Examen Programación 2

27 de Julio de 2017

Generalidades:

a. La prueba es individual y sin material. b. La duración es 3hs. c. Sólo se contestan dudas acerca de la letra.	d. Escriba las hojas de un sólo lado y con letra clara. e. Numere cada hoja, indicando en la primera el total. f. Coloque su nro. de cédula y nombre en cada hoja.

Problema 1 (25 puntos)

Implemente la operación *Combinar* que dadas dos listas ordenadas (de forma **ascendente**) de enteros *l1* y *l2* de tipo *ListaEnt*, crea y devuelve otra lista ordenada (también de forma **ascendente**) de enteros de tipo *ListaEnt* que contiene los elementos de ambas listas sin elementos repetidos. El *TAD ListaEnt* de listas con manejo explícito de posiciones tiene la siguiente especificación:

```
// Crea y devuelve una lista vacía que puede tener hasta N elementos.
ListaEnt vaciaLEnt(int N);

// Devuelve el largo de la lista l, si es vacía devuelve 0.
int largoLEnt(ListaEnt l);

// Inserta x en l en la posición p, los elementos que estaban en las posiciones
// desde p y hasta largoLEnt(l) son desplazados una posición hacia adelante.
// Pre: largoLEnt(l)<largoMaxLEnt(l) y 1<=p<=largoLEnt(l)+1.
ListaEnt insLEnt(ListaEnt l, int x, int p);

// Devuelve el elemeno de l que está en la posición p. Pre: 1<=p<=largoLEnt(l).
int darLEnt(ListaEnt l, int p);

// Devuelve true si l es vacía.
bool esVaciaLEnt(ListaEnt l);

// Devuelve el largo máximo de la lista.
int largoMaxLEnt(ListaEnt l);
```

Una lista con manejo explícito de posiciones permite insertar, borrar y devolver elementos en una posición dada, las posiciones deben tomar valores enteros.

Sol Problema 1

```
ListaEnt Combinar (ListaEnt l1, ListaEnt l2){
ListaEnt lresult= vaciaLEnt(largoLEnt(l1)+ largoLEnt(l2));
int contador=0;
int posl1=posl2=1;
int ultElem;

 if(largoLEnt(l1)+ largoLEnt(l2) >0) {

    if(largoLEnt(l1)>0 && largoLEnt(l2)>0) {
       if(darLEnt(l1, posl1) < darLEnt(l2, posl2)){
           ultElem= darLEnt(l1, pos1);
           contador++;
           lresult = insLEnt(lresult, ultElem, contador);
```

```
            pos1++;
        } else {
            ultElem= darLEnt(l2, pos2);
            contador++;
            lresult = insLEnt(lresult, ultElem, contador);
            pos2++;
     } else {
              if(largoLEnt(l1)>0){
                  ultElem= darLEnt(l1, pos1);
                  contador++;
                  lresult = insLEnt(lresult, ultElem, contador);
                  pos1++;
              } else {
                  ultElem= darLEnt(l2, pos2);
                  contador++;
                  lresult = insLEnt(lresult, ultElem, contador);
                  pos2++;
              };
     };

    };


     while (pos1<=largoLEnt(l1) && pos2<=largoLEnt(l2) ) {

          if(darLEnt(l1, posl1) < darLEnt(l2, posl2)){
             if (darLEnt(l1, posl1)> ultElem){
                  ultElem= darLEnt(l1, pos1);
                  contador++;
                  lresult = insLEnt(lresult, ultElem, contador);
             }
             pos1++;
          } else {
                if (darLEnt(l2, posl2)> ultElem){
                   ultElem= darLEnt(l2, pos2);
                   contador++;
                   lresult = insLEnt(lresult, ultElem, contador);
                }
                pos2++;
           };
     };

     while (pos1<=largoLEnt(l1)) {
          if (darLEnt(l1, posl1)> ultElem){
                  ultElem= darLEnt(l1, pos1);
                  contador++;
                  lresult = insLEnt(lresult, ultElem, contador);
            }
            pos1++;
     };
     while (pos2<=largoLEnt(l2) ) {
          if (darLEnt(l2, posl2)> ultElem){
                  ultElem= darLEnt(l2, pos2);
                  contador++;
                  lresult = insLEnt(lresult, ultElem, contador);
                }
                pos2++;
     };

  };
   return lresult;
}
```

Problema 2 (a) 10 puntos b) 25 puntos c) 5 puntos)

Considere el TAD *ColaPrio* que es una cola de prioridad no acotada, donde los elementos guardados son enteros. El entero más prioritario es el de menor valor. Se pide:

a) Especifique completamente el TAD *ColaPrio* (con pre y post condiciones).

b) Implemente el TAD definiendo el tipo para la representación elegida y desarrollando el código de todas las operaciones, de forma tal que las operaciones selectoras sean O(1) de tiempo de ejecución en el peor caso.

c) Explique cómo la implementación elegida permite cumplir los requerimientos.

