Tecnológico de Monterrey

Evidencia #2: Implementación de un simulador de almacén automatizado

Diego Andrés Figueroa Peart
A01660987

Curso:
Implementación de métodos computacionales
(TC2037.820)

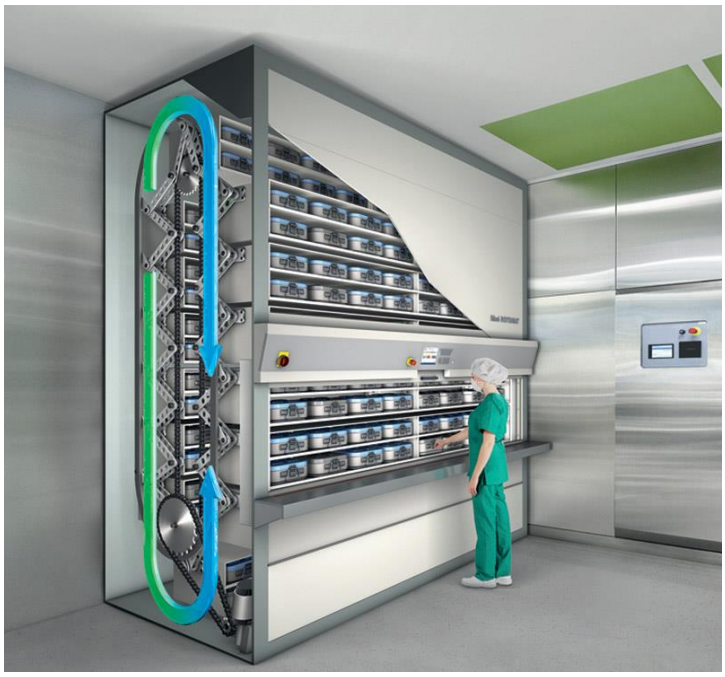
Profesores:
Román Martínez Martínez
Alberto Oliart Ros
Valentina Narváez Terán
Germán Domínguez Solís

Fecha de entrega:
7 de mayo de 2023

Evidencia #2: Carrusel vertical

La situación problema de este segundo periodo consistió en programar la simulación de un carrusel vertical de almacenamiento en el lenguaje de programación Racket, del paradigma funcional. Racket es un dialecto moderno del lenguaje Lisp, y es derivado de Scheme.

Un carrusel vertical de almacenamiento consiste en la siguiente estructura:



Este medio de almacenar productos pequeños tiene la ventaja de ser compacto y ocupar poco espacio de suelo para la cantidad de productos que puede almacenar, mientras que todos los productos permanecen fácilmente accesibles para el usuario.

Diseño de las estructuras de datos

Para simular esta estructura dentro de Racket, utilizaré listas anidadas que contengan el nombre del producto, su cantidad y su precio unitario, en el siguiente formato:

(((producto 1 cantidad precio) (producto 2 cantidad precio) (producto 3 cantidad precio))
((producto 4 cantidad precio) (producto 5 cantidad precio) (producto 6 cantidad precio))
((producto 7 cantidad precio) (producto 8 cantidad precio) (producto 9 cantidad precio)))

El tamaño del carrusel será configurable y dinámico; el código se adaptará sin importar el número de filas y columnas, lo único imperativo es que no varíe el número de columnas en cada fila y el número de filas en cada columna.

