Examen Lenguajes

P1:

a) Que los objetos, al igual que las funciones, estarían definidas en una sección del programa, por lo tanto no se podrían entregar objetos como valor y no sería posible que hayan objetos que referencien a otros y que compartan sus datos. En definitiva, no serían objetos.

b) Guardando en memoria los valores modificados de los argumentos. La memoria estaría teniendo el comportamiento de un store, y ya que la memoria es compartida para un proceso y las funciones que ejecuta (y potencialmente para todo el sistema), los cambios se ven reflejados fuera de la función, siendo el proceso el que puede acceder a dichos cambios.

c) Lo que hace la evaluación perezosa es dejar en suspenso la evaluación de una sentencia hasta que el valor de la sentencia es requerido, entonces ahí recién se fuerza su evaluación. En efecto, al querer usar mutación, no nos sirve dejar la evaluación de una sentencia para después, ya que eso haría que no se efectúe la mutación, generando un posible error después.
Uno podría querer arreglarlo al hacer que las funciones que ocupen mutación hagan la evaluación al tiro, haciendo del lenguaje una mezcla entre eager y lazy, pero el tema es que el problema no se arregla, ya que esa función podría ser entregada como un valor por otra función que no use estrictamente mutación, por lo tanto dejaría la evaluación en pausa… obteniendo el mismo problema anterior.

d)
```
(with  (x (newbox  2))
    (with  (f  (lambda  (y)  (set!  y  4)))
      (f  x)
      (openbox x)))
```
En este caso, la evaluación de f quedaría en suspenso, por lo tanto (openbox x) retornaría 2.

e) Debido a los side effects. Un programa que va modificando su estado tras su ejecución, puede no comportarse exactamente igual en su n-ésima ejecución, por lo tanto para asegurarse que el programa es correcto, hay que hacer un trabajo más minucioso y por lo tanto, más difícil.

f) TCO se refiere a cuando una función retorna un valor independiente del estado final de la función, es decir, no se deben guardar los datos que creó la función en la pila, sino que al retornar se eliminan esos datos de la pila. En este caso, la pila crea un frame para la función, que al retornar puede ser eliminado o sobrescrito por una nueva función, llamada en el retorno por la primera función.
TRO es un caso particular de TCO, donde el retorno de la función es no solamente un valor independiente a la función, sino que es una nueva llamada a la misma función (también independiente), es decir, es una función recursiva que aprovecha TCO.

g) Son macros, que se ejecutan en el preprocesador. Si éstas fuesen funciones, serían ejecutadas en el intérprete, directamente después de pasar por el parser, lo que podría

llevar a problemas en un lenguaje eager, ya que no tendríamos control de la evaluación de las sentencias. Por ejemplo, en el caso de if, si fuera una función en un lenguaje eager, el "then" del if sería evaluado antes de considerar si se debe evaluar o no. Por el contrario, en un lenguaje lazy, no es estrictamente necesario tener macros, ya que sí se puede tener el control de cuándo ejecutar una sentencia.

h) No, ya que un Store tendría que ser una función que "calcule" de alguna manera las asociaciones, o bien, que sea una función que vaya a buscar la asociación a una estructura "type"… pero entonces esa estructura vendría lo que se cataloga como store. Luego, como no es posible calcular una asociación, entonces los stores no se pueden considerar como una representación procedural.

i) En un lenguaje con funciones de primera clase, no. La clausura serviría para guardar una función y su environment al momento de definirla, pero basta con que subst sustituya las variables libres en el cuerpo de la función. Por otro lado, si se tienen funciones de primer orden, entonces sí se necesitan las clausuras, de lo contrario no se sabría cómo sustituir.