Universidad De La República | Administración Nacional de la Educación Pública

Consejo de Educación Técnico Profesional | Facultad de Ingeniería

Estructuras de Datos y Algoritmos

Examen 24/07/2012

_____	_____	_____
Nombre y Apellido	C.I.	Turno

Por favor siga las siguientes indicaciones:

- Escriba con lápiz.
- Escriba su nombre y c.i. en todas las hojas que entregue.
- Numere las hojas he 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 examen es de **100.**
- Se necesitan 60 puntos para aprobar.
- El parcial contiene un total de: **3 carillas.**
- La prueba es individual y sin material.
- Solo se contestan dudas acerca de letra de los ejercicios.

Ejercicio 1(35 Puntos)

Considere la siguiente declaración, en C/C++, del tipo `ListaEnt` de listas encadenadas de enteros:

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

a) Implemente un procedimiento iterativo ***InvNoNeg*** que dada una lista de enteros L, devuelve una nueva lista, que no comparte memoria con L.

La nueva lista contiene todos los elementos de L que sean **mayores o iguales a 0** en el orden inverso al que aparecen en L. Si L esta vacia retorna la lista vacia.

Nota: No se pueden utilizar funciones auxiliares ni recorrer la lista parámetro mas de una vez.

b) Implemente un procedimiento recursivo ***SumaPares*** que dada una lista de enteros, retorne la suma de los números pares en dicha lista.

c) ¿Cuál es el orden de tiempo de ejecución en el peor caso de las operaciones *InvNoNeg*. Justifique.

Ejemplos:

L = {4, -1, 0, 78, -3, -56, 6} → Salilda: L' = {6, 78, 0, 4}
L = {-4, -8, -3,} → Salilda: L' = {}
L = {} → Salilda: L' = {}
L = {1, 0, 7, 35, 6} → Salilda: L' = {6, 35, 7, 0, 1}

Ejercicio 2(30 Puntos)

Considere las siguientes declaraciones, en C/C++, del tipo de los árboles binarios de búsqueda de enteros, de tipo ABB, y del tipo de los árboles binarios de pares de números naturales, de tipo ABPar:

```
struct nodoABB {

    int dato;

    ABB izq;

    ABB der;

};

typedef nodoABB *ABB;
```

```
struct nodoABBPar {
    int hizq;
    int hder;
    ABB izq;
    ABB der;
};
typedef nodoABBPar *ABBPar;
```

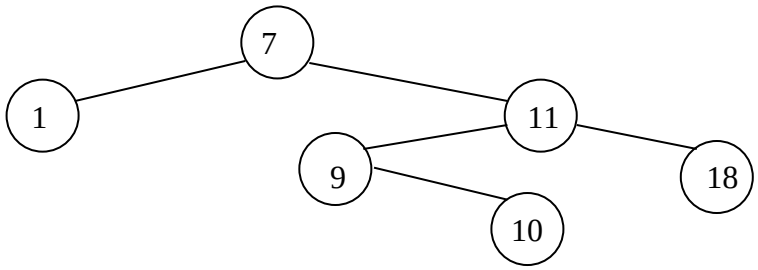
Se pide implementar las siguientes funciones:

a) ***altura***, que dado un árbol de tipo ABB devuelve la altura del mismo.

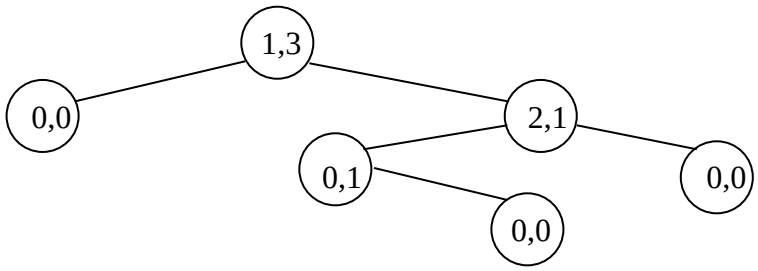
b) ***DesBalanceo***, que dado un árbol de tipo ABB devuelve un árbol estructuralmente igual de tipo ABBPar que en cada nodo guarda la la altura de cada uno de los arboles hijos (en *hizq* altura del hijo izquierdo, en *hder* altura del hijo derecho).

No esta permitido usar funciones auxiliares con excepción de la implementada en la parte a).

