Estructuras de Datos y Algoritmos

PRIMER PARCIAL – 01/10/2009

Nombre y Apellido	C.I.	Turno

Por favor siga las siguientes indicaciones:

- Escriba con lápiz.
- Escriba su nombre y número de documento en todas las hojas que entregue.
- Numere las hojas e indique el total de hojas en la primera de ellas.
- Escriba las hojas de un solo lado.
- Comience cada ejercicio en una hoja nueva.
- El total máximo de puntos del parcial es **40**.
- El parcial contiene un total de: **5** páginas.
- La prueba es individual y sin material.
- Solo se contestan dudas acerca de la letra de los ejercicios.
- Las preguntas del **problema 2** son de múltiple opción. Solo existe una **única** respuesta correcta, su respuesta debe darse en los casilleros de abajo. Cada una de esas preguntas bien contestada vale: **2** puntos, mal contestada: **-0,5** puntos, no contestada: **0** punto. Escriba con claridad, respuestas dudosas se consideraran no contestadas.

Respuestas a los ejercicios de Múltiple Opción:

	Pregunta 1:	Pregunta 2:	Pregunta 3:	Pregunta 4:
PROBLEMA 2				

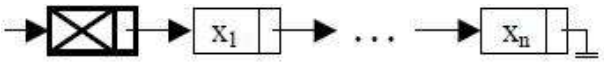
	Problema 1:	Problema 2:	Problema 3:	Total:
Solo para uso docente				

Problema 1 [14 puntos: 7 parte a) y 7 parte b)]

Considere la siguiente declaración, en C/C++, del tipo de las listas de números enteros `LInt`:

```
typedef struct LInt {
            int info;
            LInt *sig;
};
```

Donde cada lista será representada usando una celda dummy (cabecera). La lista $[X_1, \ldots, X_n]$, con $n \geq 0$, será representada:



Se pide, sin usar funciones o procedimientos auxiliares y accediendo directamente a la representación:

a) Implementar, en C/C++, la siguiente operación como una *función iterativa*:

Copia: Dada una lista de enteros de tipo `LInt` (representada con celda dummy) y dados dos enteros positivos *k1* y *k2*, retorna una nueva lista (representada con celda dummy) que contiene a los elementos de la lista parámetro que se encuentran entre las posiciones *k1* y *k2* (rango [k1, k2]). La nueva lista **no** debe compartir registros de memoria con la lista parámetro.

Ejemplos:

Entradas	**Salida de *Copia***
[1, 2, 0, -8, 3, 9], k1=2, k2=4	[2, 0, -8]
[1, 2, 0, -8, 3, 9], k1=5, k2=10	[3, 9]
[1, 2, 0, -8, 3, 9], k1=7, k2=8	[ ]
[1, 2, 0, -8, 3, 9], k1=3, k2=2	[ ]
[ ], k1=1, k2=1	[ ]

b) Implementar, en C/C++, la siguiente operación como un *procedimiento iterativo*:

BorrarMin: Dada una lista de enteros de tipo `LInt` (representada con celda dummy), elimina la primer ocurrencia del mínimo elemento de la lista. Si la lista es vacía, el procedimiento no tendrá efecto. El procedimiento no debe recorrer la lista más de una vez.

Ejemplos:

Entradas	**Salida de *BorrarMin***
[2, 8, 6, 1, 9]	[2, 8, 6, 9]
[2, 8, 6, 1, 9, 1]	[2, 8, 6, 9, 1]
[ ]	[ ]
[4]	[ ]

Problema 2 [8 puntos]

1) Suponga que está usando una representación de listas simplemente encadenadas y no circulares, con un apuntador al primer elemento. Si L apunta a la lista con elementos 1, 2, 3, 4, 5, ¿cuál será la lista resultante a la que apunta L después de la ejecución de:
`L->sig->sig->sig = NULL`?

(a) 1, 2, 3
(b) 1, 2, 3, 4
(c) 1, 2, 4, 5
(d) 1, 2, 3, 5

2) Suponga que está usando una representación de listas circulares simplemente encadenadas con un apuntador L al primer elemento. Si L apunta a la lista con elementos 1, 2, 3, 4, 5, ¿cuál será la lista resultante a la que apunta L después de la ejecución de `L->sig = NULL`?

(a) 1
(b) 1, 3, 4, 5
(c) 1, 2, 3, 5
(d) Ninguna – la lista no será una lista circular bien formada.

