ESTRUCTURA DE DATOS Y ALGORITMOS
UTU, Buceo

Turno nocturno - Segundo Parcial 30/11/2010

Escriba nombre y cédula de identidad en todas las hojas. El parcial suma 60 puntos.

Ejercicio 1 (18 puntos)

Considere la siguiente definición del tipo LISTA, de listas de enteros en memoria dinámica:

```
struct nodoLista {
        int info;
        nodoLista *sig;
};
typedef nodoLista *LISTA;
```

Se pide, sin usar funciones o procedimientos auxiliares, implementar la siguiente función iterativa:
InsComFin: Dada una lista L de enteros de tipo LISTA y dado un entero x, retorna una nueva lista (que no comparte registros de memoria con la lista parámetro) que contiene a todos los elementos de L en orden inverso y a x como primer y último elemento. La función no debe recorrer L más de una vez y la lista resultado no debe recorrerse.

Ejemplos:

Entradas	Resultado de *InsComFin*
L = [2,3,7,8], x = 4	[4,8,7,3,2,4]
L = [4,8], x = 8	[8,8,4,8]
L = [4], x = 4	[4,4,4]
L = [], x = 4	[4,4]

Solución:

```
lista InsComFin(lista l,int x)
{     lista p1,p2;

      p1 = new nodoLista;
      p1->info=x;
      p1->sig=NULL;
      while (l!=NULL)
      {     p2 = new nodoLista;
            p2->info=l->info;
            p2->sig=p1;
            p1=p2;
            l=l->sig;
            }
      p2 = new nodoLista;
      p2->info=x;
      p2->sig = p1;
      return p2;
}
```

Ejercicio 2 (30 puntos)

Considere la siguiente definición del tipo ABB de los árboles binarios de búsqueda de enteros, en memoria dinámica:

```
struct nodoABB {
        int dato;
        nodoABB * izq; // menores que dato
        nodoABB * der; // mayores que dato
};
typedef nodoABB *ABB;
```

a) Defina una función recursiva *BorrarMin* que dado un árbol binario de búsqueda de enteros A de tipo ABB, no vacío (precondición), elimine y retorne de A su mínimo elemento. No se pueden usar funciones o procedimientos auxiliares. *BorrarMin* debe evitar recorrer nodos innecesarios de A. Deberá liberarse la memoria de la celda cuyo elemento sea eliminado. A luego de la función debe ser un árbol binario de búsqueda.
int BorrarMin (ABB & A)

b) Defina un procedimiento *Borrar* que dado un árbol binario de búsqueda de enteros A de tipo ABB, no vacío (precondición), elimine el entero que se encuentra en la raíz de A. No se pueden usar funciones o procedimientos auxiliares, con excepción de la función definida en la parte (a). *Borrar* no debe recorrer A, aunque si puede hacerlo eventualmente *BorrarMin*. A luego del procedimiento debe ser un árbol binario de búsqueda.
void Borrar (ABB & A)

Solución parte a:

```
//Precondicion: A es no vacio.
int BorrarMin(ABB &A){
     ABB auxABB;
     if (A->izq != NULL){
          return BorrarMin(A->izq);
     }
     else {
          auxABB = A;
          A = A -> der;
          delete auxABB;
          auxABB = NULL;
     }
}
```

Solución parte b:

```
void Borrar(ABB &A){
     ABB auxABB=A;
     if (auxABB->derecho==NULL) {
          A=auxABB->izquierdo;
          delete auxABB;
          auxABB = NULL;
     }
     else {
          //puedo invocar BorrarMin porque ya se que el hijo derecho no es nulo
          A->valor=BorrarMin(auxABB->derecho);
     }
}
```

Ejercicio 3 (12 puntos)

Defina un orden de recorrida de un árbol binario de búsqueda, de modo que al utilizar ese orden para imprimir el árbol, los valores se impriman de mayor a menor. Escriba la función correspondiente.

Solución:

```
void Imprimir_mayor_a_menor(ABB &A){
     if (A != NULL){
          Imprimir_mayor_a_menor(A->der);
          cout << A->dato << endl;
          Imprimir_mayor_a_menor(A->izq);
     }
}
```

Se imprime primero el subárbol derecho, luego la raíz y luego el subárbol izquierdo. Este mismo criterio se aplica en cada subárbol, recursivamente. De esta forma por se un árbol binario de búsqueda, se logra que los nodos se impriman de mayor a menor. Notar que es el orden inverso al que se logra con una recorrida inorder, donde se imprime primero el subárbol izquierdo, luego la raíz y luego el subárbol derecho (así para cada subárbol) logrando que los valores se impriman de menor a mayor.