Sol Problema 2

a)

```
/ * Devuelve una cola de prioridad sin elementos. * /
ColaPrio crearPQ ( ) ;
/ * Inserta 'elem' con prioridad 'elem' en 'q'. * /
void insertarPQ (int elem , ColaPrio & q) ;
/ * Devuelve 'true' si y sól o si 'q' no t i e ne elementos. * /
bool esVaciaPQ (ColaPrio q) ;
/ * Devuelve el menor elemento de 'q'.
Precondición: !esVaciaPQ(q). * /
int prioritario (ColaPrio q) ;
/ * Elimina el menor elemento de 'q'.
Precondición: !esVaciaPQ(q) . * /
void eliminarPQ (ColaPrio &q)
```

b)

```
ColaPrio crearPQ ( ) {
return NULL ;
}
void insertarPQ (int elem , ColaPrio & q) {
nodoColaPrio * nuevo = new nodoColaPrio ;
nuevo->dato = elem ;
q  if ( ( q == NULL ) | | ( elem < q>dato ) ) {
        nuevo->sig = q ;
        q = nuevo ;
   } else {
       nodoColaPrio * cursor = q ;
       while ( ( cursor->sig != NULL ) && ( elem > cursor->sig->dato ) ) {
         cursor = cursor->sig ;
       }
       nuevo->sig = cursor->sig ;
       cursor->sig = nuevo ;
   }
}
bool esVaciaPQ (ColaPrio q)  {
    return ( q == NULL ) ;
}
int prioritario (ColaPrio q) {
    return q->dato ;
}
void eliminarPQ (ColaPrio &q ) {
   nodoColaPrio * borrar = q ;
   q = q->sig ;
   delete borrar ;
```

```
}
```

c) La cola de prioridad se implementa con una lista de nodos enlazados cuyos datos están ordenados de manera creciente. De esta manera el prioritario siempre es el primer elemento de la la lista y puede obtenerse, para prioritario, y eliminarse, para eliminarPQ, en O(1).

Problema 3 (35 puntos)

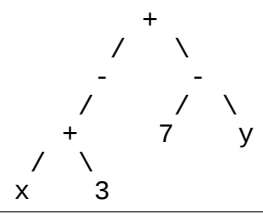
Considere la siguiente representación de árboles binarios mediante el tipo *AB*, de árboles binarios de strings:

Representación

```
struct nodo {
      string dato;
      nodo * izq, * der;
            };
typedef nodo * AB;
```

Ejemplo

Entrada



Salida

```
7 y x 3
```

Se pide implementar un procedimiento llamado *hojasPorNiveles* que dado un árbol de tipo *AB*, imprima los elementos de las hojas, por niveles de arriba hacia abajo y cada nivel de izquierda a derecha. Las hojas del árbol se deben imprimir en un sólo renglón separadas por un espacio.

Se debe implementar el procedimiento de forma iterativa, accediendo directamente a la representación del árbol, de forma tal que tenga O(*n*) de tiempo de ejecución en el peor caso, siendo *n* la cantidad de nodos del árbol.

Use el TAD *ColaAB* de elementos de tipo *AB* que se especifica a continuación para resolver el problema. No use otras estructuras de datos, TADs auxiliares, procedimientos o funciones auxiliares.

```
// Crea y devuelve una cola de AB vacío.
ColaAB vaciaColaAB();

// Inserta un AB a en una ColaAB c
ColaAB insColaAB(ColaAB c, AB a);

// Devuelve el elemento más antiguo de una ColaAB c. Pre: !esVaciaColaAB(c).
AB obtenerColaAB(ColaAB c);

// Devuelve la ColaAB c sin el elemento más antiguo. Pre: !esVaciaColaAB(c).
ColaAB eliminColaAB(ColaAB c);

// Devuelve true si la ColaAB c es vacía
bool esVaciaColaAB(ColaAB c);
```

Se puede suponer que todas las operaciones del TAD ColaAB tienen tiempo de ejecución O(1) en el peor caso.

También se puede asumir que los strings se pueden asignar y comparar como tipos básicos (por ejemplo usando los operadores: =, ==, <, >).

Sol Problema 3

```
// Crea y devuelve una cola de AB vacío.

void hojasPorNiveles ( ABB a ) {
 if ( a != NULL ) {
  ColaABB c = vaciaColaABB ( ) ;
  c = insColaABB ( c , a ) ;
  while ( ! esVaciaColaABB ( c ) ) {
      ABB frente = obtenerColaABB ( c ) ;
      if ( frente != NULL ) {
       if ( frente->izq == NULL && frente->der == NULL)
         printf ( " %s " , frente->dato ) ;
       else {
         c = insColaABB ( c , frente->izq ) ;
         c = insColaABB ( c , frente->der ) ;
       }
      }
      c = eliminarColaABB ( c ) ;
   }
  }
}
```