Nº Orden	Apellido y nombre	L.U.	Cantidad de hojas

Organización del Computador 2
Recuperatorio del Primer Parcial – 11/07/17

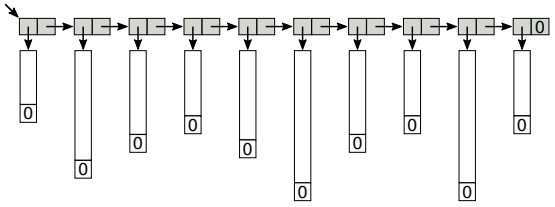
1 (40)	2 (40)	3 (20)	

Normas generales

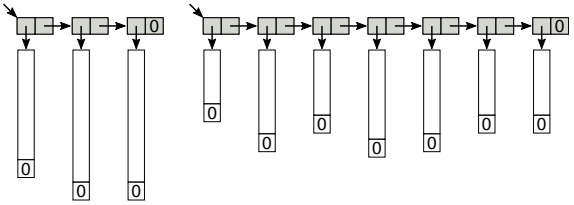
- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

Ej. 1. (40 puntos)

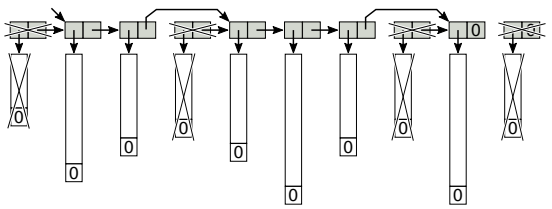
Considerar una lista simplemente encadenada que contiene punteros a cadenas de caracteres terminadas en cero (*strings*). La estructura de cada nodo es `struct node { char* text, node* next}`.



(20p) a. Escribir en ASM la función `void l_separarMaxMin(node** lista, int size, node** listaMax, node** listaMin)`, que dada una lista de *strings*, separa sus nodos en dos listas. La lista `listaMin` debe contener las strings de longitud menor o igual a `size`, mientras que la lista `listaMax` contendrá los mayores. Esta función no debe crear ni borrar nodos.



(20p) b. Escribir en ASM la función `void l_borrarMin(node** lista, int size)`, que dada una lista de *strings*, borra todos los nodos que contienen strings de menor longitud al valor dado por *size*.



Nota: La longitud de una *string* esta dada por la cantidad de caracteres sin incluir el cero.

Ej. 2. (40 puntos)

Se tiene una matriz cuadrada de `n`×`n` de números *float*, con `n` mayor que 10. Escribir las siguientes funciones en ASM:

(10p) 1. `void absoluto(float* m, uint32_t n)` recorre la matriz y remplaza todos los valores de la misma por su absoluto.

(10p) 2. `float maximo(float* m, uint32_t n)` recorre la matriz y obtiene el máximo valor de la misma.

(20p) 3. `uint8_t* escalar(float* m, uint32_t n)` aplica a cada valor de la matriz la siguiente formula: $255 \cdot \frac{X-min}{max-min}$, donde max es el máximo valor de la matriz y min el mínimo. El resultado es almacenado como un valor entero sin signo de 8 bytes. Considerar que se tiene implementada la función del ejercicio anterior y su contraparte para obtener el mínimo.

Nota: recordar la codificación de un número *float*, o *binary32* para el estándar IEEE 754-2008, o *single* para el estándar IEEE 754-1985.

bit 31	bits 30 a 23	bits 22 a 0
signo	exponente	mantisa

Ej. 3. (20 puntos)

Se tiene la siguiente función en C:

```
int read_buffer(char * src) {
  char buffer[20];
  strcpy(buffer, src);
  return 0;
}
```

Donde `strcpy` copia el contenido byte a byte de `src` en `buffer`, hasta encontrar un byte 0 que también copia.

No podemos modificar la función `read_buffer`, pero sí podemos llamarla.

(5p) a. Implementar en ASM la función `read_buffer`. Considerar que `strcpy` esta implementada.

(5p) b. Dibujar la pila antes de llamar a `strcpy`, considerar una posible implementación en ASM.

(10p) c. ¿De qué manera podría hacer que `read_buffer` ejecute código en la dirección `0x12345678`?