CC3301 Programación de Software de Sistemas – Control 1
Semestre Primavera 2025 – Profs. Mateu/Ibarra/Urrea

Pregunta 1

```
typedef unsigned long long Decimal;
typedef unsigned long long int ullong;
ullong decimal2int(Decimal x);
```

El tipo *Decimal* es un entero sin signo de 64 bits que almacena números en *formato decimal* de hasta 16 dígitos. Esto significa que cada dígito de *x* se representa con 4 bits de *x*. Por ejemplo el número 125 se representa en formato decimal como 0x125 (en binario: 1 0010 0101). Note que 0x125 representa en hexadecimal el número 293 ($1*16^2+2*16+5$), no 125. **Programe** la función *decimal2int* con el encabezado del cuadro de arriba, que convierte un entero en formato decimal al entero que representa. Por ejemplo *decimal2int*(0x125) debe entregar 125 y *decimal2int*(0x98023) entrega el entero 98023.

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

Ayuda: Si las cifras decimales de *x* son $d_{15}\ d_{14}\ d_{13}\ \ldots\ d_1\ d_0$ su valor entero se calcula como:

$$((...((d_{15}*10+d_{14})*10+d_{13})*10+\ \ldots)*10+d_1)*10+d_0$$

Multiplicar *z* por 10 sin usar * es fácil notando que z*10 = (z<<3)+(z<<1). El valor de d_{15} es x >> 60, el valor de d_{14} son los 4 bits menos significativos de *x* >> 56, el valor de d_{13} son los 4 bits menos significativos de *x* >> 52, etc.

Pregunta 2

Programe la función *invertirVocales* declarada como:

*void invertirVocales(char *str);*

Esta función recibe un string *str* e invierte solo las vocales que este contenga. Por simplicidad suponga que solo aparecen letras minúsculas. Ejemplos:

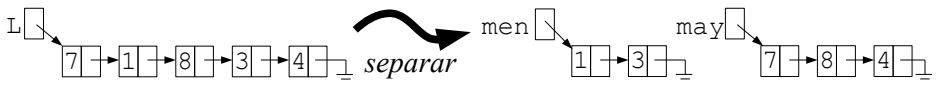
```
char s1[]="cfd"; invertirVocales(s1);        // s1 es "cfd" porque no tiene vocales
char s2[]="abcde"; invertirVocales(s2);      // s2 es "ebcda" porque se invierten a y e
char s3[]="icecream"; invertirVocales(s3);   // s3 es "acecreim", se invierten i y a, y e y e.
char s4[]="invertir vocales"; invertirVocales(s4);  // s4 es "envartor vicelis"
```

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

Pregunta 3

Programe la función *separar* con el encabezado del cuadro de la derecha. Esta función recibe en *L* una lista simplemente enlazada desordenada en donde cada nodo almacena un entero. Ud. debe desarmar la lista *L* de tal forma que en **pmen* queden los nodos que almacenan enteros menores que *z* y en **pmay* queden los nodos que almacenan enteros mayores o iguales que *z*. En el cuadro también se muestra un ejemplo de uso, en donde la lista *L* ha sido creada con 5 nodos que almacenan 7, 1, 8, 3 y 4. La figura muestra a la izquierda la lista *L* original y a la derecha las 2 listas resultantes de invocar separar.

```
typedef struct nodo {
  int x;
  struct nodo *prox;
} Nodo;
void separar(Nodo *L, int z,
  Nodo **pmen, Nodo **pmay);
Nodo *L= ...; // 7 1 8 3 4
Nodo *men, *may;
separar(L, 4, &men, &may);
```



Restricciones: No puede usar *malloc*. Debe reutilizar los nodos que recibe en la lista *L*. Los nodos en **pmen* y **pmay* deben seguir el mismo orden en que aparecían en *L*. Por claridad Ud. **debe usar recursividad**.