El carrusel no tendrá menos de 20 filas y 5 columnas, por lo tanto, el carrusel que generé para realizar pruebas tiene este tamaño, y contiene 100 productos medicinales:

```
((("Paracetamol" 99 1.9) ("Ibuprofeno" 193 4.0) ("Aspirina" 101 5.2) ("Naproxeno" 50 2.3) ("Diazepam" 113 5.6))
(("Ketorolaco" 55 6.6) ("Lorazepam" 66 5.6) ("Alprazolam" 126 2.7) ("Clonazepam" 140 7.6) ("Sertralina" 188 1.9))
(("Fluoxetina" 168 5.5) ("Paroxetina" 100 1.7) ("Citalopram" 93 7.9) ("Amitriptilina" 51 6.6) ("Venlafaxina" 46 9.9))
(("Escitalopram" 12 6.7) ("Duloxetina" 191 1.6) ("Pregabalina" 83 2.3) ("Gabapentina" 164 3.7) ("Carbamazepina" 197 6.5))
(("Valproato" 187 4.2) ("Lamotrigina" 52 6.7) ("Levotiroxina" 85 1.6) ("Metformina" 143 4.8) ("Insulina glargina" 54 8.9))
(("Insulina lispro" 83 4.1) ("Insulina aspart" 170 9.2) ("Insulina detemir" 191 9.9) ("Gliclazida" 38 1.1) ("Glimepirida" 160
4.2)) (("Rosuvastatina" 60 7.5) ("Atorvastatina" 7 8.4) ("Simvastatina" 75 4.6) ("Ezetimiba" 102 7.9) ("Fenofibrato" 41 6.3))
(("Omeprazol" 148 3.9) ("Lansoprazol" 161 5.8) ("Pantoprazol" 156 4.4) ("Esomeprazol" 31 1.8) ("Ranitidina" 115 3.8))
(("Famotidina" 91 7.1) ("Metoclopramida" 79 2.7) ("Ondansetrón" 127 6.6) ("Domperidona" 144 3.6) ("Cetirizina" 104 8.6))
(("Loratadina" 93 1.1) ("Fexofenadina" 61 4.5) ("Clorfenamina" 150 6.5) ("Desloratadina" 153 9.1) ("Pseudoefedrina" 68 6.4))
(("Oximetazolina" 28 7.8) ("Salbutamol" 59 8.6) ("Albuterol" 118 7.0) ("Budesonida" 185 3.2) ("Fluticasona" 176 8.5))
(("Montelukast" 148 3.3) ("Cromoglicato de sodio" 148 5.1) ("Ketotifeno" 14 4.0) ("Amoxicilina" 32 1.5) ("Azitromicina" 141
8.3)) (("Cefalexina" 32 5.5) ("Ceftriaxona" 116 2.9) ("Ciprofloxacino" 56 6.4) ("Doxiciclina" 172 1.6) ("Metronidazol" 141
9.9)) (("Trimetoprima" 92 6.1) ("Aciclovir" 31 8.2) ("Valaciclovir" 167 8.5) ("Oseltamivir" 77 7.2) ("Rituximab" 76 9.9))
(("Infliximab" 38 8.2) ("Adalimumab" 198 8.5) ("Etanercept" 41 8.5) ("Interferón beta-1a" 61 8.3) ("Interferón beta-1b" 120
6.1)) (("Natalizumab" 97 2.0) ("Alemtuzumab" 113 1.6) ("Bevacizumab" 55 2.8) ("Trastuzumab" 12 6.8) ("Cisplatino" 144 7.8))
(("Carboplatino" 130 9.3) ("Paclitaxel" 141 2.7) ("Docetaxel" 49 8.1) ("Irinotecán" 87 1.0) ("5-Fluorouracilo" 106 4.7))
(("Doxorrubicina" 33 2.4) ("Ciclofosfamida" 53 9.4) ("Vincristina" 83 1.3) ("Vinblastina" 112 6.5) ("Bleomicina" 91 3.6))
(("Procarbazina" 64 5.6) ("Dacarbazina" 140 6.1) ("Tamoxifeno" 135 3.5) ("Letrozol" 165 9.6) ("Anastrozol" 59 6.6))
(("Fulvestrant" 174 8.8) ("Leuprorelina" 125 1.5) ("Goserelina" 84 3.3) ("Degarelix" 49 7.5) ("Abiraterona" 71 2.6)))
```

Este será almacenado dentro de un archivo de texto llamado “carrusel.txt”, el cuál deberá estar colocado dentro del mismo directorio que el archivo de Racket. Después de la lista que representa al carrusel, este archivo de texto contendrá un elemento más que representará el producto que actualmente estará mostrando el carrusel.

