Taller 2 - Semántica de lenguajes de programación Fundamentos de Lenguajes Programación

Carlos Andres Delgado S, Msc
carlos.andres.delgado@correounivalle.edu.co

Octubre de 2021

Importante: El no cumplimiento de alguna de las normas aquí expuestas le traerá reducción en la nota o la anulación de su taller.

1. El taller debe ser entregado antes del día **Domingo, 24 de Octubre de 2021 a las 23:59:59** hora de Colombia del por el enlace dispuesto en el campus virtual. Se permiten entregas tardías, pero se descuenta 0.15 en la nota por hora o fracción de retraso. Por ejemplo, si entrega a partir delas 12:00:01 am se aplicará una penalización de 0.15, si lo entrega a partir de las 01:00:01 se aplicará 0.3 y así sucesivamente.
2. Entregue un sólo archivo comprimido. No entregue archivos comprimidos dentro de archivos comprimidos.
3. Debe entregar el código fuente organizado en carpetas dentro del primer nivel del archivo comprimido, no cree una jerarquía compleja difícil de revisar.
4. No se permite copiar código de Internet ni de sus compañeros. Si se encuentra código copiado el taller será anulado por completo a todas las partes involucradas.
5. Debe implementar el taller en Dr Racket y recuerde que debe usar lambda para especificar sus funciones.
6. El taller puede ser realizado por grupos de máximo 4 personas previa inscripción al campus virtual. Es requisito inscribirse a los grupos del campus para realizar la entrega.
7. Las primeras líneas de cada archivo deben contener los nombres y códigos de los estudiantes.

```
;Autores: Juanito Perez, 1902321. Pepita Gomez, 1954545, Juanita Delgado, 1914547
```

8. En ese mismo archivo, vendrán comentados los procedimientos que llevan al código de la declaración, las operaciones, la expresión BNF de las estructuras que se están utilizando, y algunos ejemplos de prueba. Por ejemplo, si se pidiera construir el procedimiento remove-first, deberá existir un código como:

```
;; <lista-de-simbolos> := ({<exp-simbolo>}*)
;; <exp-simbolo> := <simbolo> | <lista-de-simbolos>
;;
;; remove-first : simbolo * lista-de-simbolos -> lista-de-simbolos
;;
;; Proposito:
;; Procedimiento que remueve la primera ocurrencia de un simbolo
;; en una lista de simbolos.
;;

(define remove-first
   (lambda (s los)
      (if (null? los)
          '()
          (if (eqv? (car los) s)
              (cdr los)
              (cons (car los)
                    (remove-first s (cdr los)))))))

;; Pruebas
(remove-first 'a '(a b c))
(remove-first 'b '(e f g))
(remove-first 'a4 '(c1 a4 c1 a4))
(remove-first 'x '())
```

9. Para la evaluación de los puntos tenga en cuenta los ejemplos de la última sección.

1. Enunciado

Entregue un archivo por cada representación:

- Representación de los tablas hash de números usando listas
- Representación de los tablas hash de números usando procedimientos
- Representación de los tablas hash de números usando datatypes

Las tablas hash de números son una estructura de datos que permite indexar por una llave, por ejemplo

```
f = {a:'(1 2 3), b:'(1), c:'(1,2)}
```

f es una tabla hash de números, cuya llave a contiene $[1, 2, 3]$, su llave b contiene $[2]$ y su llave c contiene la lista $[1, 2]$. Para implementar los tablas hash de números usamos la siguiente gramática:

```
<tabla-hash> ::= '()
                  th-vacio
             ::= <item> <tabla-hash>
                 th-novacio(item,reg)

<item>  ::= <simbolo> <lnumero>
            itemElm(key,dato)


<lnumero> ::= '()
              lnumvacia
           :: <numero> <lnumero>
              lnumnovacia(num,lst)
```

Simbolo es cualquier **palabra** y numero es cualquier número. La información que se encuentra en el recuadro son los nombres que deben llevar los constructores y las variantes en el datatype.

Así mismo, si se desea implementar un extractor su nombre está compuesto por el nombre de la variante y el nombre del campo, por ejemplo:

- th-novacio->item
- lnumero->num

Los predicados usan el nombre del constructor, pero al final agregan un ?, por ejemplo lnumnovacia?, lnumvacia?.

Le objetivo del taller es implementar los tablas hash de números usando representación basada en listas, procedimientos y datatypes.

1.1. Representación basada en listas

Debe implementar los constructores y observadores (predicados y extractores) respetando los nombres indicados en la gramática usando una representación basada en listas.

Diseñe las siguientes funciones en su representación:

1. buscar-llave: Retorna el valor de acuerdo a una llave (símbolo) que se desee buscar.
2. sumar-valores: Recibe una tabla y retorna una lista de listas, donde su primera posición es la llave y su segunda posición es la suma. Para el caso de f debe retornarse $'((a,6)(b,1)(c,3))$

Si ha implementado correctamente estas funciones, para el caso de procedimientos solo necesita copiarlas y pegarlas.

1.2. Representación basada en procedimientos

Debe implementar los constructores y observadores (predicados y extractores) respetando los nombres indicados en la gramática usando una representación basada en procedimientos.

Diseñe las siguientes funciones en su representación:

1. buscar-llave: Retorna el valor de acuerdo a una llave (símbolo) que se desee buscar.
2. sumar-valores: Recibe una tabla y retorna una lista de listas, donde su primera posición es la llave y su segunda posición es la suma. Para el caso de f debe retornarse $'((a,6)(b,1)(c,3))$

Si ha realizado correctamente el taller, aquí sólo debería copiar y pegar estas funciones solicitadas, realizadas en la implementación basada en listas.

1.3. Representación usando Datatypes

Debe implementar los constructores y observadores (predicados y extractores) respetando los nombres indicados en la gramática usando define-datatype.

Diseñe las siguientes funciones en su representación:

1. buscar-llave: Retorna el valor de acuerdo a una llave (símbolo) que se desee buscar.
2. sumar-valores: Recibe una tabla y retorna una lista de listas, donde su primera posición es la llave y su segunda posición es la suma. Para el caso de f debe retornarse $'((a,6)(b,1)(c,3))$

En este caso debe usar **cases** para el desarrollo de la función.

2. Rúbricas de evaluación

Las rubricas que se van a utilizar para la evaluación consideran aspectos relacionados con las reglas del taller, calidad del informe y de realización de las implementaciones. En cada una se explica la asignación de puntos.

3. Sobre las reglas

Criterio	**Nivel 0** (0 pts)	**Nivel 1** (3 pts)	**Nivel 2** (5 pts)
Nombres y códigos presentes al inicio de cada archivo	No se cumple con esta regla	No están en todos los archivos que entrega o alguno de los estudiantes no incluye su código	Se encuentran en todos los archivos de la entrega con nombres y códigos
Funciones comentadas de acuerdo al enunciado	No se realizan comentarios	No se comentan todas las funciones realizadas o los comentarios en algunas de ellas no están de acuerdo al enunciado	Se comentan todas las funciones del código de acuerdo al enunciado
Organización de la entrega	La entrega incluye archivos comprimidos dentro del comprimido principal	Se entrega un archivo comprimido, pero los nombres de los archivos no indican claramente a qué punto se refieren	Se entrega un archivo comprimido y los nombres de los archivos de cada punto son claros
Archivos de implementación	No se cumple la regla de que cada implementación esté realizada en un archivo, es decir, 1 archivo para listas, 1 archivo para procedimientos y 1 archivo para datatypes	Se entregan 3 archivos y cada uno corresponde a una implementación, sin embargo, sus nombres no son claros	Se entregan 3 archivos, cada uno por una implementación de tablas hash de números (listas, procedimientos y datatypes) y sus nombres son claros con respecto a lo que contiene. Ejemplo: Representación-listas.rkt, Representación-procedimientos.rkt, Representación-datatypes.rkt

Total puntos 20

4. Sobre el taller

Los ejemplos no deben ir comentados y deben estar en el código entregado para ser válidos.

