CC41A Lenguajes de Programación
Control 2 - Otoño 2006
Profesor: Luis Mateu

Pregunta 1

(a) Se tiene el siguiente programa escrito en un lenguaje hipotético:

```
main( ) {
  a=1
  c= f(a, a)
  print a, c
}
```

```
f(x, y) {
  x= x+1
  y= y+1
  return y
}
```

Indique qué despliega este programa cuando en el lenguaje los parámetros se pasan (i) por valor, (ii) por referencia, (iii) por valor resultado.

(b) Se definen las clases U y V de la siguiente manera:

```
class U {
  void m(Object x, Object y)
    ... // (1)
  void m(Object x, String s)
    ... // (2)
}
```

```
class V extends U {

  void m(Object x, String s)
    ... // (3)
  void m(String s, Object y)
    ... // (4)
}
```

```
V vv= new V(); U uv= vv;  Object oo= new Object();
```

Para las siguientes llamadas indique si hay algún error o el número del método que (i) el compilador selecciona estáticamente para ser invocado y (ii) el que se invoca en tiempo de ejecución:

```
i.   uv.m(oo, "abracadabra");
ii.  Object os="abracadabra"; uv.m(oo, os)
iii. uv.m("abracadabra", oo);
iv.  uv.m("abra", "cadabra");
v.   vv.m("abra", "cadabra");  //  vv! no uv.
```

No trate de adivinar: una respuesta incorrecta invalida una correcta. No necesita explicar sus respuestas.

Pregunta 2

La siguiente es una especificación en Scheme para el tipo de datos abstracto *cola de prioridad*:

- `(make-queue p)`: Entrega una cola de prioridad *Q* sin elementos. El parámetro **p** es un procedimiento tal que **(p a b)** indica si **a** tiene mejor prioridad que **b**. Este procedimiento será utilizado en todas las colas que se deriven de *Q*. Ejemplo:

```
(define q0 (make-queue (lambda (a b) (< a b))))
```

- `(put e q)`: Entrega la cola de prioridad que resulta de agregar `e` a `q` (sin modificar `q`). Ejemplo:

```
(define q53 (put 5 (put 3 q0)))
(define q253 (put 2 q53))
```

- `(best q)`: Entrega el elemento de mejor prioridad en `q`. Ejemplo:

```
(best q253) => 2
(best q53) => 3
```

- `(remove-best q)`: Entrega la cola de prioridad que resulta de eliminar de `q` el elemento de mejor prioridad. Ejemplos:

```
(best (remove-best q53)) => 5
(best q53) => 3
```

- `(empty q)`: Entrega #t si `q` es una cola de prioridad vacía o #f en caso contrario. Ejemplos:

```
(empty q0) => #t
(empty q53) => #f
```

Implemente en Scheme puramente funcional el tipo de datos abstracto cola de prioridad. Represente una cola de prioridad mediante una lista en que el primer elemento es el procedimiento que compara prioridades. El resto de la lista son los elementos ordenados de la mejor prioridad hacia la peor.

Pregunta 3

i. Un conjunto de persona se representa mediante una lista de pares nombre y edad. Programe en Scheme un procedimiento `show` que use la cola de prioridad de la pregunta 2 (y *no* otro algoritmo de ordenamiento) para que dado un conjunto de personas, entregue una lista con los nombres de las personas ordenadas de mayor a menor edad. Ejemplo:

```
(show '((pedro 25) (ximena 16) (juan 17) (diego 32)))
     => (diego pedro juan ximena)
```

ii. Escriba en Scheme un procedimiento `iterator` que toma como parámetro una lista y entrega como resultado otro procedimiento que cada vez que se invoca entrega ordenamente los elementos de la lista. Ejemplo:

```
(define iter1 (iterator '(a b c)))
(iter1) => a
(iter1) => b
(define iter2 (iterator '(1 2 3 4 5)))
(iter2) => 1
(iter1) => c
```

Explique además por qué su implementación no es puramente funcional.