Examen Final de PRED

1. (8 puntos) Se desea implementar una guía telefónica. Se desea que la estructura de datos tenga dos formas de consulta:

- num-tel, que dado el nombre de un abonado nos devuelva su número de teléfono
- listar, que nos escribe la lista de todos los abonados, junto sus teléfonos, por orden alfabético de abonados.

Se ha decidido implementar la abstracción Guía utilizando, por una parte, hashing con encadenamiento para hacer eficiente la primera consulta y, por otra, encadenando todos los nodos de la tabla formando una lista ordenada de abonados. Es decir:

clase Guía
    t_hash: **tabla** [1..3000] **de** ^nodo; /* tabla hash de los abonados y teléfonos */
    primer: ^nodo; /* puntero que apunta al primer abonado por orden alfabético */
    … /* operaciones de la clase */
f_clase

clase nodo
    abonado: string;
    tel: entero;
    sig_hash: ^nodo; /* puntero al siguiente sinónimo de la tabla hash */
    sig_abo: ^nodo; /* puntero al siguiente abonado por orden alfabético */
f_clase

se pide:

a) Especificar la implementación de la guía. Es decir, definir el invariante y el dominio de la representación.

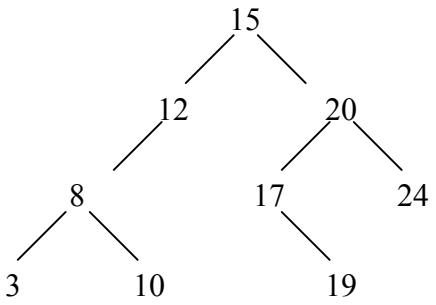
b) Especificar (por medio de pre y post condiciones) e implementar la operación *añadir* que, dado el nombre de un abonado y su número de teléfono, nos los añade a la Guia. Si el abonado ya existiera, la operación debería de substituir el número viejo por el nuevo. Justificar la corrección de la implementación de la operación. Se supone que la función:

    privada función hash (s:string) **retorna** i: entero
    Pre: cierto
    Post: $1 \leq i \leq 3000$

ya está implementada y que se puede usar para hacer las inserciones en la tabla hash.

c) Supongamos que los abonados no fueran simplemente un string, sino que estuvieran formados por tres campos (el nombre, la calle y el número de la calle) y que quisiéramos tener otra operación de consulta, pags_azules, que dada una calle nos devolviera, siguiendo el orden de los números de la calle, los abonados que viven en esa calle y sus teléfonos (por simplicidad, suponed que en cada número sólo vive un abonado). Explicad brevemente cómo cambiaríais la representación de la abstracción para poder implementar la nueva operación eficientemente.

2. (2 puntos) Un árbol binario de búsqueda es un árbol que cumple la siguiente propiedad: dado cualquier nodo del árbol, su valor es mayor o igual que todos los valores que tiene en su subárbol izquierdo y menor o igual que todos los que tiene en el derecho. Por ejemplo, el siguiente árbol es un árbol binario de búsqueda:



Se pide diseñar un algoritmo que, dado un árbol binario de búsqueda, nos escriba la lista ordenada, de menor a mayor, de todos los valores que contiene. Se considera que estos árboles están implementados con las siguientes clases:

clase arbol_bin
    raiz: ^nodo; /* puntero que apunta a la raíz del árbol */
    … /* operaciones de la clase */
f_clase

clase nodo
    valor: entero;
    hijo_izqdo: ^nodo; /* puntero al hijo izquierdo */
    hijo_dcho: ^nodo; /* puntero al hijo derecho */
f_clase