Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3641 – Lenguajes de Programación I
Septiembre–Diciembre 2024

Examen 4

(20 puntos)

A continuación encontrará 4 preguntas (y una sorpresa al final), cada una de las cuales tiene un valor de 5 puntos. Sea lo más detallado y preciso posible en sus razonamientos y procedimientos.

En algunas preguntas, se usarán las constantes X, Y y Z. Estas constantes debe obtenerlas de los últimos tres números de su carné. Por ejemplo, si su carné es 09-40325, entonces $X = 3$, $Y = 2$ y $Z = 5$.

En aquellas preguntas donde se le pida decir qué imprime un programa, incluya los pasos relevantes de la ejecución del mismo con los cuales usted pudo alcanzar su conclusión.

En aquellas preguntas donde se le pida implementar un programa, mantenga su código en un repositorio `git` remoto (preferiblemente `Github`) y coloque un enlace al mismo en lugar de su respuesta. Todo su código debe ser legible y estar debidamente documentado.

La entrega se realizará por correo electrónico a rmonascal@gmail.com hasta las 11:59pm. VET del Miércoles 11 de Diciembre de 2024.

1. Escoja algún lenguaje de programación de alto nivel, de propósito general y orientado a objetos que **no** comience con alguna de sus iniciales (ya sea de alguno de sus nombres o alguno de sus apellidos).

   (a) De una breve descripción del lenguaje escogido.

      i. Explique la manera de crear y manipular objetos que tiene el lenguaje, incluyendo: constructores, métodos, campos, etc.

      ii. Describa el funcionamiento del manejo de memoria, ya sea explícito (`new/delete`) o implícito (recolector de basura).

      iii. Diga si el lenguaje usa asociación estática o dinámica de métodos y si hay forma de alterar la elección por defecto del lenguaje.

      iv. Describa la jerarquía de tipos, incluyendo mecanismos de herencia múltiple (de haberlos), polimorfismo paramétrico (de tenerlo) y manejo de varianzas.

   (b) Implemente los siguientes programas en el lenguaje escogido:

      i. Defina un interfaz o clase abstracta `Secuencia`, que represente una colección ordenada de elementos. Debe tener los siguientes métodos:

         - **agregar:** Recibe un elemento y lo agrega a la secuencia.
         - **remover:** Devuelve un elemento de la secuencia y lo elimina de la misma. Arroja un error si está vacía.
         - **vacio:** Dice si la secuencia está vacía (no tiene elementos).

         Defina dos clases concretas `Pila` y `Cola` que sean subtipo de `Secuencia`

         - Para `Pila` los elementos se manejan de tal forma que el *último* en ser agregado es el *primero* en ser removido.
         - Para `Cola` los elementos se manejan de tal forma que el *primero* en ser agregado es el *primero* en ser removido.

      ii. Defina un tipo de datos que represente grafos como listas de adyacencias y cada nodo sea representado por un número entero (puede usar todas las librerías a su disposición en el lenguaje).
      Además, defina una clase abstracta `Busqueda` que debe tener un método **buscar**. Este método debe recibir dos enteros: `D` y `H`, y debe devolver la cantidad de nodos explorados, partiendo desde el nodo `D` hasta llegar al nodo `H`. En caso de que `H` no sea alcanzable desde `D`, debe devolver el valor -1 (menos uno).
      Esta clase debe estar parcialmente implementada, dejando solamente abstraído el orden en el que se han de explorar los nodos.
      Defina dos clases concretas `DFS` y `BFS` que sean subtipo de `Busqueda`.

         - Para `DFS` el orden de selección de nodos es a profundidad (usando un pila).
         - Para `BFS` el orden de selección de nodos es a amplitud (usando un cola).

2. Tomando como referencia las constantes X, Y y Z planteadas en los párrafos de introducción del examen, considere las siguientes definiciones de clases, escritas en pseudo–código:

```
class Bebida {
   int a = X, b = Y

   fun s(int x): int {
      a = b + x
      return t(a)
   }

   fun t(int y): int {
      return b * y + a
   }
}
```

```
class Cafe extends Bebida {
   Bebida caliente = new Marron()

   fun t(int y): int {
      return caliente.s(a + b) + y
   }
}
```

```
class Marron extends Cafe {
   int c = Z

   fun s(int x): int {
      a = c + x - 2
      c = a + b * x
      return t(a * b + c)
   }

   fun t(int y): int {
      return c - y
   }
}
```

Considere además el siguiente fragmento de código:

```
Bebida e = new Cafe()
Bebida pres = new Marron()
Cafe o = new Marron()

print(e.s(1) + pres.s(1) + o.s(1))
```

Diga qué imprime el programa en cuestión si el lenguaje tiene:

(a) Asociación estática de métodos

(b) Asociación dinámica de métodos

Recuerde mostrar paso a paso el estado del programa (la pila de ejecución).

3. Se desea que modele e implemente, en el lenguaje de su elección, un manejador de tablas de métodos virtuales para un sistema orientado a objetos con herencia simple y despacho dinámico de métodos:

   (a) Debe saber tratar con definiciones de clases, potencialmente con herencia simple. Estas definiciones tendrán únicamente los nombres de los métodos que poseerá.