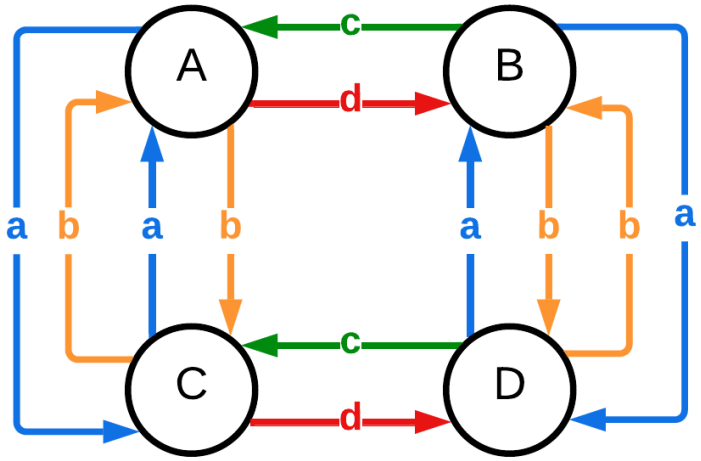
```
((("Paracetamol" 99 1.9) ("Ibuprofeno" 193 4.0) ("Aspirina" 101 5.2) ("Naproxeno" 50 2.3) ("Diazepam" 113 5.6))
(("Ketorolaco" 55 6.6) ("Lorazepam" 66 5.6) ("Alprazolam" 126 2.7) ("Clonazepam" 140 7.6) ("Sertralina" 188 1.9))
(("Fluoxetina" 168 5.5) ("Paroxetina" 100 1.7) ("Citalopram" 93 7.9) ("Amitriptilina" 51 6.6) ("Venlafaxina" 46 9.9))
(("Escitalopram" 12 6.7) ("Duloxetina" 191 1.6) ("Pregabalina" 83 2.3) ("Gabapentina" 164 3.7) ("Carbamazepina" 197 6.5))
(("Valproato" 187 4.2) ("Lamotrigina" 52 6.7) ("Levotiroxina" 85 1.6) ("Metformina" 143 4.8) ("Insulina glargina" 54 8.9))
(("Insulina lispro" 83 4.1) ("Insulina aspart" 170 9.2) ("Insulina detemir" 191 9.9) ("Gliclazida" 38 1.1) ("Glimepirida" 160
4.2)) (("Rosuvastatina" 60 7.5) ("Atorvastatina" 7 8.4) ("Simvastatina" 75 4.6) ("Ezetimiba" 102 7.9) ("Fenofibrato" 41 6.3))
(("Omeprazol" 148 3.9) ("Lansoprazol" 161 5.8) ("Pantoprazol" 156 4.4) ("Esomeprazol" 31 1.8) ("Ranitidina" 115 3.8))
(("Famotidina" 91 7.1) ("Metoclopramida" 79 2.7) ("Ondansetrón" 127 6.6) ("Domperidona" 144 3.6) ("Cetirizina" 104 8.6))
(("Loratadina" 93 1.1) ("Fexofenadina" 61 4.5) ("Clorfenamina" 150 6.5) ("Desloratadina" 153 9.1) ("Pseudoefedrina" 68 6.4))
(("Oximetazolina" 28 7.8) ("Salbutamol" 59 8.6) ("Albuterol" 118 7.0) ("Budesonida" 185 3.2) ("Fluticasona" 176 8.5))
(("Montelukast" 148 3.3) ("Cromoglicato de sodio" 148 5.1) ("Ketotifeno" 14 4.0) ("Amoxicilina" 32 1.5) ("Azitromicina" 141
8.3)) (("Cefalexina" 32 5.5) ("Ceftriaxona" 116 2.9) ("Ciprofloxacino" 56 6.4) ("Doxiciclina" 172 1.6) ("Metronidazol" 141
9.9)) (("Trimetoprima" 92 6.1) ("Aciclovir" 31 8.2) ("Valaciclovir" 167 8.5) ("Oseltamivir" 77 7.2) ("Rituximab" 76 9.9))
(("Infliximab" 38 8.2) ("Adalimumab" 198 8.5) ("Etanercept" 41 8.5) ("Interferón beta-1a" 61 8.3) ("Interferón beta-1b" 120
6.1)) (("Natalizumab" 97 2.0) ("Alemtuzumab" 113 1.6) ("Bevacizumab" 55 2.8) ("Trastuzumab" 12 6.8) ("Cisplatino" 144 7.8))
(("Carboplatino" 130 9.3) ("Paclitaxel" 141 2.7) ("Docetaxel" 49 8.1) ("Irinotecán" 87 1.0) ("5-Fluorouracilo" 106 4.7))
(("Doxorrubicina" 33 2.4) ("Ciclofosfamida" 53 9.4) ("Vincristina" 83 1.3) ("Vinblastina" 112 6.5) ("Bleomicina" 91 3.6))
(("Procarbazina" 64 5.6) ("Dacarbazina" 140 6.1) ("Tamoxifeno" 135 3.5) ("Letrozol" 165 9.6) ("Anastrozol" 59 6.6))
(("Fulvestrant" 174 8.8) ("Leuprorelina" 125 1.5) ("Goserelina" 84 3.3) ("Degarelix" 49 7.5) ("Abiraterona" 71 2.6)))
"Fulvestrant")
```

