Pregunta 1

La función pálindrome, cuyo encabezado se muestra en el cuadro de la derecha, entrega 1 si la secuencia de 32 bits del entero sin signo *x* es palíndrome en hexadecimal, es decir que se lee igual de iquierda a derecha como de derecha a izquierda. Entrega 0 si no es palíndrome. Ejemplos de uso:

```
typedef unsigned int uint;
int palindrome(uint x);
```

```
int rc1= palindrome(0x3a0ff0a3); // rc1 es 1, es palíndrome
int rc2= palindrome(0x3a0ff0a4); // rc2 es 0, no es palíndrome
int rc3= palindrome(0x3a0fe0b3); // rc3 es 0
int rc4= palindrome(0x11); // rc4 es 0, 0x11 ≡ 0x00000011
```

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

Pregunta 2

Programe la función: *void insertar(char c, char *s);*

Esta función inserta el caracter *c* al comienzo del string que se inicia en la dirección contenida en *s*. Considere que hay espacio suficiente en *s* para agregar un caracter. Ejemplos de uso:

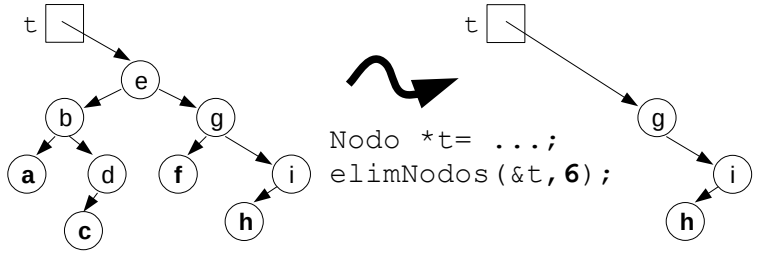
```
char s[5]= "ola";
insertar('h', s);     // s es el string "hola"
char s[20]= "oy inseguro, o no?";
insertar('S', s);     // s es el string "Soy inseguro, o no?"
```

***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. No puede pedir memoria adicional con *malloc* ni declarar arreglos. Sí deberá declarar punteros adicionales.

Pregunta 3

Programe la función: *int elimNodos(Nodo **pa, int k);*

Esta función debe eliminar del árbol **pa* los primeros *k* nodos al recorrer el árbol en orden, liberando la memoria ocupada por esos nodos. Debe retornar la cantidad efectiva de nodos eliminados: *k* si el árbol tenía al menos *k* nodos o el tamaño del árbol si tenía menos de *k* nodos. La definición de *Nodo* es la usual con campos *izq* y *der* para los subárboles izquierdo y derecho. En el siguiente ejemplo de uso se pide eliminar los primeros 6 nodos.



En el recorrido en orden de un árbol se recorre: primero el subárbol izquierdo recursivamente, luego la raíz y finalmente el subárbol derecho recursivamente.

Metodología obligatoria: Sea *a=*pa*. Si *a* es el árbol vacío o *k* es 0, termine retornando 0. Si no elimine recursivamente *k* nodos del subárbol izquierdo de *a*. Sea *k'* el valor retornado. Si *k'=k*, termine retornando *k*. Si no, necesariamente *k'<k* y debe liberar el espacio ocupado por *a*. Además elimine recursivamente *k-k'-1* nodos del subárbol derecho, dejando el árbol resultante en **pa*. Finalmente calcule y retorne la cantidad efectiva de nodos eliminados en todo el árbol.

Pregunta 4

Programe la función: *int main(int argc, char *argv[ ])* para el comando *./mediana*. Este comando recibe como parámetro el nombre de un archivo y despliega en la salida estándar la primera línea **completa** que se encuentra a partir de la mitad del archivo (en bytes). La líneas son de largo variable pero por simplicidad considere que tienen a lo más 80 caracteres y terminan con el caracter adicional ***\n***, que ocupa un solo byte. Por ejemplo el archivo *nombres.txt* de tamaño 39 bytes contiene:

```
pedro\njuan\ndiego\nmaría\nximena\nveronica\n
```

La **r** en negritas señala la mitad del archivo (posición 19).

La invocación del comando: `./mediana nombres.txt`

debe entregar en la salida estándar: `ximena\n`

No debe entregar `ria\n` porque esa línea no está completa.

Metodología obligatoria: Calcule con *fseek* y *ftell* el tamaño *T* del archivo. Use *fseek* para posicionarse en la posición *T/2-1* y lea 2 líneas. Despliegue en la salida estándar la segunda línea.