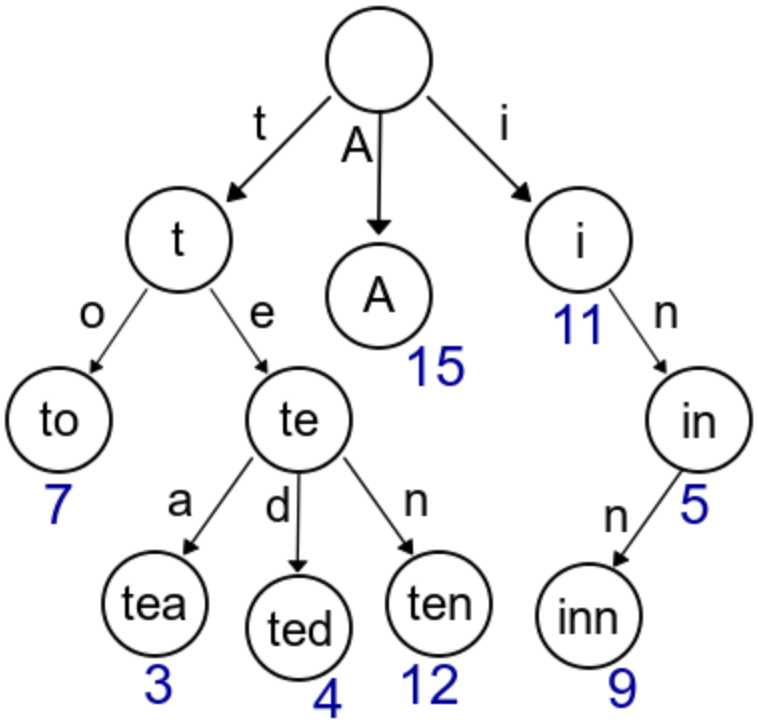
Proyecto Computación II Ene-Jun 2019 (10 puntos)

Se requiere implementar una agenda telefónica que para cada contacto contenga nombre, números de teléfono y correo electrónico, utilizando la estructura abstracta de datos Trie. Un Trie es un tipo especial de árbol que consiste en una serie de nodos organizados (en este caso alfabéticamente) de forma que la clave que se busca está almacenada en el recorrido. Cada nodo tiene asociado un valor que determina si es o no el camino que se debe tomar para llegar al destino. En este caso un nodo cualquiera puede tener hasta 26 hijos y cada uno de ellos tiene asociada una letra del alfabeto.



Se puede observar como para llegar al nodo con la clave “inn” se comienza desde la raíz, luego se busca cuál nodo tiene asociada la letra “i” y se avanza hasta él, luego se repite el proceso buscando recursivamente buscando el nodo con “n” y luego nuevamente la “n” hasta llegar al objetivo.

Interfaz y Requerimientos

Se requiere implementar una interfaz sencilla en la consola para administrar la agenda. Debe consistir de un menú que permita al usuario realizar una serie de acciones comunes enumeradas a continuación.

1. **Buscar Contacto**

   Solicitar al usuario una cadena de caracteres, seguidamente se debe mostrar en consola la información de todos los contactos cuyo nombre comience con dicha cadena ordenados lexicográficamente. Por ejemplo:

```
Buscar: An
Ana         0555-555-5555    ana@dominio.com
Anna        0666-666-6666    anna@dominio.com
Antonio     0777-777-7777    antonio@dominio.com
Anya        0888-888-8888    anya@dominio.com
```

2. **Agregar Contacto (2 ptos)**

   Solicitar al usuario que ingrese por consola el nombre, teléfono y correo del contacto para luego agregarlo en la posición correspondiente del trie. El programa debe chequear que el teléfono y el correo cumplan con un formato adecuado, de lo contrario se descarta la acción y se notifica al usuario del error. Por simplicidad NO deben haber contactos con el mismo nombre, en caso de haberlo se debe actualizar la información del contacto ya existente. Por otro lado los contactos solamente deben tener caracteres del alfabeto inglés; no se permiten espacios y las mayúsculas y minúsculas se deben tomar como la misma letra (Es decir, Juana y juana son el mismo contacto).

