ESTRUCTURAS DE DATOS Y ALGORITMOS

CURSO 2009 – PRÁCTICO 5

Objetivos: Aplicar los conceptos del TAD Árbol.

1-

a) Dar una representación para árboles binarios de naturales (BinaryTree).
b) Implementar las siguientes operaciones funcionales:

```
/* Devuelve el árbol vacío*/
BinaryTree* NullTree();

/* Crea un árbol no vacío a partir de un natural y otros dos árboles*/
BinaryTree* ConsTree(int x, BinaryTree *l, BinaryTree *r);

/* Determina si un árbol dado es o no vacío */
bool IsEmptyTree(BinaryTree *t);

/* Devuelve el valor en la raíz de un árbol no vacío */
int RootTree(BinaryTree *t);

/* Devuelve el subárbol izquierdo de un árbol no vacío */
BinaryTree* LeftTree(BinaryTree *t);

/* Devuelve el subárbol derecho de un árbol no vacío */
BinaryTree* RightTree(BinaryTree *t);
```

2- Implementar las operaciones de recorrida sobre arboles binarios devolviendo la lista que resulta:

```
LNat* InOrder(BinaryTree *t);
LNat* PreOrder(BinaryTree *t);
LNat* PostOrder(BinaryTree *t);
```

a) Utilizando solamente las operaciones del ejercicio 2.b del práctico 4 y ejercicio 1.b del presente práctico (sin acceder a la representación).
b) Accediendo directamente a la representación.

3- Considere las siguientes operaciones sobre arboles binarios:

```
int Height(BinaryTree *t);
/* Retorna la distancia máxima desde la raíz a un nodo + 1 */

int CountElements(BinaryTree *t);
/* Retorna la cantidad de nodos del árbol */

int SINV(BinaryTree *t);
/* Número de subárboles izquierdos no vacíos */
```

```
int SDNV(BinaryTree *t);
/* ídem anterior para subárboles derechos */

int NH(BinaryTree *t);
/* Cantidad de hojas, es decir nodos sin hijos */

int NT(BinaryTree *t);
/* cantidad de nodos con algún hijo */

int NE2H(BinaryTree *t);
/* Cantidad de nodos con exactamente 2 hijos */

int NE1H(BinaryTree *t);
/* Cantidad de nodos con exactamente 1 hijo */

int NSHI(BinaryTree *t);
/* Cantidad de nodos con sólo hijo izquierdo */

int NSHD(BinaryTree *t);
/* Cantidad de nodos con sólo hijo derecho */
```

a) Dar una representación e implementar las operaciones anteriores accediendo directamente a la estructura.
b) Implementar las operaciones anteriores trabajando solamente con las operaciones del ejercicio 1.b

4- Se pide:

a) Dibuje un árbol binario de búsqueda generado por la secuencia de inserciones siguiente, con las letras ordenadas alfabéticamente: H, J, A, C, B, Z, T.
b) Lo mismo para la siguiente secuencia de números en el orden habitual: 7, 3, 6, 2, 1, 4, 8, 5.
c) Aplique los tres algoritmos de recorrido a los arboles generados en las dos partes anteriores.

5- Se pide:

a) Dar una representación para los Arboles Binarios de Búsqueda de Naturales (BST – Binary Search Tree)
b) Implementar las siguientes operaciones funcionales:

```
BST* NullBST();
/* Crea el árbol vacío */

BST* InsertBST(int x, BST *st);
/* Agrega un natural dado a un árbol dado.
 * Debe de mantener la cualidad de árbol binario de búsqueda*/

bool IsEmptyBST(BST *st);
/* Determina si un árbol dado es o no vacío */

int RootBST(BST *st);
/* Devuelve el natural contenido en la raíz de un
 * árbol no vacío dado */

BST* LeftBST(BST *st);
/* Devuelve el subárbol izquierdo de uno no vacío dado */
```

```
BST* RightBST(BST *st);
/* Devuelve el subárbol derecho de uno no vacío dado */
```

c) Implementar las siguientes operaciones procedurales:

```
void NullBST(BST &*st);
/* Crea el árbol vacío */

void InsertBST(int x, BST &*st);
/* Agrega un natural dado a un árbol dado.
 * Debe de mantener la cualidad de árbol binario de búsqueda */

void IsEmptyBST(BST *st, bool &e);
/* Determina si un árbol dado es o no vacío */

void RootBST(BST *st, int &r);
/* Devuelve el natural contenido en la raíz de un árbol no vacío dado */

void LeftBST(BST &*st);
/* Devuelve el subárbol izquierdo de uno no vacío dado */

void RightBST(BST &*st);
/* Devuelve el subárbol derecho de uno no vacío dado */
```

