Lenguajes de Programación, 2022-1
Práctica 1

Facultad de Ciencias, UNAM

Profesora: Karla Ramírez Pulido
Ayud. Lab.: Manuel Ignacio Castillo López

Fecha de inicio: Miércoles 17 de agosto de 2022
Fecha de entrega: Miércoles 31 de agosto 2022

Descripción

La práctica consiste en implementar las funciones aquí descritas en un módulo de Racket (archivo *.rkt). Puede definir todas las funciones auxiliares que considere necesarias (salvo que se indique explícitamente lo contrario). Ponga atención a las restricciones y descripción de cada ejercicio.

Cada definición en Racket debe incluir un breve comentario que describa su propósito; en particular las precondiciones y poscondiciones para las que fue diseñada.

Puede usar el material de clase, laboratorio, manuales y guías en línea o apoyarse con el ayudante para resolver los ejercicios. Puede realizarla de manera individual o en equipos de 2 o 3 integrantes; se les recomienda desarrollar la práctica en equipo. Pueden solicitarnos ayuda para encontrar pareja/equipo.

Ejercicios

1. (1 pto). Defina una función que filtre el contenido de una lista dado un predicado.

Un predicado es una función que devuelve un valor booleano (`#t` ó `#f`).

- Precondiciones: Una lista heterogénea y un predicado de un único argumento.
- Poscondiciones: Una lista cuyos elementos satisfacen el predicado.

```
;; filtra-lista: (listof any) procedure → (listof any)
```

Ejemplos:

```
> (filtra-lista (list 1 "hola" 2 (list 1 2) 3 'a 'b) number?)
'(1 2 3)
>
> (filtra-lista (list 1 2 "hola" #\a 3 4 empty "adiós") string?)
'("hola" "adiós")
```

2. (1 pto). Defina una función que tome una lista y devuelva otra del mismo tamaño, cuyos elementos describen en texto el tipo de datos de los elementos en la lista original.

- Precondiciones: Una lista heterogénea.
- Poscondiciones: Una lista de cadenas que indican el tipo de datos de cada elemento en la lista dada. Los tipos de datos a describir son: `boolean`, `number`, `char`, `string`, `symbol`, `keyword`, `pair` y `list`. En caso de que la lista contenga un elemento con un tipo que no es alguno de los anteriores; debe indicar que el tipo es `otro`.

```
;; tipos-lista: (listof any) → (listof string)
```

Ejemplos:

```
> (tipos-lista (list 1 "hola" 2 (list 1 2) 3 'a 'b))
'("number" "string" "number" "list" "number" "symbol" "symbol")
>
> (tipos-lista (list 1 2 "hola" #\a 3 4 empty "adiós"))
'("number" "number" "string" "char" "number" "number" "list" "string")
```

3. (1 pto). Se dice que un número natural es raro si al sumar cada una de sus cifras elevadas al número de cifras que lo forman, se obtiene el número original. Por ejemplo, el número 153 (que tiene 3 cifras) es raro, pues $153 = 1^3 + 5^3 + 3^3$.

Defina un predicado que dado un número natural indique si es raro.

- Precondiciones: Un número natural.
- Poscondiciones: Un booleano cuyo valor de verdad indica si el número dado es raro (`#t`) o no lo es (`#f`).

```
;; raro?: number → boolean
```

Ejemplos:

```
> (raro? 12)
#f
>
> (raro? 153)
#t
```

4. (1 pto). Defina un predicado que dado un número arbitrario de números, indique si están ordenados en forma descendente o no.

- Precondiciones: Una serie de números; potencialmente vacía.
- Poscondiciones: Un booleano cuyo valor de verdad indica si la serie numérica dada tiene un orden descendente (`#t`) o no (`#f`).

```
;; descendente?: number* → boolean
```

Ejemplos:

```
> (descendente? 5 4 6 2 1)
#f
>
> (descendente? 5 4 3 2 1)
#t
```

5. (1 pto). Defina un predicado que tome una cadena e indique si esta es un palíndromo. Un palíndromo es una cadena que se escribe igual que invirtiendo el orden de sus caracteres.

- Precondiciones: Una cadena; potencialmente vacía.
- Poscondiciones: Un booleano cuyo valor de verdad indica si la cadena dada es un palíndromo (`#t`) o no lo es (`#f`).

```
;; palindromo?: string → boolean
```

Ejemplos:

```
> (palindromo? "texto")
#f
>
> (palindromo? "girafarig")
#t
```

6. (1 pto). Defina un predicado que indica si un número entero es primo. Un número primo es aquel que solo es divisible por 1 y sí mismo (divisible a su vez significa que al dividir el número por otro se obtiene cero como residuo; la división es exacta).

- Precondiciones: Un número entero.
- Poscondiciones: Un booleano cuyo valor de verdad indica si el número es primo (`#t`) o no (`#f`).

```
;; primo?: number → boolean
```

Ejemplos:

```
> (primo? 8)
#f
>
> (primo? 11)
#t
```

7. (1 pto). Defina una función que indique todas las posibles formas de obtener una cantidad entera positiva a partir de monedas de $1, $2 y $5 sin repeticiones.

