Pregunta 1

Programe la función *elimDup* con el siguiente encabezado:

```
typedef unsigned int uint32_t;
uint32_t elimDup(uint32_t x);
```

ElimDup recibe un entero *x* que debe ser visto como 8 cifras hexadecimales (de 4 bits cada cifra) y debe retornar el resultado de eliminar las cifras consecutivas duplicadas, agregando ceros a la izquierda. Ejemplo: *elimDup(0x5c**cc**0**0**5c)* debe entregar 0x**000**5c05c. Las cifras en subrayadas fueron eliminadas porque aparecen duplicadas consecutivamente y se agregaron 3 ceros a la izquierda. Observe que *c* aparece 4 veces pero la cuarta aparición no se elimina porque no es consecutiva con otra *c*.

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

Pregunta 2

Programe la función *comprimir* con el siguiente encabezado:

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

Comprimir reemplaza todas las apariciones de caracteres consecutivamente repetidos por un caracter numérico ('2' a '9') que indica el número de apariciones seguido del caracter consecutivamente repetido. El string de entrada no contiene caracteres numéricos. Por simplicidad considere que **ningún caracter se repite más de 9 veces**. Ejemplo:

```
char s[ ]= "CCGGAAATCAAAA";
comprimir(s); // s es "2C2G3ATC4A"
```

Restricciones: Ud. no puede usar el operador de subindicación [ ], ni su equivalente **(p+i)*. Use aritmética de punteros como *p++* o *p+i*. No puede usar *malloc* ni declarar arreglos. Sí necesitará declarar punteros adicionales.

Pregunta 3

Programe la función *balancear* con el encabezado del cuadro de arriba en la siguiente columna. *Balancear* recibe en *h* una altura y en **pa* un árbol binario de búsqueda (*abb) desbalanceado al extremo*, es decir en todos sus nodos el subárbol izquierdo es nulo, como el de la tarea 3. Debe extraer de **pa* un *abb* balanceado de altura máxima *h* con máximo 2^h - 1 nodos y retornarlo. En **pa* debe quedar el *abb* desbalanceado al extremo con todos los nodos que estaban inicialmente en **pa* pero que quedaron fuera del *abb* retornado. El *abb* puede quedar con menos hojas en el subárbol izquierdo que en el derecho, o ser de altura inferior a *h* si no hay suficientes nodos en **pa*. Recuerde que al ser un *abb*, sí es relevante el orden en que quedan los nodos tanto en **pa* como en el *abb* retornado, lo que logrará con la siguiente metodología.

```
typedef struct nodo {
  int x;
  struct nodo *izq, *der;
} Nodo;
Nodo *balancear(Nodo **pa,
                int h);
```

Metodología obligatoria: Detenga la recursividad cuando **pa* es NULL o h==0. Para el caso recursivo, use *balancear* para extraer de **pa* un *abb IZ* de altura *h*-1. A continuación, si no quedan nodos en **pa* retorne solo *IZ*. De lo contrario, sea *a* el primer nodo que quedó en **pa*. Extraiga *a* de **pa* y use nuevamente balancear para extraer de **pa* un *abb DE* de altura *h*-1. Retorne un *abb* con raíz *a* y subárboles *IZ* y *DE*.

En los siguientes ejemplos de uso el tipo de *a* y *b* es Nodo*: