Rellenar estos cuadros:

Nombre:	¿Viene de UTU o de Secundaria?
C.I.:	

TECNÓLOGO EN INFORMÁTICA

ESTRUCTURAS DE DATOS Y ALGORITMOS

Primer Parcial

24 de Octubre de 2008

Tiempo Total: 2:30 hs.

Ejercicio 1 (12 puntos)

Supongase que se desea manipular polinomios de la forma p(x)=c1.x^{e_1} + c2.x^{e_2} +.....+cn.x^{e_n.}
donde e_1>e_2>.....>e_n >= 0

a) Representar el polinomio mediante una lista.

b) Escribase un programa para diferenciar polinomios, que devuelva en otra lista el polinomio diferenciado.

c) Escriba programas para sumar polinomios que devuelva el resultado en una tercera lista.

Ejercicio 2 (12 puntos)

a) De una representación para el tipo NodoLNat, para que el tipo LNat sea una lista de naturales.

```
typedef struct NodoLNat* LNat;
…..
…..
```

Nota: el tipo natural esta definido de la siguiente forma

```
typedef unsigned int Natural;
```

b) Utilizando la representación de listas definida en la parte "a)" Implemente accediendo directamente a la representación la función **LNat ConcatEspejo(LNat l)** de forma **iterativa** que dada una lista de naturales devuelva una nueva lista formada por la lista original concatenada con la lista espejo de la original. (La lista espejo es una lista con los mismos elementos pero en orden inverso a la lista original. Ej: si la lista original es [2,3] su lista espejo es [3,2]).

-La lista resultado no debe compartir memoria con la lista original

-Solo se debe recorrer una vez la lista original.

-La lista original no debe ser modificada

Ejemplos:

Original	**Resultado**
l=[1,2,3,6]	Lres=[1,2,3,6,6,3,2,1]
L=[]	Lres=[]
L=[1]	Lres=[1,1]
L=[7,2]	Lres=[7,2,2,7]

Ejercicio 3 (12 puntos)

Dados los siguientes operaciones sobre listas de Naturales.

- void Cons(Natural x, LNat &l);
  /* Inserta un elemento al principio de la lista de naturales. */
- bool IsEmpty(LNat l);
  /* Verifica si la lista de naturales está vacía. */
- Natural getHead(LNat l);
  /* Retorna el primer elemento de la lista de naturales. Precondición: la lista no es vacía.*/
- LNat Null();
  /* Crea la lista de naturales vacía. */
- void Tail(LNat &l);
  /* Elimina de la lista de naturales `l` su primer elemento. Precondición: la lista no es vacía.*/

Se pide:

Natural ElimUlt (LNat &l, int& cant);

Implementar una función recursiva **ElimUlt,** que dada una lista l de tipo LNat :
- elimine todas las ocurrencias del ultimo elemento de la lista en la misma
- retorne el ultimo elemento de l
- y en el parámetro cant devuelva la cantidad de veces que fue eliminado este valor de la lista.

La función debe eliminar todas las ocurrencias del último elemento.
ElimUlt tiene como precondición que l no es la lista vacía.

Ejemplos:

Lista Original	**Resultado**
l= [2,8,6,8,8]	l= [2,6], retorna: 8 y cant en 3
l= [6,2,3]	l= [6,2] , retorna: 3 y cant en 1
l= [2,2]	l= [], retorna: 2 y cant en 2

NOTAS:

-La función no debe recorrer la lista de manera explícita más de una.
- No se permite usar funciones o procedimientos auxiliares, aunque si pueden utilizarse las 5 operaciones rimitivas dadas sobre listas de tipo LNat. De hecho, no puede asumirse una representación particular de listas de tipo LNat, la única manera de manipularlas es a través de las 5 operaciones referidas.

-No pueden definirse estructuras de datos adicionales a la lista de tipo LNat.
Asuma que en la primera llamada a la función el parámetro cant viene igualado a cero

Ejercicio 4 (4 puntos)
De la representación del TAD Contacto utilizado en el laboratorio el curso.