Se considera una combinación repetida aquella en la que únicamente cambian de lugar las monedas. Por ejemplo, para obtener 4; podríamos tener las combinaciones ($1, $2, $1) y ($2, $1, $1); pero en ambos casos tenemos una moneda de $2 y dos de $1; por lo que solo están cambiando de lugar y las consideraremos repetidas.

- Precondiciones: Un número entero positivo.

- Poscondiciones: Un número entero positivo que indica las distintas formas de combinar monedas de $1, $2 y $5 (sin repeticiones) para obtener la cantidad dada.

```
;; num-comb-monedas: number → number
```

Ejemplos:

```
> (num-comb-monedas 7)
6
>
> (num-comb-monedas 10)
10
```

8. (1 pto). Define una función que tome una lista de números y devuelva el promedio, moda y mediana de los elementos en la lista.

El promedio es la suma de todos los elementos dividida por el total de elementos en la lista.

La moda es el conjunto de valores que más se repite en la lista.

Por último, la mediana es el valor en la posición central de la lista; es decir, en listas de tamaño impar, es el elemento cuya posición corresponde la mitad del tamaño de la lista y en listas de tamaño par, es el promedio de los valores cuya posición corresponde con el piso y techo de la mitad del tamaño de la lista.

- Precondiciones: Una lista numérica.
- Poscondiciones: Un primer número que representa el promedio de los elementos en la lista, un segundo número que representa la moda y un último número que representa la mediana.

```
;; prom-mod-med: (listof number) → number (listof number) number
```

Ejemplos:

```
> (prom-mod-med (list 10 7 4 6 8 10 10 9))
8
'(10)
7
>
> (prom-mod-med (list 2 8 7 3 5))
5
'(2 8 7 3 5)
7
```

9. (1 pto). Defina una función que genere en una lista, todas las posibles rotaciones de elementos de una lista dada.

- Precondiciones: Una lista.
- Poscondiciones: Una lista con todas las posibles rotaciones de los elementos de la lista original.

```
;; rota: (listof any) → (listof (listof any))
```

Ejemplos:

```
> (rota (list 1 2 3))
'('(1 2 3) '(2 3 1) '(3 1 2))
>
> (rota (list "hola" #f 5))
'('("hola" #f 5) '(#f 5 "hola") '(5 "hola" #f))
```

10. (1 pto). Defina una función que tome una lista de enteros y determine si representa el inicio de una secesión geométrica.

Los elementos en una sucesión geométrica tienen la forma $a_i = a_1 * r^{i-1}$. Si en efecto la lista dada representa una sucesión cuyos elementos satisfacen esta regla, devuelve una lista de cadenas del mismo tamaño que la original en la que cada entrada representa los elementos de la lista original en su forma $a_i = a_1 * r^{i-1}$. En otro caso, devuelve la lista vacía.

Considere que la razón de una sucesión geométrica se obtiene como $r = \frac{a_n}{a_{n-1}}$ y que a_1 es el primer elemento en la lista dada.

- Precondiciones: Una lista numérica.
- Poscondiciones: Una lista de cadenas; cuyo cada elemento representa los elementos de la lista original en su forma $a_i = a_1 * r^{i-1}$ en caso de que sea represente el inicio de una sucesión geométrica, en otro caso devuelve la lista vacía.

```
;; extender-suc-geom: (listof number) → (listof string)
```

Ejemplos:

```
> (extender-suc-geom (list 1 2 4 8 16 32))
'("1*2^(1-1)" "1*2^(2-1)" "1*2^(3-1)" "1*2^(4-1)" "1*2^(5-1)" "1*2^(6-1)")
>
> (extender-suc-geom (list 5 10 20 40 80))
'("5*2^(1-1)" "5*2^(2-1)" "5*2^(3-1)" "5*2^(4-1)" "5*2^(5-1)")
```

11. (1 pto. Extra) Defina funciones que realicen una prueba unitaria de cada uno de los ejercicios en la práctica; es decir, para obtener el punto extra deberá definir 10 funciones de prueba. Utilice la función incluida en el lenguaje `plai` `test` para realizarlo.

Ejemplos:

```
> (prueba-filtra-lista)
good (filtra-lista (list 1 2 "hola" #\a 3 4 empty "adiós") string?) at line 146
        expected: '("hola" "adiós")
        given: '("hola" "adiós")
>
> (prueba-num-comb-monedas)
good (num-comb-monedas 7) at line 152
        expected: 6
        given: 6
```

Requerimientos

Deberán entregar su práctica a través de la plataforma **Google classroom** antes que termine el día 31 de agosto; tal y como lo indican los lineamientos de entrega. Deberán adjuntar su código en un archivo llamado **practica01.rkt**. Revisen bien mayúsculas, minúsculas y espacios para el nombre de archivo que se pide. La liga del classroom es la siguiente: `https://classroom.google.com/c/NTI1OTc5ODU1NzI5?cjc=2nbnbrh`

Deberán indicar en los comentarios solicitados para sus funciones, el número del ejercicio que resuelven y su descripción debe resumir lo que solicita el ejercicio. Las funciones auxiliares deben esclarecer en sus comentarios que son funciones auxiliares y en qué ejercicios se utilizan.

Recuerden que estamos atentos a cualquier duda, ¡suerte!