6- Implementar las siguientes operaciones sobre Árboles Binarios de Búsqueda accediendo directamente a la representación:

```
bool IsElementBST(int x, BST *bs);
/* Determina si el elemento x se encuentra el árbol */

int MaxBST(BST *bs);
/* Computa el máximo valor de un árbol no vacío */

int MinBST(BST *bs);
/* Computa el mínimo valor de un árbol no vacío */

void RemoveMaxBST(BST &*bs);
/* Elimina el máximo de un árbol */

void RemoveMinBST(BST &*bs);
/* Elimina el mínimo de un árbol */

void RemoveBST(int x, BST &*bs);
/* Elimina de un árbol un valor dado */
```

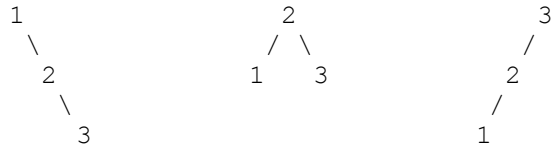
7- Considerar las siguientes declaraciones, en C/C++, de un árbol binario de búsqueda de naturales y de una lista de naturales que siguen:

```
typedef struct ABB {
            int info;
            ABB* left;
            ABB* rigth;
};
```

```
typedef struct LNat {
            int data;
            LNat* next;
};
```

Se pide:

a) Definir una función que dados dos árboles binarios de búsqueda de naturales retorne **true** sii ambos arboles representan a un mismo conjunto. Por ejemplo para el conjunto {1, 2, 3} hay tres arboles binarios de búsqueda que lo representan:



b) Definir una función que dado un árbol binario de búsqueda de naturales y un natural k, retorne la lista de los elementos que se encuentran en el nivel k, ordenados de mayor a menor. La raíz de un árbol no vacio se encuentra en el nivel 1.si no existe el nivel k, la lista a retornar debe ser vacia.

8- Considere la definición de la función f que sigue:

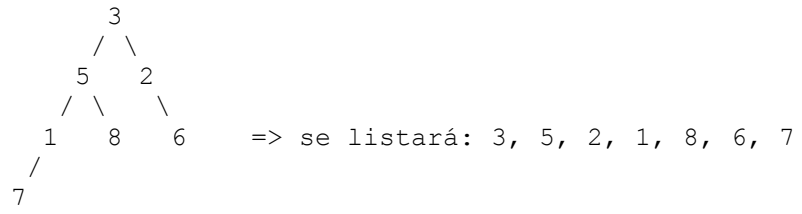
```
bool f(ABBNode *t, int k) {
      if (t == NULL)
            return (k==0);
      else
            return (f(t->izq, k-1) && f(t->der, k-1));
}
```

a) ¿Qué retorna la función f?
b) ¿Cuántos nodos tiene el árbol si la función retorna true? Justifique.

9- Considerar la siguiente declaración, en C/C++, de un árbol binario de búsqueda de naturales:

```
typedef struct ABB {
            int info;
            ABB* left;
            ABB* rigth;
};
```

Escribir un procedimiento que imprima los nodos de un árbol binario de naturales recorriéndolo por niveles, como sigue: los nodos de un mismo nivel deben aparecer listados de izquierda a derecha y después que los nodos de los niveles superiores. Esto es, de arriba hacia abajo, por niveles y de izquierda a derecha. Ejemplo: el árbol



10- Considerar las declaraciones, en C/C++, del tipo de los nodos de un árbol binario de búsqueda de naturales y del tipo de los nodos de una lista de naturales que siguen:

```
typedef struct ABB {
            int info;
            ABB* left;
            ABB* rigth;
};
```

```
typedef struct LNat {
            int data;
            LNat* next;
};
```

Se pide:

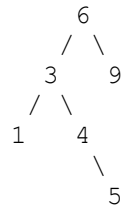
a) Definir una función en c/c++ que dado un árbol binario de naturales devuelva en una lista el camino más largo entre la raíz y una hoja. En caso de haber más de un camino de igual longitud a la del camino más largo, retorna cualquiera de ellos. La lista que contiene el camino devuelto empieza con el dato de la raíz del árbol y termina con el dato de la hoja del árbol, siempre que el árbol no sea vacio.

b) ¿Cuántos nodos tiene como mínimo y como máximo el camino más largo desde la raíz a una hoja, para un árbol de *n* nodos?

11- Considere el siguiente tipo de datos en C/C++ para los nodos de un árbol binario de naturales:

```
typedef struct ABB {
            int info;
            ABB* left;
            ABB* rigth;
};
```

Escribir un procedimiento *caminos* en C/C++ que dado un árbol binario de búsqueda de naturales, imprima todos los caminos desde la raíz hacia una hoja. Por ejemplo, para:




```
imprime: 6, 3, 1
         6, 3, 4, 5
         6, 9
```