ICOM2015 – Examen Final A

Diciembre 2015

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 - *Anagramas.*

Un anagrama es una palabra o una frase que se construye a partir de la transposición de las letras de otra palabra o frase.

Por ejemplo:

Salvador Dali -> Avida Dollars
Argentinos -> Ignorantes
Japones -> Esponja

Se solicita implementar las funciones para:

a. Dadas dos frases, decidir si una es un anagrama de la otra. El prototipo de la función deberá ser:

```
int is_anagrama(const char *str1, const char *str2);
```

PROBLEMA 2 – *Imagen panorámica.*

Una imagen panorámica es una combinación de imágenes digitales que se construye al empalmar imágenes individuales tomadas secuencialmente. Las panorámicas se popularizaron con el uso de las cámaras digitales y los smartphones, donde el usuario toma una serie de fotos desplazando la cámara en una o más direcciones.
Un algoritmo trivial para realizar el empalme consiste tomar dos imágenes, la original y la imagen actual, y crear una nueva imagen con un tamaño suficiente para almacenar la panorámica. Finalmente se ubican ambas imágenes adyacentes manteniendo la coherencia con la dirección en la que se movió la cámara, y se sobrescribe la imagen original.

El movimiento de la cámara puede ser bidimensional y por ello al cambiar de dirección es posible que en la nueva panorámica aparezcan zonas donde no se tiene información (ver ejemplo de la figura). Para mantener la topología rectangular de la imagen, estas zonas se representan simplemente como una imagen de pixeles negros.

Original

Izquierda

Abajo

A partir de la siguientes estructuras para representar imágenes en mapa de bits

```
typedef struct {
    unsigned char R, G, B;
} RGB;

typedef struct {
    int ancho, alto;
    RGB **pixel;
} ImagenRGB;
```

Se solicita implementar la siguiente función:

```
typedef enum{
    arriba, abajo, izquierda, derecha,
} Direccion;

typedef struct {      // esta estructura mantiene la posición de la
    int posX, posY;   // ultima imagen pegada, debe inicializarse
} StitchContext;      // con 0,0, después la mantiene updatePano


// Actualiza una imagen panoramica, agregando la imagen nueva y teniendo
// en cuenta la dirección de la nueva imagen
void updatePano(ImagenRGB *panImg, ImagenRGB *nueva,
                Direccion dir, StitchContext *ctx);
```

*Hint: La updatePano debe actualizar ctx, de acuerdo a la posición en donde se pegó la última imagen.

PROBLEMA 3 - Uso de un Trie.

Se desea realizar una aplicación que encuentre la frecuencia de aparición de palabras (es decir la cantidad de veces que aparece cada palabra) en un texto. Para ello se decide utilizar una estructura de trie, como la del pasado parcial, aprovechando la velocidad de búsqueda que se logra con la misma.

Los archivos **`trie.h`** y **`trie.c`** son la solución al problema 1 del pasado parcial de ICOM.

En ese código existe un atributo en cada nodo: **`defineKey`** que puede tomar valor **`TRUE`**/**`FALSE`** indicando si ese nodo define o no una clave

Se solicita:

a. Introducir los cambios necesarios para modificar la forma en que un nodo indica si el mismo define una clave o no. Se solicita reemplazar el atributo **`BOOL defineKey`** por el atributo **`unsigned int numReps`** que indique la cantidad de veces que esa clave se repite, con eso **`numReps == 0`** indicaría que el nodo no es clave, y un **`numReps > 0`** indicaría que el nodo es clave y se repite **`numReps`** veces.
b. Utilizando los servicios del **`trie`** resultante, procesar el contenido del archivo **`libro.txt`** para registrar y contar las distintas palabras que en el archivo aparecen. El archivo pose una única palabra por línea.
c. Implementar un nuevo servicio para listar las claves en forma alfabética e indicando la cantidad de veces que cada palabra aparece. El prototipo deberá ser:

```
void trieList(Trie_t *pTrie);
```