ICOM2016 – 2do Parcial

24 de noviembre de 2016

Notas:

1. Al finalizar, enviar por e-mail los archivos fuente de cada ejercicio con nombre APELLIDO_NOMBRE_Ejer_N.c a icom.cabib@gmail.com
2. Uso de prácticos: **se pueden utilizar los trabajos prácticos propios realizados.**
3. Uso de Internet**: solo para la consulta de referencias de funciones de C**

Problema 1: Uso de tries

Es una característica común en las aplicaciones de navegación satelital (GPS) contar con una facilidad de búsqueda de destinos (claves). En tal facilidad el usuario ingresa el nombre del sitio deseado utilizando un teclado virtual táctil en la pantalla del equipo.

Para eliminar la posibilidad de búsquedas de sitios que no se encuentran en la base de datos del sistema se desea diseñar un teclado en donde, a un dado momento, solo se encuentren habilitadas las teclas que permiten expandir el prefix actual (destino parcialmente ingresado) en un nuevo prefix o clave **existentes** en la base de datos del programa.

Dada la estructura de Trie vista en el trabajo práctico 15, y sus funciones de manipulación que se encuentran ya implementadas en los archivos trie.h y trie.c se solicita implementar las siguientes funciones:

a. Función que debe poner `enabledKeys[i] = 1` o `0`, indicando si la tecla `i` debe estar habilitada o no, dado el `prefix` enviado. Prototipo: (respetarlo!)

```
void trieGetEnabledKeys(const Trie_t *trie, const char *prefix, int enabledKeys[ALPHABET_SIZE]);
```

b. Función que debe retornar una lista de claves que comparten el `prefix` dado. Podría suponerse que no existen claves más largas que MAX_KEY_LEN (por ejemplo 32). Prototipo:

```
ListWord_t trieGetKeys(const Trie_t *trie, const char *prefix);
```

En donde:

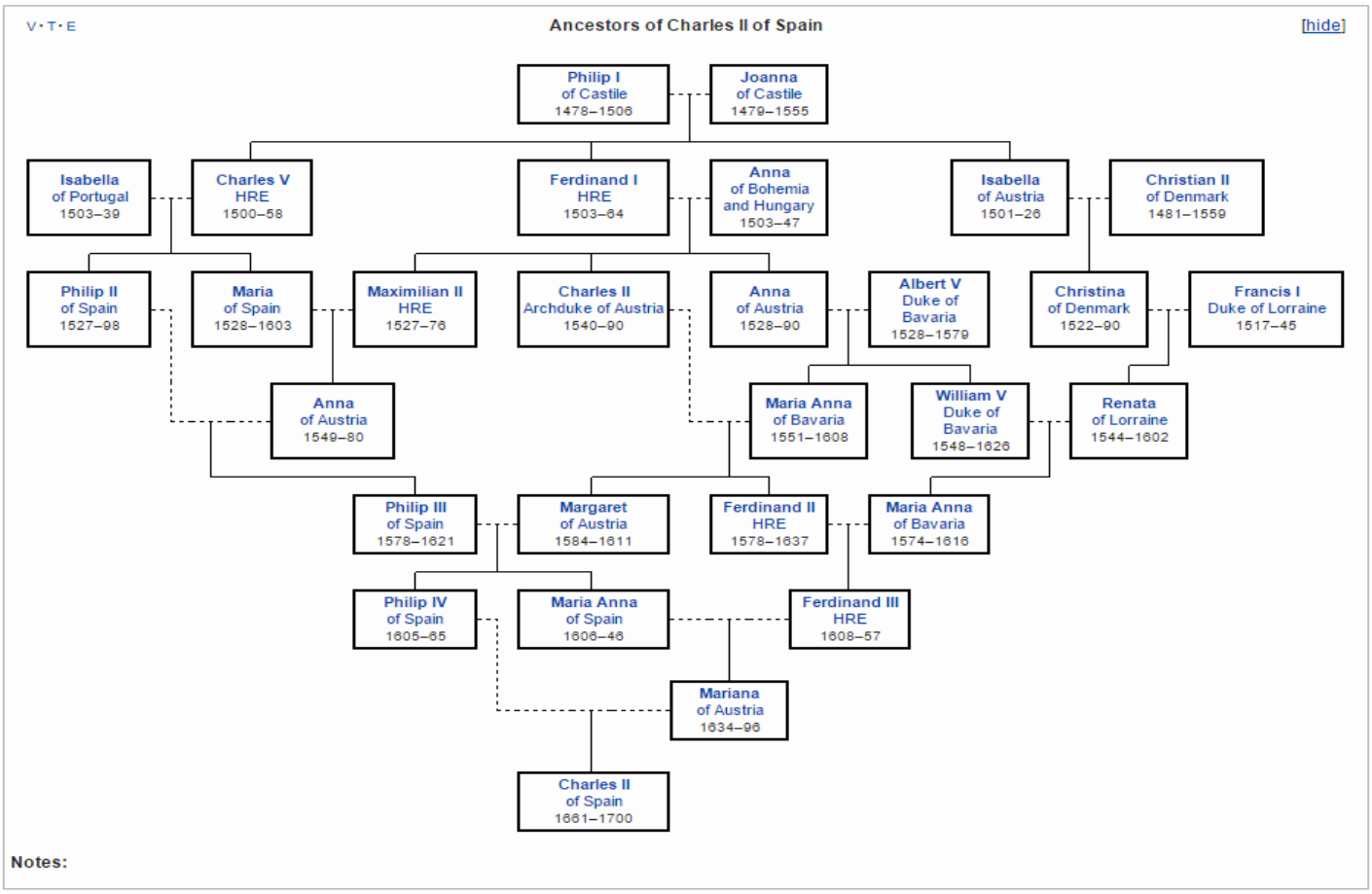
```
typedef struct WordNode {
    char key[MAX_KEY_LEN];
    struct WordNode *pNext;
} *ListWord_t;
```

En trie.c hay un main praparado para leer un archivo de calles y probar el código.

Hint: leer con atención la implementación de las funciones de manejo del trie para aprovechar la reutilización de funciones.

Problema 2: Árbol Genealógico

Dado que un individuo tiene exactamente un padre y una madre, el árbol genealógico es un árbol binario. Bueno, mas o menos. Debido la procreación entre personas emparentadas (**endogamia**), un ancestro puede aparecer más de una vez en el árbol. A esto se lo conoce como **colapso de pedigri** y hace que el árbol no crezca exponencialmente.
Diversos desórdenes genéticos son atribuidos a la endogamia. Basta con ver el árbol familiar de Carlos II de España para entender por qué le llamaban “el hechizado”.



La relación de parentesco entre una persona **A** que es ancestro de otra persona **O**, separada por **n** generaciones, está dada por $p_{AO} = 2^{-n}$ suponiendo que no hay otras relaciones endogámicas entre ellos, o sea que la relación con cada padre es de ½, con cada abuelo ¼, etc.
Se puede calcular un coeficiente de relación de parentesco entre dos personas cualesquiera **B** y **C**, como la suma del producto de las relaciones de parentesco de cada persona con cada ancestro común $r_{BC} = \Sigma\, p_{AB} \cdot p_{AC}$ donde **A** es uno de los ancestros comunes y la suma es sobre todo **A**.
Dados los árboles genealógicos de dos personas ficticias “coco” y “pepe”, se pide calcular el coeficiente de relación entre ellos. Identifique los subproblemas e implemente funciones para resolverlos.