Lenguajes de Programación.
Prueba 1.
Lunes 10/05/21.
Profesor: Paul Leger
Ayudante: Sebastián Arce

Pregunta 1.

V1. Los lenguajes de programación que son [generales / específicos / **Turing-complete /** cuánticos] permiten expresar lo máximo que un programa computacional puede realizar.

V2. Los lenguajes de programación NO contienen el siguiente elemento:

a. Sintaxis
b. Librerías
c. Expresividad
d. Semántica

Pregunta 2.

V1. Un *idiom* permite a los programadores usar [más funciones / **convenciones** / patrones de programaciones / orientación a objetos].

V2. Mientras sintaxis es lo que [**el programador escribe** / el programa significa / el interprete ejecuta / sistema operativo ejecuta] y la semántica es lo que [el programador escribe / el programa significa / el interprete ejecuta / sistema operativo ejecuta]

Pregunta 3.

V1. Indique cuál NO es un paradigma de programación visto en clases:

a. Future-Oriented Programming
b. Object-Oriented Programming
c. Functional Programming
d. Aspect-Oriented Programming

V2. Event-Oriented Programming es usado para escribir:

a. Programas que captura los eventos del sistema operativo
b. Programas que captura los eventos del usuario
c. Programas en base los eventos que puede recibir
d. Event-Oriented Programming no existe

Pregunta 4.

V1. Los valores de primera clase pueden:
a. Ser asignados a una clase.
b. Crear clases en tiempo de ejecución.
c. Ser asignados a una variable.
d. Creados en tiempo de ejecución.

V2. Los valores de orden superior pueden:
a. Ser asignados a una clase.
b. Crear clases en tiempo de ejecución.
c. Ser asignados a una variable.
d. Creados en tiempo de ejecución.

Pregunta 5.

V1. Mientras Java tiene un Scope [**fuertemente tipado** / débilmente tipado / Gradualmente tipado], Python tiene un Scope [fuertemente tipado / **débilmente tipado** / Gradualmente tipado].

V2. Mientras Java tiene una estrategia de evaluación [**temprana (eager)** / flojamente (lazy)], Python tiene un Scope [temprana (eager) / **flojamente (lazy**)].

Pregunta 6.

V1. La expresión “x = (function (x) {return x + 1;}) (10)” entrega:
a. Función.
b. 10.
c. 11.
d. Un error.

V2. La expresión “x = (function (x, y) {return x + y + 1;}) (10,2)” entrega:
a. Función.
b. 13.
c. 10.
d. Un error.

Pregunta 7.

V1. Escriba una función que muestre la suma de los números primos. Asuma que exista una función *(primo n)* que retorna verdadero si n es primo, falso en otro caso. Por ejemplo, (sumar-primos '(6 2 5 7 9 10)) -> 14.

a.

```
(define (sumar-primos lista)
  (cond
    ((= (length lista) 0) 0)
    ((primo (car lista)) (+ (car lista)  (sumar-primos (cdr lista))))
    (else (sumar-primos (cdr lista)))))
```

b.

```
(define (sumar-primos lista)
  (cond
    ((primo (car lista)) (+ (car lista)  (sumar-primos (cdr lista))))
    (else (sumar-primos (cdr lista)))))
```

c.

```
(define (sumar-primos lista)
  (cond
    ((= (length lista) 0) 0)
    ((primo (car lista)) (+ (car lista)  (sumar-primos (cddr lista))))
    (else (sumar-primos (cdr lista)))))
```

d.

```
(define (sumar-primos lista)
  (cond
    ((= n c) #t)
    ((= (remainder n c) 0) #f)
    (else (primo1 n (+ c 1)))))
```

V2. Escriba una función que filtren los números primos. Asuma que exista una función *(primo n)* que retorna verdadero si n es primo, falso en otro caso. Por ejemplo, (filtrar-primos '(6 2 5 7 9 10)) -> (2 5 7).

a.

```
(define (filtrar-primos lista)
  (cond
    ((= (length lista) 0) '())
    ((primo (car lista)) (cons (car lista)  (filtrar-primos (cdr lista))))
    (else (filtrar-primos (cdr lista)))))
```

b.

```
(define (filtar-primos lista)
  (cond
    ((= (length lista) 0) 0)
    ((primo (car lista)) (+ (car lista)  (filtrar-primos (cdr lista))))
    (else (filtrar-primos (cdr lista)))))
```

c.

```
(define (filtrar-primos lista)
  (cond
    ((primo (car lista)) (cons (car lista)  (filtrar-primos (cdr lista))))
    (else (filtrar-primos (cdr lista)))))
```

d.

```
(define (filtrar-primos lista)
  (cond
    ((= n c) #t)
    ((= (remainder n c) 0) #f)
    (else (primo1 n (+ c 1)))))
```

Pregunta 8.

V1. ¿Cuáles de las siguientes expresiones es un par en Scheme?

a. ‘(1 2 3)

b. ‘(1 2)

c. ‘(1 . 2)

d. ‘(1 (2))

V2. ¿Qué significa la siguiente expresión: (>> 5 a)?

a. **La aplicación de una función >> que aplica a los parámetros 5 a.**

b. Una lista con tres elementos.

c. Un error.

d. La definición de una función

Pregunta 9.

V1. Un árbol de sintaxis abstracta es usado por:

a. El interprete.

b. El sistema operativo.

c. El parser.

d. No existe árbol de sintaxis abstracta.

V2. Una sintaxis concreta es usada por:
a. El interprete.
b. El sistema operativo.
c. El parser.
d. No existe árbol de sintaxis abstracta.