Estructuras de Datos y Algoritmos

PRIMER PARCIAL – 01/10/2009

Solución

_____ _____

Nombre y Apellido C.I.

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, 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	**A**	**D**	**B**	**D**

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

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

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.

```
// función iterativa
LInt* Copia(LInt* l, int k1, int k2) {
      LInt *iterCopia, *resCopia;
      int pos;

      l = l->sig;
      pos = 1;

      /* Creamos la lista para contener el resultado
       * de la funcion.
       * Creamos una celda dummy para simplificar el
       * ingreso de nuevos elementos.*/
      resCopia = new LInt;
      iterCopia = resCopia;
      if (k1 <= k2) {

            /* recorremos la lista l hasta la posicion k1*/
            while ((l != NULL) && (pos < k1)) {
                  pos = pos + 1;
                  l = l->sig;
            }

            /* recorremos la lista l para tomar los elementos
             * hasta la posicion k2*/
            while ((l != NULL) && (pos <= k2)) {
                  iterCopia->sig = new LInt;
                  iterCopia = iterCopia->sig;
                  iterCopia->info = l->info;
                  pos = pos + 1;
                  l = l->sig;
            }
      }
      iterCopia->sig = NULL;
      return resCopia;
}
```

b) Implementar, en C/C++, la siguiente operación como una *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.

```
// procedimiento iterativo
void BorrarMin(LInt *& l) {
      LInt *iterl, *posMin, *borrar;
      int min;

      if (l->sig != NIL) {
            /* Asumimos como minimo actual el primer elemento
             * de la lista. Almacenamos su valor y su posicion */
            min = l->sig->info;
            posMin = l;
            iterl = l->sig;

            /* recorremos la lista l comparando sus elementos
             * contra el minimo actual */
            while (iterl->sig != NULL) {

                  /* Si el elemento de la lista es menor que el
                   * minimo actual, lo tomamos como el nuevo
                   * minimo.
                   * Es importante notar que para poder eliminar
                   * posteriormente el minimo de la lista sin
                   * romperla, debemos mantener la posicion del
                   * elemento anterior al minimo, de forma tal de
                   * poder reencadenar la lista, excluyendo de
                   * ella al minimo */

                  if (iterl->sig->info < min) {
                        min = iterl->sig->info;
                        posMin = iterl;
                  }
                  iterl = iterl->sig;
            }

            borrar = posMin->sig;
            posMin->sig = posMin->sig->sig;
            delete borrar;
      }
}
```

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

Definimos la propiedad EDA_UNO sobre el tipo LNat, de tal forma que dada una lista de Naturales **L 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**).

Se pide:

a)

```
bool esPrimo(Natural N) {
      int divisor, raiz;

      raiz = (int) sqrt(N);

      /*  N es un numero primo si sólo es divisible por N
       o por la unidad. */
      for (divisor=2; ((N%divisor)!=0) && divisor<=raiz; divisor++);

      /* Si se llego a dividir N entre todos los números menores
       que su raíz cuadrada entonces es un primo */

      if (divisor == raiz + 1)
            return true;
      else
            return false;

}
```

b)

En primer lugar definimos la función:

$$cantidadPrimos: LNat * int \rightarrow int$$

- $cantidadPrimos(x.xs, 0) = 0$
- $cantidadPrimos(x.xs, pos) = \begin{cases} 1 + cantidadPrimos(xs, pos-1) \; si \; esPrimo(x) \\ cantidadPrimos(xs, pos-1) \; en \; otro \; caso \end{cases}$

De acuerdo a lo anterior pasamos a la implementación en C/C++:

```
Natural cantidadPrimos(LNat *L, Natural pos) {

      if (pos == 0)
            return 0;
      else {
            if (esPrimo(Primero(L)))
                  return (1 + cantidadPrimos(Resto(L), pos-1));
            else
                  return (cantidadPrimos(Resto(L), pos-1));
      }
}
```

c)

Al igual que la parte anterior, definimos la función de la siguiente forma:

$$EDA\_UNO: LNat * int \rightarrow boolean$$

- $EDA\_UNO(x.xs, pos) = \begin{cases} true\ si\ cantidadPrimos(x.xs, pos) = \dot{2} \\ false\ en\ otro\ caso \end{cases}$

De acuerdo a la definición anterior, implementación la función en C/C++:

```
bool EDA_UNO(LNat* L, Natural pos) {
      return (cantidadPrimos(L, pos)%2 == 0);
}
```