Práctica tipo Examen

Preguntas

1. Conteste en su cuaderno indicando el número de campo en blanco (_____) a rellenar que corresponda, respetando el orden de la numeración. Sino lo respeta recibe 0.

El lenguaje 1)_____ es uno declarativo pero que no permitiría escribir con él una Máquina de Turing. 2) El lenguaje _____ es del paradigma 3)_____ principalmente diseñado para programación numérica y científica y apareció en el año 4)_____. El lenguaje 5)_____ fue especificado por un histórico comité industria-universidad liderado por una mujer venida del campo militar.
Pensado para sustituir el lenguaje ensamblador en programación de sistemas y ser muy portable es 6)____. Su gran popularidad influencia con su sintaxis a muchos lenguajes modernos posteriores. Este lenguaje llamado 7)_____ se inspira en el 8)_____ de Church; aparece en el año 9) ____. Creado bajo este principio de portabilidad 10) _____ (por sus siglas en inglés) y una mayor simpleza en OOP que C++ es el lenguaje 11)_____. La razón de no tener destructores en dicho lenguaje es 12):_____. El lenguaje 13)_____ es considerado el primer lenguaje orientado a objetos. Sabemos que 14)_____ es una técnica de compilación pero que ocurre en tiempo de ejecución a partir de la representación intermedia ejecutable conocida como 15)_____ en `Java`. Dicha representación intermedia es ejecutada por la denominada 16) _____. Los parsers detectan errores de 17)_____ en los programas. Otro tipo de errores son de 18)_____de los cuáles unos se detectan en tiempo de 19)_____ como por ejemplo: 20)_____;y otros errores solamente en tiempo de 21)_____ como por ejemplo: 22)_____. Para poder evaluar una expresión aritmética en `Java` como `x+x+log(y)*z/2+1` esta es usualmente primero transformada en una estructura de 23)_____ Su dibujo es 24) :_____ y de esa estructura se deriva por un recorrido la forma postfija que en este caso resulta ser 25)_____.

2) Terco el Carlitos: no quiere usar operador ternario en `JS` y por eso definió uno propio (`ite`), según él. Pero obtiene el resultado mostrado al correrlo, algo para él inesperado. ¿Por qué se da ese resultado? Sea breve y preciso.

```
let ite = (c, x, y) => c ? x : y;
console.log(ite(false, console.log('Yes'), console.log('No')))
```

```
Yes
No
undefined
```

3) Considere el código adjunto en `samples.js`. Explique la siguiente salida de manera clara y precisa (de una respuesta para cada línea 0), …,5):

```
PP:node samples.js
0) a = { y: 777,
  z: [Function: f],
  w: [Function: w],
  f: [Function: f],
  me: [Function: me],
  self: [Function: self] }
1) a.z                = 7770
2) a.w                = NaN
3) a.f()              = 1200
4) a.me() == a        = true
5) a.self() == this = true
```

4) Considere `comp` la composición de `f` con `g` definida así:

```
const comp = f => g => x => g(f(x))
```

a) ¿Qué función es `h = comp(x => x+x)(x => x*x)`? Expréselo en su respuesta así sin usar comp: `const h = ? => ?` donde `?` representa algo que Ud. debe proveer. Se califica eficiencia.
b) Pruebe formalmente que `a.map(f).map(g) = a.map(comp(f)(g))` donde `a` es un arreglo cualquiera, `f` y `g` son funciones unarias cualesquiera apropiadas (bien definidas). Indique el tiempo de corrida cada lado de la igualdad.

5) Asuma un flujo de trabajo de funciones dadas `a,b,c,d` tales: `a` y `c` no tiene entradas; `b` ocupa lo que retorne de `a`. Y finalmente `d` ocupa los retornos de a, b y c (en ese orden).
a) Escriba una función `main` que implemente ese flujo asincrónicamente usando promesas.

```
function main(a, b, c, d){
 // ...  @return d(a(), b(a()), c())
}
```

b) Lo mismo que a) pero usando `async/await`.

6) Asuma la clase `PPromise` (en módulo `eif400.promise` adjunto). Añada un nuevo método `thenAll` tal que si se le llama ese método a una `PPromise` digamos `pp`, con un arreglo de funciones `[f0,f1,…,fn]` entonces `thenAll` se comportaría así: `pp.thenAll([f0,f1,…,fn])` haría exactamente lo mismo que haría `pp.then(f0).then(f1)….then(fn)`. No use estado mutable. Use combinadores.

Ejemplo: El siguiente código imprime 46

```
PPromise.from(6).thenAll([x=>x*x, x => x + 10])
                .then(x => console.log(x))
```

7) El código adjunto tiene a) Un error de lógica. Explique cuál es. y b) no cumple un estilo FP. Explique por qué. Sea breve y muy preciso en ambas explicaciones. c) Corrija ambos problemas.

```
let testP = start => {
    Promise.resolve()
           .then(_ => start += 1)
           .then(_ => start *= 2)
    return start;
}
console.log("Devil's number is ", testP(332))
```

8) Escriba una función `subArrays(s)` que reciba un arreglo `s` y retorne un arreglo con todos los sub-arreglos de s. Ejemplo: `subArrays([2,1,1])` retornaría `[[], [1], [2], [1,2]]`. No deben aparecer repetidos en su salida. Por ejemplo `[1,2]` es lo mismo que `[2,1]`. El orden no es relevante. Use recursión y combinadores. No use estado mutable.