Lenguajes de Programación.
Prueba 1.
Lunes: 18/10/21
Profesor: Paul Leger
Ayudante: Scarlet Tobar

Pregunta 1.

V1. Los lenguajes de programación que NO son [generales / específicos / **Turing-complete /** cuánticos] tienen problemas para expresar todo lo posible que un computador puede hacer.

V2. Los lenguajes de programación contienen los siguientes elementos:

a. Sintaxis, Librerías, Idioms, expresividad
b. Librerías, Sintaxis, procesamiento, semántica
c. Sintaxis, Librerías, Idioms, semántica
d. Semántica

semantica: paradigma , firs class value,high order value,type system,scope,estrategia evaluacion(iger , lazy)

idioms: convensiones para programar(cada programa tiene su propio estandar)

Pregunta 2.

sintaxis: lo que uno escribe

V1. Un programador que NO sabe de los *idioms* de un particular lenguaje de programación generalmente comete errores de [funciones / **convenciones** / patrones de programaciones / orientación a objetos] al programar.

V2. Mientras sintaxis influye en [**lo escrito** / el significado / lo ejecutado / el sistema operativo] y la semántica influye en [lo escrito / **el significado** / lo ejecutado / el sistema operativo] de un programa.

Pregunta 3.

V1. Indique cuál paradigma de programación puede ser usado en conjunto a otro paradigma:

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

V2. React-Oriented Programming es usado para escribir:

a. Programas que reaccionan a los eventos del sistema operativo en un IDE
b. Programas que reaccionan a los eventos del usuario
c. Programas donde los valores de variables cambian de acuerdo con los cambios de valores de otras variables
d. React-Oriented Programming no existe

Pregunta 4.

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

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

Pregunta 5.

V1. Mientras C es [**estatícamente tipado** / débilmente tipado / Gradualmente tipado], C++ es [**estatícamente tipado** / débilmente tipado / Gradualmente tipado].

V2. Mientras JavaScript es orientación a objetos basados en [**prototipos** / clases], Python es orientación a objetos basados en [prototipos / **clases**].

Pregunta 6.

V1. La expresión “((lambda (x y) (+ x y)) 4 6)” entrega:
a. 9.
b. 10.
c. 11.
d. Un error.
e. Función (clousure)

V2. La expresión “(lambda (x y) (+ 4 6))” entrega:
a. 9.
b. 10.
c. 11.
e. Función (clousure)

Pregunta 7.

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

V1. Escriba una función que calcule el producto punto (mirar imagen) entre dos listas. Por ejemplo, (calcular-producto-punto '(6 2 5) ‘(2 0 10)) -> 62. NOTA: Puede asumir que ambas listas tienen el mismo número de elementos.

a.

```
(define (calcular-pp l1 l2)
    (if (= (length l1) 0)
        0
        (+ (* (car l1) (car l2)) (calcular-pp (cdr l1) (cdr l2)))))
```

b.

```
(define (calcular-pp l1 l2)
    (if (= (length l1) 0)
        0
        (* (+ (car l1) (car l2)) (calcular-pp (cdr l1) (cdr l2)))))
```

c.

```
(define (calcular-pp l1 l2)
    (if (= (length l1) 0)
        0
        (+ (* (cdr l1) (cdr l2)) (calcular-pp (car l1) (car l2)))))
```

e.

```
(define (calcular-pp l1 l2)
    (if (>= (length l1) 0)
        0
        (+ (* (car l1) (car l2)) (calcular-pp (cdr l1) (cdr l2)))))
```

V2. Escriba una función (o varias funciones) que determine si un número es perfecto o no, es decir, retorna *verdadero* si el número es perfecto, *falso* en otro caso. Un número perfecto es aquel número que la suma de sus divisores exactos es igual al número, por ejemplo, los divisores de 6 suman 6 (= 3 + 2 + 1).

a.

```
(define (perfecto n)
  (= n (p-f n 2)))

(define (p-f n c)
   (cond
   ((= n c) 1)
   ((= (mod n c) 0) (+ c (p-f n (+ c 1))))
   (else (p-f n (+ c 1)))))
```

b.

```
(define (perfecto n)
  (= n (p-f n 2)))

(define (p-f n c)
   (cond
   ((= n c) 0)
   ((= (mod n c) 0) (+ c (p-f n (+ c 1))))
   (else (p-f n (+ c 1)))))
```

c.

```
(define (perfecto n)
  (= n (p-f n 2)))

(define (p-f n c)
   (cond
   ((= n c) 1)
   ((= (mod n c) 1) (+ c (p-f n (+ c 1))))
   (else (p-f n (+ c 1)))))
```

d.

```
(define (perfecto n)
  (= n (p-f n 2)))

(define (p-f n c)
   (cond
   ((= n c) 1)
   ((= (mod n c) 0) (+ c (p-f n (+ c 1))))
   (else (p-f n (+ c 2)))))
```

Pregunta 8.
V1. ¿Cuáles de las siguientes expresiones es la aplicación de una función?
a. ‘(+ 2 3)
b. (+ 2)
c. ‘(5 + 2)
d. (“+” 5 4)

V2. ¿Qué significa la siguiente expresión: (>= 10 5)?
a. **La aplicación de una función >= que aplica a los parámetros 10 5.**
b. Una lista con tres elementos.
c. Un error.
d. La definición de una función

Pregunta 9.

V1. El interprete usa el siguiente elemento para ejecutar un programa:
a. Sintaxis concreta.
b. Árbol de sintaxis abstracta. lse traduce nuestro codigo a un lenguaje intermedio que el interprete escapaz de entender
c. Un compilador.
d. No existe el interprete.

V2. Un parser usa el siguiente elemento para funcionar:
a. La sintaxis concreta de un programa.
b. Un árbol de sintaxis concreta.
c. Un compilador.
d. No siempre el parser usa un elemento para funcionar.

[NO CAMBIAR EL ORDEN DE LAS RESPUESTAS]
Pregunta 10.

V1. Un lenguaje que soporta un sistema fuertemente tipado permite:
a. Optimizar el código.
b. Verificar si un código está bien escrito en términos de tipos.
c. Volver el código multi-plataforma (diferentes sistemas operativos).
d. Solo a) y b).
e. Solo b) y c).

V2. Un lenguaje que soporte un sistema débilmente tipado permite:
a. Flexibilizar el estilo de programación.
b. Optimizar el código.
c. Programar de manera más rápida.
d. Solo a) y b).
e. Solo a) y c).