Taller Pre-Parcialito 2 [10/05] - Jueves

1. Muestre paso a paso cómo ordenar la siguiente lista de palabras usando *radix-sort*: `[COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX]` ¿De qué orden es este algoritmo?

2. A) Mostrar cómo se modifica la estructura interna de un árbol *AVL* al realizar las siguientes operaciones: insertar 14, insertar 3, insertar 8, insertar 5, insertar 6, insertar 7, insertar 2, insertar 10, insertar 1, insertar 13.

B) Mostrar el resultado de los recorridos *INORDER*, *POSORDER* y *PREORDER*.

3. Se tiene un *árbol binario de búsqueda* con cadenas como claves y función de comparación *strcmp*. Implementar una primitiva `lista_t* abb_mayores(const abb_t* abb, const char* cadena)` que dado un ABB y una cadena, devuelva una lista ordenada que contenga las claves del árbol que sean mayores a la cadena recibida por parámetro (que no necesariamente está en el árbol).

Suponer que la estructura del TDA es:

```
typedef struct abb {
    const char* clave;
    struct abb* izq;
    struct abb* der;
} abb_t;
```

Aclaración: se debe realizar la menor cantidad posible de comparaciones.

4. Implementar una primitiva `void ab_espejar(ab_t* ab)` que dado un árbol binario, devuelva el mismo árbol espejado. Indicar y justificar el orden.

5. A) Para un hash cerrado, implementar una primitiva `lista_t* hash_claves(const hash_t*)`, que reciba un hash y devuelva una lista con sus claves. Aclaración: no se puede utilizar el iterador externo del hash. Indicar y justificar el orden.

B) Repetir lo mismo para un hash abierto.