Pregunta 1

Programe la función elimCifra con el siguiente encabezado:

```
typedef unsigned int uint;
uint elimCifra(uint x, int h);
```

El parametro *x* corresponde a un entero de 32 bits que deben ser visto como 8 cifras hexadecimales de 4 bits cada una. La función *elimCifra* debe entregar el resultado de eliminar todas las apariciones de la cifra *h* en *x*. Ejemplos de uso:

```
uint rc1= elimCifra(0x3a0ff0a3, 3);   // rc1 es 0x00a0ff0a
uint rc2= elimCifra(0x3a0ff0a4, 0);   // rc2 es 0x003affa4
uint rc3= elimCifra(0x3a0fe0b3, 0xf); // rc3 es 0x03a0e0b3
uint rc4= elimCifra(0x3a0fe0b3, 0xd); // rc3 es 0x3a0fe0b3
```

Restricciones: No use los operadores de multiplicación, división o módulo (* / %). Use eficientemente los operadores de bits, sumas y restas.

Pregunta 2

Programe la siguiente función:

```
void elimEspacios(char *s);
```

Esta función reemplaza múltiples espacios contiguos en el string *s* por un solo espacio. Ejemplo de uso:

```
char s[]= "hola     que     tal";
elimEspacios(s); //s es "hola que tal"
```

Por simplicidad, el string *s* no tiene espacios en blanco al principio o al final.

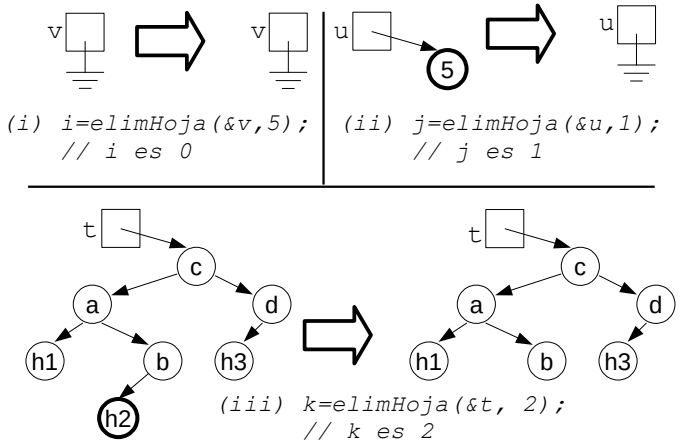
***Restricciones*:** No puede invocar otras funciones predefinidas como *strlen*, *strcpy*, *etc*. No use el operador de subindicación de arreglos [ ] ni su equivalente **(p+i)*, use aritmética de punteros como *p*++ o *p+i*. No puede pedir memoria adicional con *malloc* ni declarar arreglos. Sí deberá declarar punteros adicionales.

Ayuda: Use un primer puntero para recorrer el string *s* y un segundo puntero hacia el mismo string *s*, pero en donde va copiando los caracteres que necesitan ser preservados en el string. No olvide terminar el string resultante.

Pregunta 3

```
typedef struct nodo {
  char str[8];
  struct nodo *izq, *der;
} Nodo;
int elimHoja(Nodo **pa,
             int k);
```

Programe la función *elimHoja*, con el encabezado del cuadro de la derecha. Esta función debe eliminar del árbol binario **pa* la *k*-ésima hoja, con $k \geq 1$, *liberando la memoria ocupada por ese nodo*. Debe retornar la cantidad de hojas visitadas: *k* si se logró eliminar la *k*-ésima hoja, o la cantidad de hojas que tenía el árbol, si eran menos que *k*. Una hoja es un nodo en que tanto el subárbol izquierdo como el subárbol derecho son vacíos. La hojas se enumeran desde 1 al recorrer el árbol en orden (primero se recorre el subárbol izquierdo, luego la raíz y finalmente se recorre el subárbol derecho). Los siguientes son ejemplos de uso. En la figura el tipo de *v*, *u* y *t* es *Nodo** y el de *i*, *j* y *k* es *int*.



Ayuda: Los casos (i) y (ii) corresponden a los 2 casos de término de la recursión, y (iii) es el caso recursivo. Al llamar recursivamente a *elimHoja* con el subárbol izquierdo, si el valor retornado es *k* la eliminación tuvo éxito y retorne *k*. Si es menor que *k*, no se eliminó la hoja y el valor retornado es la cantidad de hojas encontradas en el subárbol izquierdo. Tendrá que llamar recursivamente a *elimHoja* con el subárbol derecho, descontando de *k* las hojas del subárbol izquierdo. Al retornar, recuerde contabilizar también las hojas del subárbol izquierdo. Por ejemplo si en (iii) se pide eliminar la 3^{era} hoja, se eliminará el nodo *h3* y se retornará 3. Si se pide eliminar la 4ta hoja, no se elimina nada y se retorna 3.