Pregunta 1

Parte a.- Programe la siguiente función:

```
typedef unsigned int uint;
uint borrar_bits(uint x, uint pat, int len);
```

Esta función debe retornar el resultado de reemplazar en *x* todas las apariciones del patrón *pat* de *len* bits por ceros. En los siguientes ejemplos de uso la notación *0b...* expresa números en base 2, aunque no es parte del estándar de C.

```
borrar_bits(0b10001000100101, 0b101, 1) es 0
borrar_bits(0b11101101001, 0b10, 2) es 0b11001001
borrar_bits(0b11101101001, 0b101, 3) es 0b11000001
```

Restricción: Ud. no puede usar los operadores de multiplicación, división o módulo (* / %). Use los operadores de bits eficientemente.

Parte b.- Programe la función:

```
void elim_str(char *str, char *pat);
```

Esta función elimina del string *str* todas las apariciones del patrón *pat*. Ejemplo:

```
char s1[]= "las palas van"; // 13 caracteres
elim_str(s1, "las"); // s1 es " pa van" (7 caracteres)
char s2[]= "11101101";
elim_str(s2, "10"); // s2 es "11101"
```

Restricciones: Ud. no puede usar el operador de subindicación [ ], ni su equivalente **(p+i)*. Para recorrer el string use aritmética de punteros. No puede pedir memoria adicional ni declarar otros arreglos de caracteres. Use múltiples punteros para direccionar distintas partes del string.

Pregunta 2

Programe la función *reemplazarNodoK* con el siguiente encabezado:

```
typedef struct nodo {
  char c;
  struct nodo *izq, *der;
} Nodo;
int reemplazarNodoK(Nodo **pa, int k, Nodo *b);
```

Sea *a= *pa*. Esta función reemplaza el *k*-ésimo nodo del árbol *a* por el nodo *b*. El *k*-ésimo nodo de *a* es el *k*-ésimo nodo al enumerar los nodos de *a* en orden[1]. Por ejemplo en la figura 1, *r* es el nodo 1, *s* el 2, *t* el 3, *u* el 4. Esta función retorna *k* si se hizo el reemplazo, es decir cuando el árbol *a* tenía al menos *k* nodos. Si no, entrega el número de nodos encontrados en *a*, que será inferior a *k*. Las siguientes figuras sirven para explicar algunos ejemplos de uso. Los punteros *a* y *b* son de tipo *Nodo**.

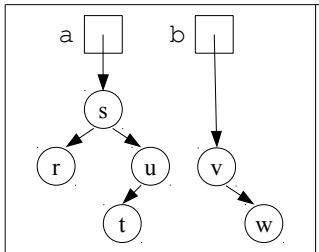

Figura 1

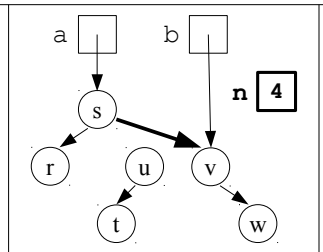

Figura 2

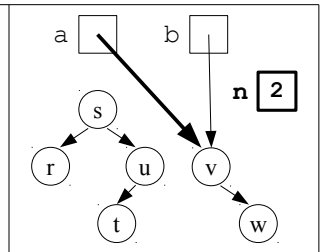

Figura 3

La figura 2 se obtiene cuando a partir de la figura 1 se llama:

```
int n= reemplazarNodoK(&a, 4, b);
```

Se reemplazó *u* por *v* y la función retornó 4. La figura 3 se obtiene cuando a partir de la misma figura 1 se llama:

```
int n= reemplazarNodoK(&a, 2, b);
```

Acá se reemplazó la raíz *s* del árbol por *v*. Por eso se requiere que el puntero a la raíz del árbol se pase por referencia (*&a*). Por último si a partir de la figura 1 se intentara reemplazar el quinto nodo de *a* que no existe, no se haría ningún reemplazo y la función retornaría 4.

Ayuda: Sea *a= *pa*, el árbol en donde se hará el reemplazo. Pruebe en orden los siguientes 4 casos. (1) Si *a* es nulo, no se puede hacer ningún reemplazo, retorne 0. Note que *pa* es siempre distinto de nulo. (2) Intente hacer recursivamente el reemplazo en el subárbol izquierdo de *a*. Si tiene éxito retorne *k*. (3) Si el número de nodos del subárbol izquierdo es *k-1*, reemplace el nodo *a* y retorne *k*. (4) Intente hacer recursivamente el reemplazo en el subárbol derecho de *a*, etc.

1 Por ejemplo si el subárbol izquierdo de *t* posee *i* nodos, estos nodos se enumeran recursivamente de *1* a *i* y la raíz de *t* será el nodo *i+1*. Luego se enumeran recursivamente los nodos del subárbol derecho de *t* a partir de *i+2*.