   (b) Una vez iniciado el programa, pedirá repetidamente al usuario una acción para proceder. Tal acción puede ser:

      i. `CLASS <tipo> [<nombre>]`
      Define un nuevo tipo que poseerá métodos con nombres establecidos en la lista proporcionada. El `<tipo>` puede ser:
      - Un nombre, que establece un tipo que no hereda de ningún otro.
      - Una expresión de la forma `<nombre> : <super>`, que establece el nombre del tipo y el hecho de que este tipo hereda del tipo con nombre `<super>`.

      Por ejemplo: `CLASS A f g` y `CLASS B : A f h`

      Notemos que es posible reemplazar definiciones de una super clase en clases que la heredan.
      El programa debe reportar un error e ignorar la acción si el nombre de la nueva clase ya existe, si la clase `super` no existe, si hay definiciones repetidas en la lista de nombres de métodos o si se genera un ciclo en la jerarquía de herencia.

      ii. `DESCRIBIR <nombre>`
      Debe mostrar la tabla de métodos virtuales para el tipo con el nombre propuesto.

      Por ejemplo: `DESCRIBIR B`

      Esto debe mostrar:

      ```
      f -> B :: f
      g -> A :: g
      h -> B :: h
      ```

      El programa debe reportar un error e ignorar la acción si el nombre del tipo no existe.

      iii. `SALIR`
      Debe salir del simulador.

   Al finalizar la ejecución de cada acción, el programa deberá pedir la siguiente acción al usuario.

   Investigue herramientas para pruebas unitarias y cobertura en su lenguaje escogido y agregue pruebas a su programa que permitan corroborar su correcto funcionamiento. Como regla general, su programa debería tener una cobertura (de líneas de código y de bifuración) mayor al 80%.

4. Considere las funciones `foldr` y `const`, escritas en un lenguaje muy similar a Haskell:

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr _ e []     = e
foldr f e (x:xs) = f x $ foldr f e xs

const :: a -> b -> a
const x _ = x
```

Considere también la siguiente función que aplica una función solamente sobre la cola de una lista y agrupa la cabeza con otro valor dado:

```
what :: a -> ([b] -> [(a, b)]) -> [b] -> [(a, b)]
what _ _ []     = []
what x f (y:ys) = (x, y) : f ys
```

(a) Considere la siguiente implementación de una función `misteriosa`, usando `foldr`:

```
misteriosa :: ???
misteriosa = foldr what (const [])
```

Considere también la siguiente función, que genera una lista de números enteros a partir de un cierto valor inicial:

```
gen :: Int -> [Int]
gen n = n : gen (n + 1)
```

Muestre la evaluación, paso a paso, de la expresión `misteriosa "abc" (gen 1)`, considerando que:

i. El lenguaje tiene orden de evaluación *normal*.
ii. El lenguaje tiene orden de evaluación *aplicativo*.

Si sospecha que en algún momento uno de estos programas puede caer en una evaluación recursiva infinita, realice las primeras expansiones, detenga la evaluación y argumente las razones por las que cree que dicha evaluación no terminaría.

(b) Considere el siguiente tipo de datos que representa árboles binarios con información en las ramas:

```
data Arbol a = Hoja | Rama a (Arbol a) (Arbol a)
```

Construya una función `foldA` (junto con su firma) que permita reducir un valor de tipo `(Arbol a)` a algún tipo `b` (de forma análoga a `foldr`). Su implementación debe poder tratar con estructuras potencialmente infinitas.

Su función debe cumplir con la siguiente firma:

```
foldA :: (a -> b -> b -> b) -> b -> Arbol a -> b
```

(c) Considere una versión de la función `what` que funciona sobre árboles (aplica la función proporcionada a ambos sub–árboles) y llamésmola *what tree function*:

```
whatTF :: a
       -> (Arbol b -> Arbol (a, b))
       -> (Arbol b -> Arbol (a, b))
       -> Arbol b
       -> Arbol (a, b)
whatTF _ _ _ Hoja         = Hoja
whatTF x f g (Rama y i d) = Rama (x, y) (f i) (g d)
```

Usando su función `foldA` definimos la función `sospechosa`:

```
sospechosa :: ???
sospechosa = foldA whatTF (const Hoja)
```

Definimos también la siguiente función, que genera un árbol de números enteros a partir de un cierto valor inicial:

```
genA :: Int -> Arbol Int
genA n = Rama n (genA (n + 1)) (genA (n * 2))
```

Finalmente, definimos el valor `arbolito` como una instancia de `Arbol Char`:

```
arbolito :: Arbol Char
arbolito = Rama 'a' (Rama 'b' Hoja (Rama 'c' Hoja Hoja)) Hoja
```

Muestre la evaluación, paso a paso, de la expresión `sospechosa arbolito (genA 1)`, considerando que:

i. El lenguaje tiene orden de evaluación *normal*.

ii. El lenguaje tiene orden de evaluación *aplicativo*.

Si sospecha que en algún momento uno de estos programas puede caer en una evaluación recursiva infinita, realice las primeras expansiones, detenga la evaluación y argumente las razones por las que cree que dicha evaluación no terminaría.

5. **RETO EXTRA: *¡MEME LORD!***

   En esta ocasión, el reto es sencillo:

   Crea uno o más memes que tenan que ver con la materia o las clases.

   Todo el que entregue un meme que cumpla con los requerimientos (que tenga que ver con la materia o las clases) tendrá dos (2) puntos extra.

   Nota 1: Son 2 puntos extra en total, ***no*** *2 puntos extra por meme.*

   Nota 2: Díganme si tengo permiso de compartir sus memes, de forma anónima, en el grupo de telegram y/o redes sociales.