Ejemplo:



El resultado de Anotar para el árbol anterior seria:



Ejercicio 3(35 Puntos)

a) Especifique en C/C++, con pre y postcondiciones, un TAD Diccionario no acotado de Strings* que contenga operaciones constructoras, predicados y la operación de búsqueda de palabras.

b) Implemente el TAD Diccionario anterior usando Listas encadenadas como representación. Desarrolle el código de las operaciones de inserción, borrado y búsqueda de una palabra.

c) ¿Puede implementarse el T AD Diccionario de forma estática (utilizando arreglos)?

d) En caso afirmativo ¿Qué cambios le haría al TAD de la parte a)? Explique muy brevemente.

*Asuma que los Strings se pueden asignar simplemente utilizando el operador =. Y se pueden comparar con los operadores <, >, ==, !=.

Nota: Recuerde que un diccionario es un conjunto de palabras ordenado alfabéticamente que no contiene palabras repetidas.

SOLUCION

Ejercicio 1(35 Puntos)

a)
```
ListaEnt InvNoNeg(ListaEnt l){
            ListaEnt ret = NULL;

            while(l != NULL){
                    if (l->info >= 0){
                       ListaEnt nodoLista = new nodoLista;
                       nodoLista->info = l->info;
                       nodoLista->sig = ret;
                       ret = nodoLista;
                    }
                    l = l->sig;
            }
            return ret;
   }
```

b)
```
int sumaPares(ListaEnt l){
        if (l == NULL)
               return 0;
        else if (l->info % 2 == 0)
              return l->info + SumarPares(l->sig);
         else
              return SumarPares(l->sig);

   }
```

c) Es de O(n), donde n es el tamaño de la lista l, por que debo recorrer toda la lista.

Ejercicio 2(30 Puntos)

a)
```
int altura(ABB a){
      if (a == NULL)
         return 0;
      else{
         int aizq = altura(a->izq);
         int ader = altura(a->der);
         if (aizq > ader)
            return aizq;
         else
            return ader;
      }
   }
```

b)
```
ABBPar desBalanceo(ABB a){
        ABBPar res = NULL;
        if (a != NULL){
              res = new nodoABBPar();
              res->izq = desBalanceo(a->izq);
              res->der = desBalanceo(a->der);
              res->hizq = altura(a->izq);
              res->hder = altura(a->der);
        }
        return res;
   }
```

Ejercicio 3(35 Puntos)

a)
```
Diccionario CrearDiccionario();
//post: retorna un Diccionario vacio

void InsetarPalabra(Diccionario &d, String s);
//pre: s no pertenece al Diccionario
//post: agrega la palabra s al Diccionario d, respetando el
//orden

bool esVacio(Diccionario d);
//post: retorna true si el Diccionario d es vacio, false en otro
//caso
```

```
bool Pertenece(Diccionario d, String s);
//retorna true si s pertenece a d, false en otro caso

void borrarPalabra(Diccionario &d, String s);

//pre: s pertenece a d
//post: borra s del Diccionario d
```

b)

```
struct nodoLista {
      string pal;
      nodoLista* sig; };
typedef nodoLista* Diccionario;

void InsetarPalabra(Diccionario &d, String s){
      Diccionario aux = new nodoLista;
      aux->pal = s;

      if ((d != NULL) && (d->pal > s)){
            aux->sig = d;
            d = aux;
      }
      else{
            Diccionario iter = d
            while ((iter->sig != NULL) && (iter->sig->pal < s)){
                  iter = iter->sig;
            }
            aux->sig = iter->sig;
            iter->sig = aux;
      }
}

void Pertenece(Diccionario d, String s){
      while((d!=NULL) && (d->pal != s)){
            d = d->sig;
      }
      if(d == NULL)
            return false;
      else
            return true;
}
```

```
void borrarPalabra(Diccionario &d, String s){
      Diccionario temp;
      if (d->pal == s){
            temp = d;
            d = d->sig;
            delete temp;
      }
      else{
            Diccionario iter = d;
            while ((iter->sig!=NULL) && (iter->sig->pal != s)){
                  iter = iter->sig;
            }
            temp = iter->sig;
            iter->sig = temp->sig;
            delete temp;
      }
}
```

c) Si se puede.
d) Para usar arreglos la representación del TAD Diccionario debe ser acotada, por lo que necesitamos una nueva operación que nos indique si el diccionario esta o no esta lleno.