J.P. Vidocevich / I. Sánchez
Facultad de Ingeniería y Ciencias Hídricas
Universidad Nacional del Litoral

GTP 2 — Guía de Trabajos Prácticos 2 - Programación 1/3

1. Listas

1. **BasicSort**. Escribir una función `basicsort(L)`, que ordena los elementos de L de menor a mayor. Para ello emplear el siguiente algoritmo simple: utilizando una lista auxiliar L2, tomar el menor elemento de L, eliminarlo de L e insertarlo al final de L2 hasta que L esté vacía. Luego copiar L2 en L.

2. **SelectionSort.** Escribir una función `selectionsort(L)`, que ordena los elementos de L de menor a mayor. Para ello debe tomarse el menor elemento de L e intercambiarlo (swap) con el primer elemento de la lista. Luego intercambiar el menor elemento de la lista restante, con el segundo elemento, y así sucesivamente. Esta función debe ser in-place.

3. **Concatena.** Escriba procedimientos para concatenar: a) dos listas L1 y L2 usando insert; b) una lista LL de n sublistas usando insert; c) una lista LL de n sublistas usando splice. Cada procedimiento debe retornar el resultado en una lista nueva.

4. **Invierte**. Escribir una función `invert(L)`, que invierte el orden de la lista L. Este algoritmo debe implementarse in place y debe ser O(n). Restricción: no utilizar el método `len`. Implementar aplicando:

   a) programación recursiva

   b) *list comprehension*

   Restricción: no utilizar el método `len`.

5. **Junta.** Escribir una función `junta(L, c)` que, dada una lista L, agrupa de a c elementos, dejando su suma *in-place*. Por ejemplo, si se le pasa como argumento la lista $L = (1, 3, 2, 4, 5, 2, 2, 3, 5, 7, 4, 3, 2, 2)$, después de aplicar el algoritmo `junta(L,3)` debe quedar $L = (6, 11, 10, 14, 4)$ (notar que se agrupan los últimos elementos, pese a no completar los tres requeridos). El algoritmo debe tener un tiempo de ejecución O(n).

6. **ReemplazaSecuencia**. Dada una lista de enteros L y dos listas SEQ y REEMP, posiblemente de distintas longitudes, escribir una función `reemplaza(L, SEQ, REEMP)`, que busca todas las secuencias de SEQ en L y las reemplaza por REEMP. Por ejemplo, si L=(1,2,3,4,5,1,2,3,4,5,1,2,3,4,5), SEQ=(4,5,1) y REEMP=(9,7,3), entonces después de llamar a `reemplaza(L,SEQ,REEMP)`, debe quedar $L = (1, 2, 3, 9, 7, 3, 2, 3, 9, 7, 3, 2, 3, 4, 5)$. Para implementar este algoritmo primero buscar desde el principio la secuencia SEQ, al encontrarla, reemplazar por REEMP, luego seguir buscando a partir del siguiente elemento al último de REEMP.

7. **Ascendente**. Escribir una función `ascendente(L, LL)` que, dada una lista L, la subdivide (manteniendo el orden de los elementos de la lista original) y produce una lista de listas LL de tal forma de que cada sublista es ascendente.

8. **Camaleón**. Implemente los predicados `menor(x, y)`, `mayor(x, y)` y `dist(x, y)` que retornen verdadero si x es menor, mayor o menor en valor absoluto que y respectivamente. Luego implemente una función `ordena(L, f)` que ordene la lista L dependiendo de la función `f` pasada cómo parámetro.

9. **Problema de Josephus**. Un grupo de soldados se halla rodeado por una fuerza enemiga. No hay esperanzas de victoria si no llegan refuerzos y existe solamente un caballo disponible para el escape. Los soldados se ponen de acuerdo en un pacto para determinar cuál de ellos debe escapar y solicitar ayuda. Forman un círculo y se escoge un número n al azar. Igualmente se escoge el nombre de un soldado. Comenzando por el soldado cuyo nombre se ha seleccionado, comienzan a contar en la dirección del

reloj alrededor del círculo. Cuando la cuenta alcanza el valor n, este soldado es retirado del círculo y la cuenta comienza de nuevo, con el soldado siguiente. El proceso continúa de tal manera que cada vez que se llega al valor de n se retira un soldado. El último soldado que queda es el que debe tomar el caballo y escapar. Entonces, dados un número n y una lista de nombres, que es el ordenamiento en el sentido de las agujas del reloj de los soldados en el círculo (comenzando por aquél a partir del cual se inicia la cuenta), escribir un procedimiento `josephus(nombres,n)` que retorna una lista con los nombres de los soldados en el orden que han de ser eliminados y en último lugar el nombre del soldado que escapa.

10. **Palíndromo**. Escribir un predicado `es_palindromo(S)`, que reciba como parámetro una cadena de texto S y determine si ésta es un palíndromo, ignorando los espacios entre palabras. Un palíndromo es una secuencia de caracteres que se lee igual hacia adelante que hacia atrás, por ejemplo: alli si maria avisa y asi va a ir a mi silla. Recordar que un string puede indexarse como un vector. Con el fin de utilizar la estructura `<list>`, primero deben pasarse los elementos del string a una lista y solo utilizar ésta en el algoritmo.

11. **Compacta**. Escribir una función `compacta(L, S)` que toma un elemento entero n de S y, si es positivo, saca n elementos de L y los reemplaza por su suma. Esto ocurre con todos los elementos de S hasta que se acaben, o bien se acaben los elementos de L.

12. **maxSubList**. Programar una función `maxsublist(L)` la cual reciba una lista de enteros y encuentre y retorne la sublista Lmax que obtenga la mayor suma entre todos sus elementos. Notar que debido a que algunos elementos pueden ser negativos el problema no se resuelve simplemente tomando todos los elementos. También es posible que la sublista resultado no contenga ningún elemento, en el caso de que todos los elementos de L sean negativos. Si hay varias sublistas que den la misma suma, debe retornar la que comience primero y sea más corta. Por ejemplo: [1,2,-5,4,-3,2]->[4], [5,-3,-5,1,7,-2]->[1,7], [4,-3,11,-2]->[4,-3,11].

13. **Merge**. Escribir una función `merge(L1, L2, L)` la cual recibe dos listas ordenadas (que pueden ser de distinto tamaño) de forma ascendente L1 y L2 y retorna una lista L, pasada como parámetro, con los elementos de ambas ordenados también en forma ascendente. Por ejemplo si L1=[1, 3, 6, 11] y L2=[2, 4, 6, 10], la lista L debe quedar como L=[1, 2, 3, 4, 6, 6, 10, 11].

14. **MergeSort**. Programar una función `mergesort(L)` que reciba una lista L desordenada y la ordene en forma ascendente mediante la siguiente estrategia recursiva: Si la lista está vacía o tiene un sólo elemento ya está ordenada. Si no, se parte la lista en dos sublistas y se las ordena a cada una de forma recursiva. Luego se mezclan (fusionan) cada una de las sublistas ya ordenadas.