Para que los tipos de datos sean correctos debe implementarse:

1. tabla-hash
2. item
3. lnumero

Criterio	**Nivel 0** (0 pts)	**Nivel 1** (5 pts)	**Nivel 2** (10 pts)	**Nivel 3** (15 pts)
Diseño de representación con listas	No realiza las funciones o estas no se pueden ejecutar	Realiza los constructores y los predicados, sin embargo, hacen falta uno o más tipos de datos	Realiza las funciones correctamente, pero no incluye ejemplos de su funcionamiento o no usa los nombres indicados en la gramática	Realiza las funciones correctamente, respeta sus nombres e incluye al menos 5 ejemplos de construcción de tablas hash de números y uso de los observadores
Diseño de representación con procedimientos	No realiza las funciones o estas no se pueden ejecutar	Realiza los constructores y los predicados, sin embargo, hacen falta uno o más tipos de datos	Realiza las funciones correctamente, pero no incluye ejemplos de su funcionamiento o no usa los nombres indicados en la gramática	Realiza las funciones correctamente, respeta sus nombres e incluye al menos 5 ejemplos de construcción de tablas hash de números y uso de los observadores
Diseño de funciones solicitadas	No realiza las funciones o entrega un código que no funciona	Realiza correctamente 1 función, o bien alguna de las funciones no trabaja correctamente. También aplica si el estudiante no usa los extractores y predicados diseñados para la función	Realiza correctamente las 3 funciones, pero no incluye ejemplos de su funcionamiento	Realiza correctamente las funciones correctamente e incluye al menos 3 ejemplos de funcionamiento de cada una

Criterio	**Nivel 0** (0 pts)	**Nivel 1** (5 pts)	**Nivel 2** (10 pts)	**Nivel 3** (15 pts)
Unicidad de las funciones solicitadas	No realiza las funciones o entrega un código que no funciona o bien las funciones realizadas no son las mismas para ambas representaciones	Una función trabaja correctamente y es la misma para ambas representaciones	Dos funciones trabajan correctamente y son las mismas para ambas representaciones	Las tres funciones trabajan correctamente y son las mismas para ambas representaciones
Diseño de representación con datatypes	No realiza las funciones o estas no se pueden ejecutar	Implementa los datatypes, pero no están todos los tipos de datos necesarios o no se considera el tipo de cada uno de los campos de las variantes	Realiza los datatypes correctamente, pero no incluye ejemplos de su funcionamiento o no usa los nombres solicitados en la gramática	Realiza los datatypes correctamente, incluye 5 ejemplos de construcción de datos usando las funciones que proveen los datatypes y usa los nombres solicitados en la gramática
Diseño de funciones buscar-llave, buscar-listas y buscar-nolistas con cases	No realiza las funciones o estas no se pueden ejecutar	Realiza la función usando los cases, pero la extracción de los datos no considera el tipo de cada uno de los campos	Realiza 1 función correctamente o no incluye ejemplos de cada una de las funciones	Realiza las 2 funciones correctamente e incluye al menos 3 ejemplos de funcionamiento de cada una

Total 90 puntos.

Total del taller 110 puntos que equivalen a 5.0, la formula para obtener su nota es:

$$\text{Nota taller} = 5{,}0 * \frac{\text{puntos obtenidos}}{110}$$

Sobre rúbricas

Su entrega será valorada a partir de las rúbricas consignadas previamente; cada una contiene una lista de criterios y niveles de desempeño. Para obtener la máxima calificación posible de su taller, debe cumplir las especificaciones del nivel más alto para cada criterio.

La nota se calculará con la suma de los puntos asignados al nivel que usted logre alcanzar para cada caso.