Para simular el movimiento del carrusel, se utilizarán cuatro letras para moverse:

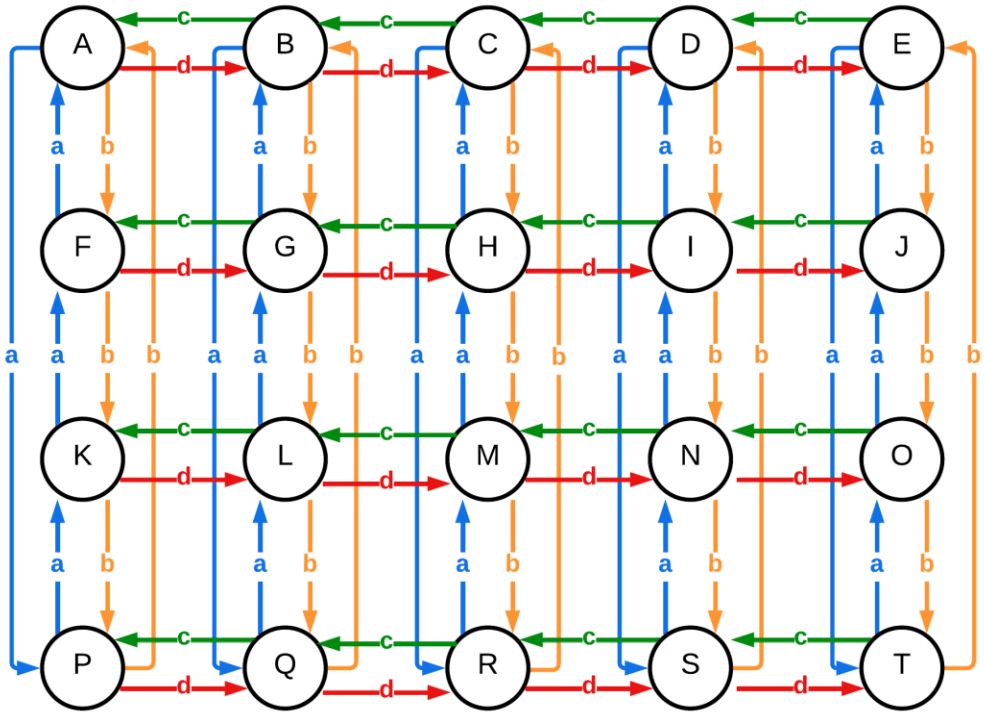
- a, hacia arriba.
- b, hacia abajo.
- c, hacia la izquierda.
- d, hacia la derecha

