Pregunta 1

Parte a.- Programe la siguiente función:

```
int contar(unsigned long b, unsigned int p, int n);
```

Esta función cuenta el número de apariciones del patrón *p* de *n* bits en el número *b*. En los siguientes ejemplos de uso la notación *0b...* expresa números en base 2. Esta notación no es parte del estándar de C, pero se emplea acá para facilitar la comprensión de los ejemplos.

```
int f= contar(0b10010011, 0b1, 1)  ; // 4
int g= contar(0b1011101, 0b101, 3); // 2
int h= contar(0b0111, 0b11, 2);     // 1
```

El patrón se debe buscar desde los bits menos significativos de *b* hacia los más significativos. Si una secuencia de bits coincide con el patrón, ninguno de esos bits puede formar parte de otra secuencia coincidente.

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

Parte b.- Programe la siguiente función:

```
void eliminar_duplicados(char *str);
```

Esta función elimina del string *str* todos los caracteres que aparezcan duplicados. Ejemplo de uso:

```
char *ro= "aabcbddd1b221"; // solo lectura
char rw[strlen(ro)+1];     // lectura y escritura
strcpy(rw, ro);            // copia el string ro en rw
eliminar_duplicados(rw);   // la función pedida
printf("%s\n", rw);        // abcd12
```

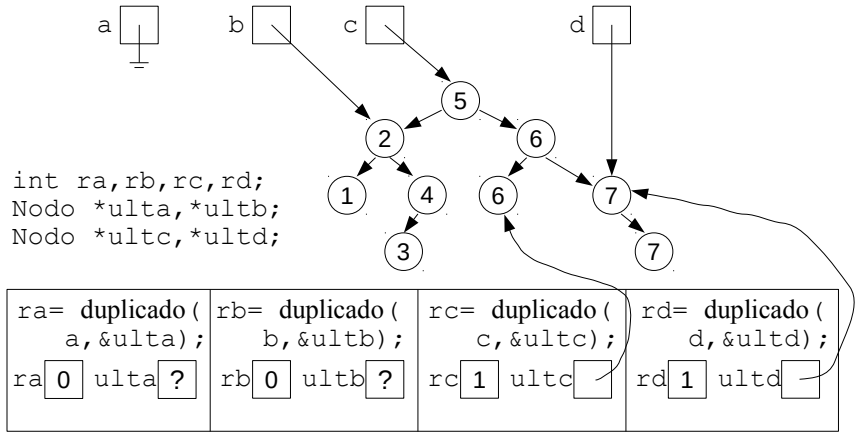
Restricciones: Ud. no puede usar el operador de subindicación [ ], ni su equivalente **(p+i)*. Para recorrer el string use el operador ++. Use múltiples punteros para direccionar distintas partes del string. No tiene sentido usar *malloc*.

Pregunta 2

Programe la función *duplicado* que entrega el primer nodo que contenga una etiqueta duplicada en un árbol de búsqueda binaria al recorrerlo en orden (recuerde los 3 tipos de recorridos: en orden, pre-orden y post-orden). El tipo de un nodo y el encabezado de la función pedida son los siguientes:

```
typedef struct nodo {
  int etiqueta;
  struct nodo *izq, *der;
} Nodo;
int duplicado(Nodo *a, Nodo **pult);
```

La función *duplicado* debe retornar 1 (verdadero) si se encontró un nodo duplicado y en tal caso **pult* es la dirección del primer nodo duplicado. De otra forma la función retorna 0 (falso) y el valor de **pult* no está especificado. En la siguiente figura se muestran los resultados de llamar la función *duplicado* para los árboles *a*, *b*, *c* y *d*:



A modo de ayuda programe *duplicado* de esta manera:

```
int duplicado(Nodo *a, Nodo **pult) {
  *pult= NULL; // para indicar que el primer nodo no tiene nodo previo
  return dup_aux(a, pult);
}
```

Programe *dup_aux* haciendo un recorrido tradicional del árbol en orden. Haga que **pult* apunte al nodo previamente visitado. Primero considere el caso del árbol vacío; visite después recursivamente el subárbol izquierdo; más tarde visite el nodo actual comparando su etiqueta con la del nodo previo (la única excepción es el primer nodo, en cuyo caso **pult==NULL);* y finalmente visite recursivamente el subárbol derecho.

Observe que para el caso del árbol *c*, el primer nodo por visitar es el nodo etiquetado con 1, no la raíz etiquetada con 5. Además primero se visita la hoja etiquetada con 6 y luego su ancestro etiquetado con 6. Por eso el primer duplicado es la hoja y no su ancestro. En cambio para el caso del árbol *d*, primero se visita la raíz etiquetada con 7 y luego la hoja etiquetada con 7, y por eso el primer duplicado es la raíz y no la hoja.