3. **Eliminar Contacto (3 ptos)**

   Solicitar al usuario que ingrese por consola una cadena de caracteres, si la cadena ingresada concuerda exactamente con el nombre de un usuario, indicarlo en un mensaje y pedir confirmación, en caso de haber más de un resultado debe listarlos y preguntar al usuario cuál desea eliminar. Por ejemplo:

```
Eliminar: Juli
Julia       0555-555-5555    julia@dominio.com
Juliana     0666-666-6666    juliana@dominio.com
Julio       0777-777-7777    julio@dominio.com
¿Qué contacto desea eliminar? : Julio
¿Seguro que desea eliminar a Julio? Si
Contacto Julio eliminado

Eliminar: Angelina
¿Seguro que desea eliminar a Angelina? Si
```

```
Contacto Angelina eliminado
```

4. **Exportar Agenda (1 pto)**
   Solicitar al usuario una cadena cualquiera que será el nombre de un archivo de salida en el que se guardarán todos los contactos almacenados en el trie en orden lexicográfico.

```
Archivo de Salida:
Ana         0555-555-5555    ana@dominio.com
Anna        0666-666-6666    anna@dominio.com
Antonio     0777-777-7777    antonio@dominio.com
Anya        0888-888-8888    anya@dominio.com
Julia       0555-555-5555    julia@dominio.com
Juliana     0666-666-6666    juliana@dominio.com
Julio       0777-777-7777    julio@dominio.com
```

5. **Importar Agenda (1 pto)**
   Solicitar al usuario una cadena cualquiera que será el nombre de un archivo de entrada del que se leerán una serie de contactos en cualquier orden con sus respectivos datos y serán almacenados en un nuevo trie que reemplazará al anterior.

```
Formato de Archivo de entrada: (Separados por tabulación)
Nombre1    Teléfono1  Correo1
Nombre2    Teléfono2  Correo2
Nombre3    Teléfono3  Correo3
```

6. **Salir**
   Cierra el programa inmediatamente.

Estructuras de Datos y Funciones Requeridas

Contacto (1 pto)

```c
typedef struct contacto {
    char nombre[64];
    char telefono[32];
    char correo[64];
} _contacto;
```

```
typedef _contacto* Contacto;

Contacto nuevoContacto(char* nombre, char* telefono, char* correo);
void liberarContacto(Contacto);
```

Lista de Tries (2 ptos)

```
typedef struct nodoT {
    struct trie* trie;
    struct nodoT* sig;
} _nodoT;

typedef _nodoT* ListaT;

ListaT nuevoNodoT(char);
```

Crea una nueva instancia de nodo de lista enlazada. Internamente debe crear un Trie vacío con la letra del parámetro asociada.

```
void eliminarNodoT(ListaT, char);
```

Elimina la instancia de memoria del nodo indicado. Debe tener cuidado de eliminar todo aquello que esté referenciado dentro del nodo antes de proceder.

Trie

```
typedef struct trie {
    char letra;
    Contacto contacto;
    ListaT hijos;
} _trie;
```

El primer campo representa la letra que está asociada a este nodo. El segundo puede tener almacenado un contacto si todo el camino recorrido concuerda con el nombre del mismo, si no hay ningún contacto significa que el nodo es simplemente parte del recorrido de otros contactos. El último campo es una lista enlazada que debe tener almacenado en orden

lexicográfico todos los hijos del nodo, en un principio está vacía hasta que se agregue un contacto cuyo recorrido deba pasar por el nodo.

```
typedef _trie* Trie;
```

```
Trie nuevoTrie(char, Contacto);
```

Crea una nueva instancia en memoria de un nodo de Trie. El primer parámetro denota la letra asociada al nodo, el segundo tiene a un contacto en caso de ser necesario, de lo contrario recibe NULL.

```
void eliminarTrie(Trie);
```

Elimina una instancia de Trie. Si hay otras instancias hijas también deben ser eliminadas. Recordar que no se pueden dejar fugas de memoria.

```
void agregarContacto(Trie, Contacto);
```

Recibe la raíz de un Trie y un contacto, debe recorrer el trie hasta llegar al lugar adecuado. Si ya hay un contacto debe ser sobreescrito si el camino no existe deber ser creado recursivamente.

```
void eliminarContacto(Trie, char*);
```

Elimina del Trie recibido en el primer parámetro el contacto indicado en el segundo parámetro. En caso de no haber nodos hijos al llegar al destino, se debe eliminar también el nodo.

```
void listarContactos(Trie, char*);
```

Recibe la raíz de un trie y lo recorre por el camino indicado en la cadena de caracteres del segundo parámetro. Al llegar al destino debe de hacer un recorrido inorder del trie mostrando todos los contactos que se encuentren.

```
void exportarAgenda(Trie, char*);
```

Recorre el trie desde la raíz inorder y cada contacto conseguido debe ser almacenado en un archivo de salida usando el nombre indicado en el segundo parámetro.

```
Trie importarAgenda(char*);
```

Recibe el nombre de un archivo y lee secuencialmente cada registro del mismo mientras almacena cada contacto en un nuevo trie. Debe devolver el trie resultante.

```
void salir();
```

¿Qué se imaginan que hace esto?

Recuerden que pueden utilizar cualquier otra función que consideren pertinente. ¡Éxitos!