Auxiliar 4 - Testing, Estructuras y Listas

Profesor: Patricio Inostroza
Auxiliares: Miguel Sepúlveda
Cristóbal Loyola

30 de Agosto de 2019

P1. Polinomios.

a) Es posible representar un polinomio con una lista de la siguiente forma (formato **A**):

$$6X^3 + 0X^2 + 4X^1 + 7X^0 \longleftrightarrow (7(4(0(6()))))$$

donde la profundidad de la lista corresponde al exponente. Programe la función **evaluarPolA(lista, x)** que evalúa un polinomio representado en la lista del formato A en x.

b) Un polinomio también puede ser representado de la siguiente manera (formato **B**):

$$6X^3 + 0X^2 + 4X^1 + 7X^0 \longleftrightarrow ((70)((41)((63)())))$$

donde cada entrada de la lista contiene la información sobre el coeficiente y exponente, y los coeficientes nulos son omitidos. Defina una estructura que contenga el coeficiente y exponente de un término del polinomio, y programe la función **evaluarPolB(lista, x)** que evalúa un polinomio representado en la lista del formato B en x.

c) Programe la función **ConvertirAB(lista)** que recibe una lista en el formato A y retorna una lista en el formato B, que representa el mismo polinomio.

P2. Vectores. Un vector en $\mathbf{R}^3$ es un elemento que se puede definir como $V = (a_1, a_2, a_3)$, donde a_1, a_2 y a_3 corresponden a las coordenadas de los ejes x, y, z respectivamente. Por medio de estructuras implementaremos el módulo vector con algunas de sus operaciones mas comunes:

a) Cree la estructura **Vector**, que recibe tres componentes numéricas de nombres x, y, z.

b) Cree una función llamada **esVector(V)**, que asegura que el parámetro V representa un vector válido, y entregue True en este caso. Si no, que entregue False o arroje un `AssertionError`.

c) Cree la función **aString(V)** que recibe un vector, y retorna un string de la forma (x, y, z).

d) Cree una función llamada **suma(V1,V2)** que recibe dos vectores,y entregue un nuevo donde x, y, z son las componentes de su suma.

e) Cree una función llamada **productoPunto(V1,V2)**, que recibe dos vectores, y entrega el producto punto de estos dos. Recuerde que el producto punto entre dos vectores $v_1 = (x_1, y_1, z_1) y v_2 = (x_2, y_2, z_2)$ esta dado por $v_1 \Delta v_2 = x_1x_2 + y_1y_2 + z_1z_2$.

f) Cree una función llamada **norma(V)**, que use la función anterior para calcular el módulo de un vector. Recuerde que la norma de un vector está dada por $||v||^2 = v \Delta v$.

P3. Lista de supermercado. Usted quiere ir de compras al "Lyder" y necesita hacer una lista para no olvidarse. En este problema usaremos strings para representar los elementos a comprar.

a) Programe la función **crearListaCompra(lista=listaVacia)** que interactivamente (y recursivamente) va formando una lista a medida que usted ingresa las cosas que va a comprar. Notar que por omisión recibe una lista vacía.

b) Programe la función **mostrarListaCompra(lista)** que muestra en pantalla todos los elementos de la lista de compra, recursivamente.

c) A veces usted necesita saber cuantos elementos contiene la lista, para saber si debe llevar un carro o no. Programe la función **contarLista(lista)** que devuelve un entero que representa la cantidad de elementos de la lista.

d) Usted se dió cuenta que en la b) el programa le muestra toda la lista de compras, pero de forma invertida. Esto no le acomoda ya que generalmente las primeras cosas que ingresa en la lista son las más importantes. Programe la función **invertirLista(...)** que retorna la misma lista pero invertida (debe retornar una lista, no mostrarla en pantalla).

e) Cuando la lista de supermercado es larga, es difícil ver a simple vista si falta algo. Programe la función **listaContiene(elemento, lista)** que retorna True si el elemento está o no en la lista.

f) A veces es necesario borrar un elemento de la lista, sobre todo cuando andamos cortos de dinero. Programe la función **quitarDeLista(elemento, lista)** que retorna una lista sin el elemento, en caso de encontrarse. Asuma que el elemento no está repetido.