Pregunta 1

La función *posicion*, cuyo encabezado se muestra en el cuadro de la derecha, entrega la posición de la cifra hexadecimal *c* en el número *x*, de 8 cifras hexadecimales (32 bits en total), o -1 si *c* no aparece en *x*. Ejemplos de uso:

typedef unsigned int ***uint****;*
int ***posicion****(uint x, uint c);*

```
int rc1= posicion(0xa35fc531, 0x1);      // rc1 es 0
int rc2= posicion(0xa35fc531, 0x5);      // rc2 es 2
int rc3= posicion(0xa35fc531, 0xf);      // rc3 es 4
int rc4= posicion(0xa35fc531, 0x0);      // rc4 es -1
```

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

Pregunta 2

Programe la función *desplazarDer* con el siguiente encabezado:

```
void desplazarDer(char *s, int n);
```

Esta función desplaza el string *s* en *n* caracteres hacia la derecha agregando *n* espacios en blanco al principio y eliminando los últimos *n* caracteres. Por simplicidad puede considerar que *s* tiene al menos *n* caracteres. Está permitido usar *strlen*. Este es un ejemplo de uso:

```
char s[]= "quien vive";
desplazarDer(s, 3); // s es el string "    quien v" con 3 espacios al inicio
```

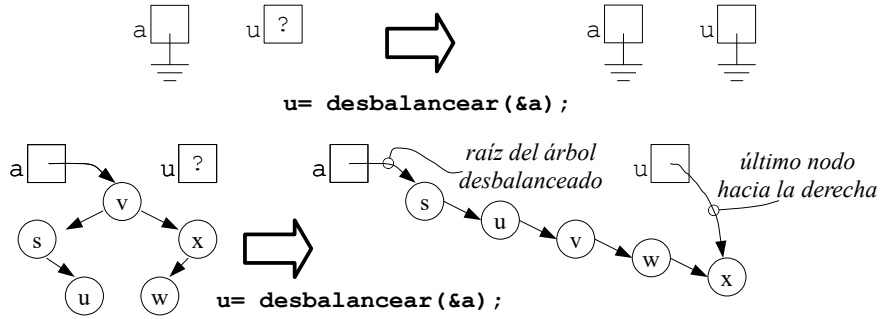
Restricciones: No use el operador de subindicación de arreglos [ ] ni su equivalente **(p+i)*, use aritmética de punteros. No puede pedir memoria con *malloc* ni declarar arreglos.

Ayuda: Si *L* es el largo del string, Ud. *debe* recorrer los caracteres desde la posición *L-n-1* hacia la posición 0, copiándolos en su posición de destino. Si recorre en orden inverso sobreescribirá incorrectamente caracteres que no ha copiado todavía. No olvide colocar *n* espacios al inicio del string.

Pregunta 3

Programe la función: *Nodo *desbalancear(Nodo **pa);*

La estructura *Nodo* es la usual con campos *izq* y *der* para los subárboles izquierdo y derecho. La función *desbalancear* recibe en **pa* un árbol binario de búsqueda (ABB) y entrega en el mismo **pa* un árbol *equivalente* pero desbalanceado al extremo a la derecha, es decir un árbol en el que todos los subárboles izquierdos son NULL. Además retorna la dirección del nodo de más a la derecha del árbol resultante. La función debe reutilizar los mismos nodos del árbol original. No se puede pedir memoria con *malloc*. La figura muestra 2 ejemplos de uso. Las variables *a* y *u* son de tipo Nodo *.



Metodología obligatoria: Sea *a=*pa*. Considere el caso en que *a* no es NULL, el subárbol izquierdo de *a* no es NULL y el derecho sí es NULL. Desbalancee recursivamente el subárbol izquierdo de *a* y obtendrá *izq*, el subárbol izquierdo desbalanceado, y *ultIzq*, el último nodo de ese subárbol. Enlace el subárbol derecho de *ultIzq* con *a*, los subárboles izquierdo y derecho de *a* con NULL. El nuevo valor de **pa* debe ser *izq* y retorne *a*. Proceda de manera similar con los otros casos.

Pregunta 4

Programe la función: *int main(int argc, char *argv[ ])* para el comando *./palindrome*. Este comando recibe como parámetro el nombre de un archivo de texto y determina si el archivo es palíndrome, es decir la líneas del archivo se leen igual de arriba hacia abajo como de abajo hacia arriba. Las líneas son de largo variable, pero no de más de 81 caracteres (con el **\n**). El siguiente es un ejemplo de uso con el archivo *nom.txt* del cuadro de la derecha:

```
pedro
juan
diego
juan
pedro
```

nom.txt

```
./palindrome nom.txt
```

debe entregar en la salida estándar: `palindrome`

Si el archivo no es palíndrome debe mostrar:

```
no es palindrome
```

Metodología obligatoria: Calcule con *fseek* y *ftell* el tamaño *T* del archivo. Inicialice el *inicio* del archivo en *0* y el *fin* en *T*. Use *fseek* para posicionarse en *inicio*; lea una línea y determine su largo *L*; use *fseek* para posicionarse en *fin-L*; lea la línea y compruebe que es igual a la primera línea. Ahora el *inicio* es *L* y el *fin T-L*. Proceda de la misma manera para comprobar el siguiente par de líneas. Siga así hasta que *inicio≥fin*.