3) ¿Cuál de los siguientes códigos borrará exitosamente el último elemento de una lista circular doblemente encadenada con un apuntador L al primer elemento de la lista? (Asuma que la lista tiene como mínimo cinco elementos)

(a)
```
L->ant->sig = L->sig
L->sig->ant = L->ant
```

(b)
```
x = L->ant
x->ant->sig = x->sig
x->sig->ant = x->ant
```

(c)
```
x = L->sig
x->ant->sig = x->sig
x->sig->ant = x->ant
```

(d) Más de uno de los anteriores.

4) Suponga que está trabajando con listas simplemente encadenadas y no circulares con un apuntador L al primer elemento. A continuación se muestra un código para insertar un nuevo elemento con dato 'A' al comienzo de la lista. ¿Bajo qué condiciones el código fallará?

```
x = new Nodo;
x->dato = 'A';
x->sig = L;
L = x;
```

(a) La lista está vacía.
(b) La lista tiene un número impar de elementos.
(c) La lista tiene un solo elemento.
(d) No fallará.

Problema 3 [18 puntos: 5 parte a), 6 parte b) y 7 parte c)]

Consideré el tipo Natural y el tipo Lista de Naturales (LNat) según las siguientes definiciones en C/C++:

```
typedef unsigned int Natural;
typedef struct LNat; //opaco
```

y la siguiente **especificación del TAD LNat**:

```
// ****** Constructoras ******

LNat* Vacia();
// Post: Devuelve la lista vacía.

LNat* Insertar(Natural x, LNat* L);
// Post: Devuelve la lista resultado de insertar el natural
// x al principio de la lista L.

// ****** Predicados ******

bool EsVacia(LNat* L);
// Post: Devuelve true sii la lista L es vacía.

// ****** Selectoras ******

Natural Primero(LNat* L);
// Pre: !EsVacia(L)
// Post: Devuelve el primer elemento de la lista L.

LNat* Resto(LNat* L);
// Pre: !EsVacia(L)
// Post: Retorna un alias de la lista L sin su primer elemento.

Natural getElem(LNat* L, Natural pos);
// Pre 1: !EsVacia(L)
// Pre 2: 0 < pos <= Largo(L)
// Post: Devuelve el natural que se encuentra en la posición
// 'pos' de la lista L.

// ****** Otras ******

Natural Largo(LNat* L);
// Post: Devuelve la cantidad de elementos de la lista L
```

Definimos la propiedad EDA_UNO sobre el tipo LNat, de tal forma que dada una lista de Naturales **L** (no vacia) y una determinada posición **POS** de alguno de sus elementos, **L** cumple la propiedad EDA_UNO(**POS**), si la cantidad de números primos de la lista desde la posición 1 hasta la posición **POS** (ambas inclusive) es par. Donde 0 < **POS** ≤ Largo(**L**).

Ejemplo 1:
Se la lista **L**=[7, 3, 10],

- EDA_UNO(1) =FALSE ya que la cantidad de números primos desde la posición 1 hasta pos=1 es impar(7 es primo).
- EDA_UNO(2) =TRUE ya que 7 y 3 son primos (posiciones 1 y 2 respectivamente).

- EDA_UNO(3) =TRUE ya que los elementos en la posición 1 y 2 son primos, y el valor de la posición 3 (valor 10) no es primo.

Ejemplo 2:
Se la lista **L**= [4],

- EDA_UNO(1) =TRUE ya que la cantidad de números primos desde la posición 1 hasta pos=1 es cero (4 no es primo).

Se pide:

a) Implemente la función **esPrimo**, que retorna verdadero si un número natural es primo. Utilice el siguiente encabezado para implementar la función:

```
bool esPrimo(Natural a);
```

b) Sin acceder a la representación y utilizando las funciones del TAD LNat, y la función **esPrimo** de la parte a). Defina e implemente recursivamente la función:

```
Natural cantidadPrimos(LNat* L, Natural pos);
// Pre: 0< pos <= Largo(L)
// Pre: !EsVacia (L)
```

Que retorna la cantidad de naturales que son primos en la lista L, entre la posición 1 y la posición pos (ambas inclusive).

c) Sin acceder a la representación y utilizando si las necesita, las funciones dadas para manipular el TAD LNat, y las funciones de las partes a) y b), implemente la función recursiva:

```
bool EDA_UNO(LNat* L, Natural pos)
// Pre: 0< pos <= Largo(L)
// Pre: !EsVacia (L)
```

Que retorna **`true`** si se cumple la propiedad EDA_UNO (pos), y **`false`** en otro caso.

Nota: No se pueden utilizar funciones ni procedimientos auxiliares, salvo los indicados en la letra.