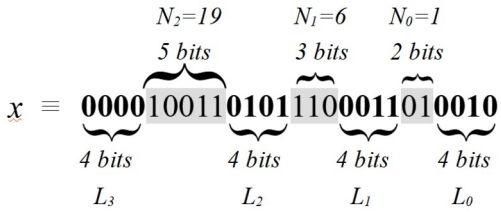
Pregunta 1

Programe la función *suma* con el siguiente encabezado:

```
typedef unsigned long long uint64;
uint64 suma(uint64 x);
```

El parámetro *x* es de 64 bits y almacena una secuencia de números de largo variable. La función *suma* debe entregar la suma de la secuencia. Por ejemplo, en la siguiente figura la secuencia en *x* es (19,6,1) y por lo tanto el valor retornado por *suma* debe ser 19+6+1≡26.



La secuencia en *x* se decodifica así: Los 4 bits de más a la derecha de *x* (0b**0010**≡2) corresponden al largo L_0 en bits del primer número. Continuando hacia la izquierda viene el primer número N_0 codificado en L_0 bits (0b01≡1). Siguiendo hacia la izquierda viene el tamaño L_1 del segundo número (0b**0011**≡3) y el segundo número N_1 codificado en L_1 bits (0b110≡1). Finalmente viene L_2 (0b**0101**) y N_2 (0b10011≡19). La secuencia termina porque el último largo L_3 es 0. En general la secuencia se lee de derecha a izquierda, por cada número vienen primero 4 bits con el largo ***L*** y luego el número codificado en ***L*** bits. La secuencia termina con un largo 0.

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

Pregunta 2

Programe la función *reemplazo* con el siguiente encabezado:

```
char *reemplazo(char *s, char c, char *pal);
```

Esta función entrega un nuevo string resultante de reemplazar en *s* todas las ocurrencias del caracter *c* por el string *pal*. Note que *c* no es un string, es un caracter. Ejemplo:

```
char *res= reemplazo("hola que tal", 'a', "xyz");
```

El string resultante es "hol**xyz** que t**xyz**l".

Restricciones: Ud. no puede usar el operador de subindicación [ ], ni su equivalente **(p+i)*. Use aritmética de punteros como *p*++ o *p*+*i*. Debe usar *malloc* para pedir la mínima cantidad de memoria necesaria para almacenar el resultado. No puede modificar el string *s*.

Ayuda: Recorra una vez el string *s* para calcular la cantidad de memoria necesaria para el resultado. Recorra una segunda vez *s* para copiar o reemplazar en el string resultante. Puede usar funciones como *strlen*, *strcpy*, etc.

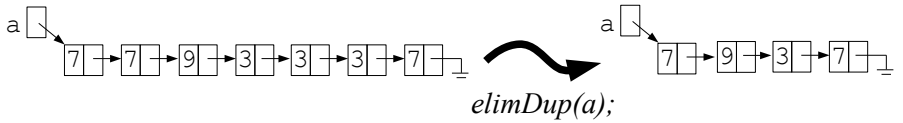
Pregunta 3

Programe la función *elimDup* con el siguiente encabezado:

```
typedef struct nodo {
  int x;
  struct nodo *prox;
} Nodo;
void elimDup(Nodo *L);
```

La función *elimDup* recibe en *L* una lista enlazada con números enteros. Debe eliminar de la lista todos los nodos consecutivos repetidos dejando una sola copia. Debe liberar la memoria ocupada por los nodos eliminados.

Por ejemplo, en la siguiente figura, *a* es de tipo *Nodo** y contiene la dirección de una lista enlazada con elementos 7 7 9 3 3 3 7. Al invocar *elimDup(a)*, la lista queda formada por 7 9 3 7.



Observe que no se elimina el último 7, porque no está duplicado consecutivamente.