Pregunta 1

Programe la siguiente función:

```
void elimChar(char *str, char *c);
```

Esta función debe eliminar del string *str* todas las apariciones del caracter *c*. Ejemplo de uso:

```
char s[ ] = "  hola que   tala";
elimChar(s, 'a'); // s es "  hol que   tl"
```

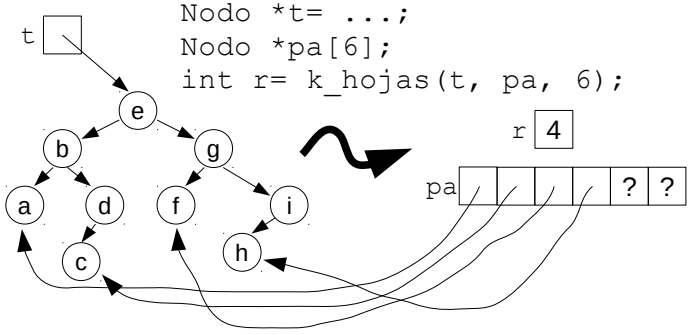
Restricciones: No use el operador de subindicación de arreglos *[ ]* ni su equivalente **(p+i),* use aritmética de punteros. No use *malloc*.

Pregunta 2

Se define una hoja en un árbol de búsqueda binaria como un nodo que no tiene ningún hijo. Escriba la función *k_hojas* con el siguiente encabezado:

```
typedef struct nodo {
  char *s;
  struct nodo *izq, *der;
} Nodo;
int k_hojas(Nodo *t, Nodo **pa, int k);
```

Esta función debe recorrer en orden el árbol *t* y guardar en el arreglo *pa* las direcciones de las *k* primeras hojas de *t*. Debe retornar el número efectivo de hojas que se depositaron en *pa*, que no debe exceder *k*. En el siguiente ejemplo de uso el árbol tiene 4 hojas (a, c, f y h) y por lo tanto se guardan sus direcciones en el arreglo *pa* y se retorna 4:



Metodología obligatoria: Si *t* es el árbol nulo, termine retornando 0. Si no, si *t* es una hoja, guárdela en *pa[0]* y termine retornando 1. Si no, obtenga recursivamente hasta *k* hojas del subárbol izquierdo guardándolas en el arreglo *pa*. El valor retornado le dirá cuantas hojas se encontraron en el subárbol izquierdo. Quedaron guardadas al principio del arreglo *pa*. Si logró obtener las *k* hojas, termine retornando *k*. Si no, obtenga recursivamente las hojas restantes del subárbol derecho, para llegar hasta el máximo de *k* hojas. Ud. debe discurrir cuáles serán los parámetros *pa* y *k* que debe pasar en esta segunda invocación recursiva y qué debe retornar la función. Note que los elementos del arreglo *pa* son punteros a nodos.

Observación: Ud. debe ser capaz de programar en C este algoritmo que está dado casi por completo en palabras.

Pregunta 3

La siguiente es una solución incorrecta e ineficiente del problema de los lectores/escritores. Pero sí funciona el 99,99% de las veces y además evita la hambruna. Reprograme este código de manera que sea siempre correcto, eficiente y que preserve la ausencia de hambruna. No cambie ni borre ningún carácter. Solo agregue código.

```
int readers= 0, disp= 0, serial= 0;
```

```
void enterRead() {
  int myNum= serial++;
  while (myNum!=disp)
    ;
  readers++; disp++;
}
void exitRead() {
  readers--;
}
```

```
void enterWrite() {
  int myNum= serial++;
  while (readers>0 || myNum!=disp)
    ;
}
void exitWrite() {
  disp++;
}
```

Pregunta 4

Programe la función *maxArreglo* que entrega el máximo valor almacenado en el arreglo *a* de *n* enteros. El cálculo debe hacerse en paralelo usando *p* procesos pesados (además del proceso que invoca la función). Invoque *fork* para crear los procesos y use *pipes* para que los procesos hijos entreguen su máximo parcial al proceso padre. El encabezado de la función es:

```
int maxArreglo(int a[], int n, int p);
```

Por simplicidad puede suponer que *n* es múltiplo de *p* y que la constante MININT almacena el entero más negativo.