Diseño del autómata

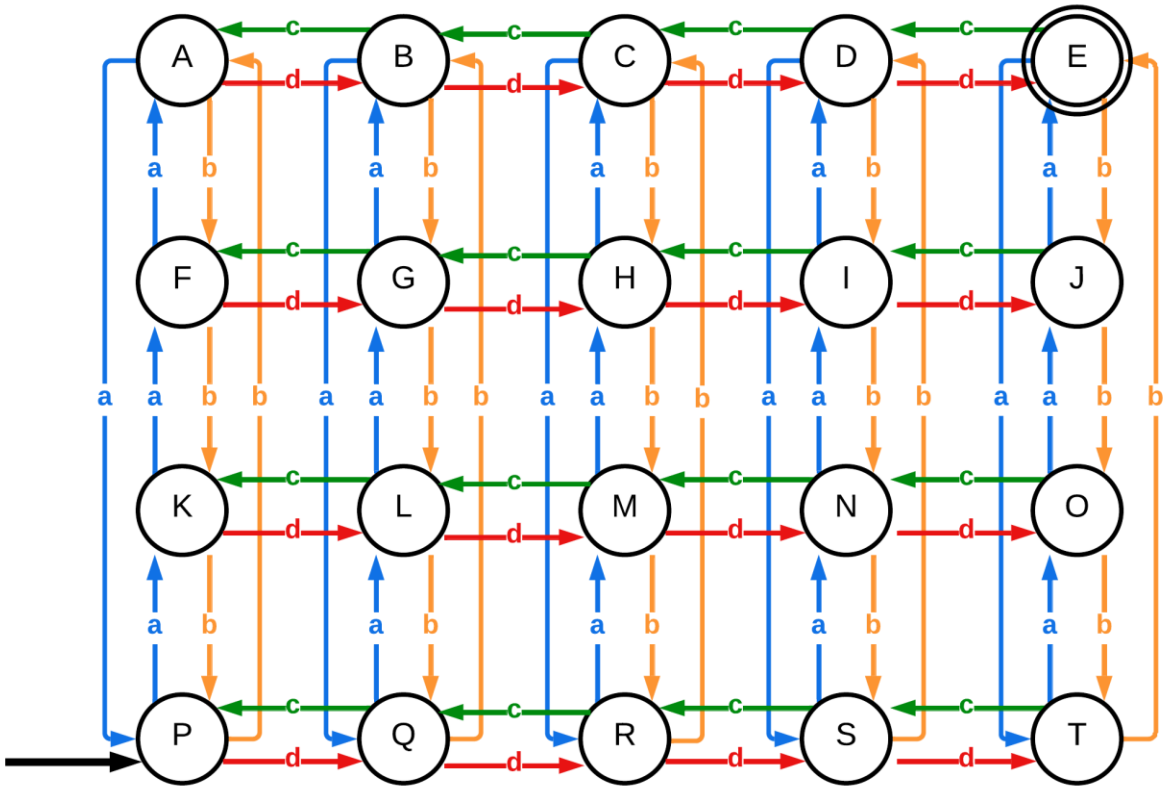
Para representar el movimiento del carrusel, será necesario generar un autómata de estados finitos que identifique los cuatro movimientos posibles y las transiciones entre los distintos productos. Aquí adjunto un autómata simulando un carrusel de 2 filas y 2 columnas:



Como es posible observar, si es que el carrusel se encuentra en la primera fila y se mueve hacia arriba, se regresará a la última fila, y viceversa. Si este se encuentra en la primera columna y se intenta mover a la izquierda, este será un movimiento inválido. Expandiendo este a un carrusel más grande, se obtendrá el siguiente autómata:



Por ejemplo, si el carrusel actualmente está mostrando el producto P, y se quiere llegar al producto E, el estado inicial y el estado aceptador serían los siguientes:



La ruta más eficiente para llegar del estado P al estado E será: (b d d d d).

Las transacciones que se desean realizar con el carrusel deberán estar almacenadas en un archivo llamado “transacciones.txt”, también almacenado en el mismo directorio que el archivo Racket. Las transacciones posibles en el sistema serán:

- **(a) –** El carrusel se moverá hacia arriba.
- **(b) –** El carrusel se moverá hacia abajo.
- **(c) –** El carrusel se moverá hacia la izquierda.
- **(d) –** El carrusel se moverá hacia la derecha.
- **(retirar cantidad {producto}) –** Si no se especifica producto, se retirará la cantidad del producto actualmente mostrado. Si se especifica un producto existente, el carrusel se moverá a la posición del producto especificado, y mostrará la ruta más eficiente para llegar a esta posición. Si se intentan retirar más de las existencias del producto, se retirará todo y se marcará un error.
- **(agregar cantidad {producto}) –** Al igual que la función retirar, si no se especifica un producto se agregará al producto actual, y si sí, se moverá el carrusel a la posición del producto utilizando la ruta más eficiente.

Ejemplo de archivo de transacciones:

```
(retirar 5)
(d)
(agregar 3 "Paroxetina")
(c)
(a)
(a)
```

Al finalizar la ejecución de las transacciones en el archivo, se mostrarán en pantalla dos datos: el valor total del inventario actual, y aquellos productos con poco o cero inventario, en ese caso, especifiqué que los productos con menos de 20 existencias serán considerados con poco inventario.

Algoritmo

Para determinar la ruta más eficiente, se determina si la distancia vertical entre el producto de origen y el de destino es mayor a la mitad de la cantidad de filas, si no la es, es más eficiente moverse hacia arriba, en caso contrario es más eficiente ir hacia abajo. Si la distancia coincide exactamente, será indistinta la dirección en la que se mueva.

