PREGUNTAS EXAMEN PF

1. ¿Cuándo una función se considera de orden superior?
   Se considera una FOS si devuelve una función, devuelve una estructura que contiene una función o alguno de sus argumentos es una función.

2. Justifica razonadamente la certeza de la siguiente afirmación: “Las S-expresiones sólo pueden ser pares o átomos”.
   No es cierto, ya que () es un átomo, pero por ejemplo en la lista (s1 . L) el cdr es otra lista, mientras que en la lista (s1 . s2), s2 no es una lista.

3. Justifica razonadamente la diferencia entre la evaluación ansiosa y perezosa de la siguiente expresión, e indica que lenguajes soportan cada una de ellas: f (1/0,5)
   Si tenemos una estructura de función tal que: f (g(...) , h(...))
   Evaluación ansiosa -> Se analizan primero los argumentos g(...) y h(...).
   Evaluación perezosa -> solo se analizan los argumentos si es necesario f.

   f (x,y)::== si x>y entonces x
   sino y

   f (2,1/0) ->Ansiosa = error

   -> Perezosa = 2

4. ¿En qué caso una función invocada con menos argumentos de los indicados en su definición no devuelve error? Pon un ejemplo.
   Esto es posible si la función está definida utilizando el "rest parameter", que permite que la función acepte un número variable de argumentos.

   Por ejemplo, considera la función sum que suma una lista de números:

   (define (sum . nums)
   (apply + nums))

   (sum) ; Devuelve 0
   (sum 1 2 3) ; Devuelve 6
   (sum 5 10) ; Devuelve 15

5. ¿Analiza y justifica la siguiente afirmación: “la FOS filter de Scheme puede considerarse un patrón de programación”.
   Correcto, puesto que si abstraemos la sintaxis tenemos: filter **<función>** lista, donde función puede ser cualquier función definida en Scheme o incluso funciones lambda.

6. Justifica razonadamente la certeza de la siguiente afirmación: “En programación funcional pura, los símbolos locales a la evaluación de una función son únicamente los argumentos de esta”.
   La afirmación es cierta, ya que las funciones en este paradigma están restringidas a operar únicamente en los valores pasados como argumentos y no tienen acceso a ningún otro símbolo local a menos que esté definido en su alcance léxico.

7. ¿En qué consiste la técnica de Currying? Pon un ejemplo.
   Consiste en definir una función de n argumentos como n funciones de 1 argumento. Devuelve una expresión lambda con x argumentos ya introducidos que espera por el resto:

   ```
   (define (op x)
     (curry + 5)
   ```

8. Con la función letrec de Scheme nos permite asociar a un símbolo X una expresión que contiene al propio símbolo X que estamos definiendo. ¿Qué finalidad tiene? Pon un ejemplo.
   Se utiliza para definir variables locales mutuamente recursivas, es decir, variables que se refieren entre sí en sus definiciones. Esto es útil para definir varias funciones o valores que dependen unas de otras en un solo bloque de código, lo que mejora la legibilidad y la organización del código. Un ejemplo típico es definir funciones que se llaman entre sí, como funciones para determinar si un número es par o impar.