*...*/`
   (C) `//...`
   (D) `;;...`

9. Given the following Smalltalk definition, what returns 4 ?
   ```
   a := [:x| x + 1].
   ```
   (A) `a 3.`
   (B) `a at: 3.`
   (C) `a value: 3.`
   (D) `a x: 3.`

10. Who write code for Charles Babbage's difference engine (if it had been built) ?
    (A) Alonzo Church
    (B) Grace Hopper
    (C) Ada Lovelace
    (D) Alan Turing

11. Which C++ operator is ***not*** lazy ?
    (A) `&&`
    (B) `<<`
    (C) `? :`
    (D) `||`

12. Which can be a Prolog fact ?
    (A) `foo(X,Y).`
    (B) `foo(X,y).`
    (C) `foo(x,Y).`
    (D) `foo(x,y).`

I found the UCSC campus not an inspiring place, and the longer I stayed there, the more depressing it became. The place seemed most successful in hiding all the usual symptoms of a seat of learning. In the four-person apartment we occupied, only one of the four desks had a reading lamp, and the chairs in front of the desks were so low that writing at the desks was not comfortable. Probably it doesn't matter. Can UCSC students write ? Do they need to ? The notice boards showed ads from typing services "Grammar and spelling corrected.". (One of these ads itself contained a spelling error !) Blackboards were hardly sufficient ; there used to be neither bucket not sponge ; the acoustics were from bad to terrible ; the PA systems in the large lecture hall and in the theatre were inadequate ; ... in the one and only public place where we could meet in the evening — "Idler's Cafe" — conversation was impossible as the result of the summum of vulgarity : a row of noisy pin-ball machines ! And — as I am told : even by dormitory standards — the food was terrible. ...

— EWD714 : "Trip report E.W.Dijkstra, Mission Viejo, Santa Cruz, Austin, 29 July – 8 September 1979."
prof. dr. Edsger W. Dijkstra, 14 September 1979,
Plataanstraat 5, 5671 AL NUENEN, The Netherlands
`https://www.cs.utexas.edu/users/EWD/transcriptions/EWD07xx/EWD714.html`