Mutabilidad

El código que simula al carrusel es completamente mutable y sigue los principios del paradigma funcional; no se utilizan variables para cambiar datos, sino se utiliza la recursión y la escritura a archivos para lograr la mutabilidad de sus estructuras de datos. El código se adapta automáticamente a cualquier tamaño de carrusel sin necesidad de cambiar el código, y es posible cambiar los nombres, las cantidades y los precios de los productos dentro del carrusel en el mismo archivo de texto, y el código lo manejará.

Ventajas y desventajas

Una de las mayores ventajas de implementar este proyecto en este lenguaje es su paradigma, el cuál es funcional, esto significa que es posible dedicar más tiempo a la solución teórica del problema y menos en programar exactamente que deberá hacer la computadora; sin embargo, la desventaja principal para mí es que el código escrito no es muy legible y es difícil de analizar y explicar, debido a las llamadas a funciones y la recursión que se utilizan ampliamente en este paragidma.

Código

```
#lang racket/load

; Solución a la situación problema #2
; Diego Andrés Figueroa Peart
; A01660987


(define (archivo)   ; Se lee el archivo que contiene el carrusel y el estado actual.
  (define lectura (open-input-file "carrusel.txt"))
  (define contenido (read lectura))
  (close-input-port lectura)
  contenido)

(define (carrusel) (car (archivo)))   ; Se genera la lista del carrusel cada vez que se ejecuta una transacción.
(define (est-actual) (cadr (archivo)))
(define filas (length (carrusel)))
(define columnas (length (car (carrusel))))

(define (coordenadas lst f x y)
  (if (null? lst) f
      (if (<= y columnas)
          (coordenadas (cdr lst) (append f (list (list (caar lst) (cons x y)))) x (+ y 1))
          (coordenadas lst f (+ x 1) 1))))

(define (diccionario) (coordenadas (apply append (carrusel)) '() 1 1))

(define (buscar producto)   ; Se busca un producto a partir de su nombre, se regresan sus coordenadas.
   (cadar (filter (lambda (p) (equal? producto (car p))) (diccionario))))

(define (buscar-f producto)   ; Se busca un producto especificado por el usuario, si se regresa falso es que no existe.
  (define x (filter (lambda (p) (equal? producto (car p))) (diccionario)))
  (if (null? x) #f
      (cadar x)))

(define (detalles x y lst)   ; Se reciben las coordenadas del producto y se regresa su nombre, cantidad y precio.
  (define (elemento y lst)
    (if (not (= 1 y))
        (elemento (- y 1) (cdr lst))
        (car lst)))
  (if (not (= 1 x))
      (detalles (- x 1) y (cdr lst))
```

```
    (elemento y (car lst))))

(define (reemplazar-elem producto x y lst f)  ; Se cambia la cantidad del producto en el carrusel para luego sobreescribir el archivo.
  (define (reemplazar-fila producto y lst f)
    (cond
      ((null? lst) (list f))
      ((= y 1) (reemplazar-fila producto (- y 1) (cdr lst) (append f (list producto))))
      (else (reemplazar-fila producto (- y 1) (cdr lst) (append f (list (car lst)))))))
  (cond
    ((null? lst) f)
    ((= 1 x) (reemplazar-elem producto (- x 1) y (cdr lst) (append f (reemplazar-fila producto y (car lst) '()))))
    (else (reemplazar-elem producto (- x 1) y (cdr lst) (append f (list (car lst)))))))

(define valor-total (apply + (map (lambda (x) (* (cadr x) (caddr x))) (apply append (carrusel)))))  ; Se calcula el valor total del inventario.

(define (productos-bajos-aux lst f)
  (if (null? lst) f
      (productos-bajos-aux (cdr lst) (append f (list (list (caar lst) (cadar lst) (buscar (caar lst))))))))
(define productos-bajos (productos-bajos-aux (filter (lambda (x) (> 20 (cadr x))) (apply append (carrusel))) '()))  ; Se regresan los productos con menos de 20 unidades.

(define (escribir lst)   ; Se sobreescribe el archivo del carrusel con la lista recibida.
  (define output (open-output-file "carrusel.txt" #:exists 'replace))
  (write lst output)
  (close-output-port output))

(define (a)   ; Movimiento hacia arriba.
  (define coords (buscar (est-actual)))
  (define lista (carrusel))
  (if (= 0 (- (car coords) 1))
      (begin
        (displayln (detalles filas (cdr coords) lista)))
      (begin
        (displayln (detalles (- (car coords) 1) (cdr coords) lista))
        (escribir (list (carrusel) (car (detalles (- (car coords) 1) (cdr coords) lista)))))))

(define (b)   ; Movimiento hacia abajo.
  (define coords (buscar (est-actual)))
  (define lista (carrusel))
  (if (= filas (+ (car coords) 1))
      (begin
        (displayln (detalles 1 (cdr coords) lista)))
      (begin
```

```
      (displayln (detalles (+ (car coords) 1) (cdr coords) lista))
      (escribir (list lista (car (detalles (+ (car coords) 1) (cdr coords) lista)))))))

(define (c)  ; Movimiento hacia la izquierda.
  (define coords (buscar (est-actual)))
  (define lista (carrusel))
  (if (= 0 (- (cdr coords) 1))
      (begin
        (displayln "Movimiento inválido."))
      (begin
        (displayln (detalles (car coords) (- (cdr coords) 1) lista))
        (escribir (list lista (car (detalles (car coords) (- (cdr coords) 1) lista)))))))

(define (d)   ; Movimiento hacia la derecha.
  (define coords (buscar (est-actual)))
  (define lista (carrusel))
  (if (< columnas (+ (cdr coords) 1))
      (begin
        (displayln "Movimiento inválido."))
      (begin
        (displayln (detalles (car coords) (+ (cdr coords) 1) lista))
        (escribir (list lista (car (detalles (car coords) (+ (cdr coords) 1) lista)))))))

(define (arriba x1 x2 lst)
  (if (= x1 x2) lst
      (if (= x1 1)
          (arriba filas x2 (append lst (list 'a)))
          (arriba (- x1 1) x2 (append lst (list 'a))))))

(define (abajo x1 x2 lst)
  (if (= x1 x2) lst
      (if (= x1 filas)
          (abajo 1 x2 (append lst (list 'b)))
          (abajo (+ x1 1) x2 (append lst (list 'b))))))

(define (distancia-y y1 y2 lst)
  (if (= y1 y2) lst
      (if (> y1 y2)
          (distancia-y (- y1 1) y2 (append lst (list 'c)))
          (distancia-y (+ y1 1) y2 (append lst (list 'd))))))

(define (distancia x1 y1 x2 y2 lst)   ; Se calcula la distancia entre el producto actual y el producto al que se desea llegar, se determina la ruta más rápida.
  (cond
```

```
   ((< x1 x2) (if (> (- x2 x1) (/ filas 2))
                  (distancia 0 y1 0 y2 (arriba x1 x2 lst))
                  (distancia 0 y1 0 y2 (abajo x1 x2 lst))))
   ((> x1 x2) (if (> (- x1 x2) (/ filas 2))
                  (distancia 0 y1 0 y2 (abajo x1 x2 lst))
                  (distancia 0 y1 0 y2 (arriba x1 x2 lst))))
   (else (begin
           (display "Ruta más corta: ")
           (displayln (distancia-y y1 y2 lst))))))

(define (retirar-n cantidad)  ; Se retira la cantidad del producto que se muestra actualmente.
  (define coords (buscar (est-actual)))
  (define lista (carrusel))
  (define producto (detalles (car coords) (cdr coords) lista))
  (if (< (- (cadr producto) cantidad) 0)  ; Se revisa si la cantidad a retirar es mayor que la existencia.
      (begin
        (displayln "Se intentó retirar más de la cantidad existente.")
        (escribir (list (reemplazar-elem (list (car producto) 0 (caddr producto)) (car coords) (cdr coords) lista '()) (car producto))))
      (escribir (list (reemplazar-elem (list (car producto) (- (cadr producto) cantidad) (caddr producto)) (car coords) (cdr coords) lista '()) (car producto)))))

(define (retirar-p cantidad prod)  ; Se retira la cantidad del producto especificado.
  (define origen (buscar (est-actual)))
  (define destino (buscar-f (car prod)))
  (define lista (carrusel))
  (if (not destino)
      (displayln "Producto no encontrado.")
      (begin
        (if (equal? origen destino)
            (void)
            (distancia (car origen) (cdr origen) (car destino) (cdr destino) '()))
        (if (< (- (cadr (detalles (car destino) (cdr destino) lista)) cantidad) 0)
            (begin
              (displayln "Se intentó retirar más de la cantidad existente.")
              (escribir (list (reemplazar-elem (list (car (detalles (car destino) (cdr destino) lista)) 0 (caddr (detalles (car destino) (cdr destino) lista))) (car destino) (cdr destino) lista '()) (car (detalles (car destino) (cdr destino) lista)))))
            (escribir (list (reemplazar-elem (list (car (detalles (car destino) (cdr destino) lista)) (- (cadr (detalles (car destino) (cdr destino) lista)) cantidad) (caddr (detalles (car destino) (cdr destino) lista))) (car destino) (cdr destino) lista '()) (car (detalles (car destino) (cdr destino) lista))))))))

(define (retirar cantidad . producto)  ; Se especifica un argumento opcional para la función retirar.
  (if (null? producto)
      (retirar-n cantidad)
      (retirar-p cantidad producto)))
```

```
(define (agregar-n cantidad)  ; Se agrega la cantidad al producto que se muestra actualmente.
  (define coords (buscar (est-actual)))
  (define lista (carrusel))
  (define producto (detalles (car coords) (cdr coords) lista))
  (escribir (list (reemplazar-elem (list (car producto) (+ (cadr producto) cantidad) (caddr producto)) (car coords) (cdr coords) lista '()) (car producto))))

(define (agregar-p cantidad prod)  ; Se agrega la cantidad al producto especificado.
  (define origen (buscar (est-actual)))
  (define destino (buscar-f (car prod)))
  (define lista (carrusel))
  (if (not destino)
      (displayln "Producto no encontrado.")
      (begin
        (if (equal? origen destino)
            (void)
            (distancia (car origen) (cdr origen) (car destino) (cdr destino) '()))
        (escribir (list (reemplazar-elem (list (car (detalles (car destino) (cdr destino) lista)) (+ (cadr (detalles (car destino) (cdr destino) lista)) cantidad) (caddr (detalles
(car destino) (cdr destino) lista))) (car destino) (cdr destino) lista '()) (car (detalles (car destino) (cdr destino) lista)))))))

(define (agregar cantidad . producto)  ; Se especifica un argumento opcional para la función agregar.
  (if (null? producto)
      (agregar-n cantidad)
      (agregar-p cantidad producto)))

(define (realizar-transacciones arch)  ; Se leerán y ejecutarán las transacciones línea por línea.
  (load arch))

(realizar-transacciones "transacciones.txt")

(display "Valor total del inventario: ") (displayln valor-total)
(display "Productos con poco o nulo inventario: ") (displayln productos-bajos)
```

Pruebas:

Estado inicial: “Fulvestrant”.

Transacciones:

- (retirar 5)
- (d)
- (agregar 3 "Paroxetina")
- (c)
- (a)
- (a)

Estado final: “Paracetamol”.

```
(Leuprorelina 125 1.5)
Ruta más corta: (b b b)
(Fluoxetina 168 5.5)
(Ketorolaco 55 6.6)
(Paracetamol 99 1.9)
Valor total del inventario: 55730.499999999985
Productos con poco o nulo inventario: ((Escitalopram 12 (4 . 1))
(Atorvastatina 7 (7 . 2)) (Ketotifeno 14 (12 . 3)) (Trastuzumab 12 (16 . 4)))
```

Estado inicial: “Paracetamol”.

Transacciones:

- (agregar 11 "Albuterol")
- (c)
- (c)
- (c) – Movimiento inválido.
- (a)

Estado final: “Loratadina”.

```
Ruta más corta: (b b b b b b b b b b d d)
(Salbutamol 59 8.6)
(Oximetazolina 28 7.8)
Movimiento inválido.
(Loratadina 93 1.1)
Valor total del inventario: 55691.599999999984
Productos con poco o nulo inventario: ((Escitalopram 12 (4 . 1))
(Atorvastatina 7 (7 . 2)) (Ketotifeno 14 (12 . 3)) (Trastuzumab 12 (16 . 4)))
```

Experiencia de aprendizaje

Fue un trabajo arduo y complejo resolver esta situación problema; nunca había tenido experiencias con Racket o cualquier otro lenguaje del paradigma funcional, y en realidad considero que este tipo de trabajo lo podría haber desarrollado más fácilmente con un lenguaje de un paradigma distinto, como Python, o incluso C++. Sin embargo, siempre me es divertido aprender algo nuevo y este fue un reto muy interesante que desafió y desarrolló mi pensamiento recursivo y mi solución de problemas.