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

Organización del Computador 2
Recuperatorio del Primer Parcial – 26/11/2019

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.

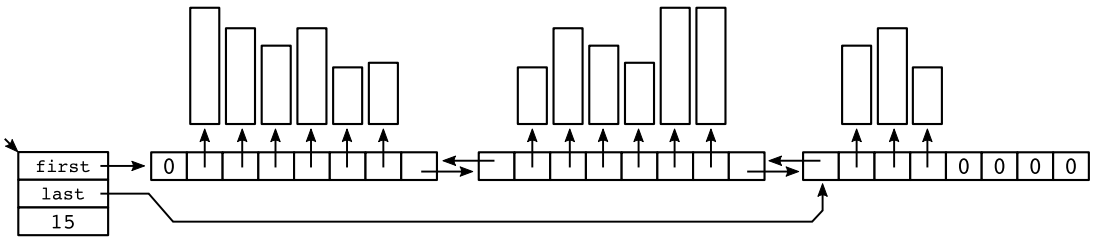
Ej. 1. (40 puntos)

Sea una estructura de datos denominada `veclis`:

```
typedef struct s_veclis {
    nodeVeclist_t* first,
    nodeVeclist_t* last,
    uint32_t       size
} veclis_t;
```

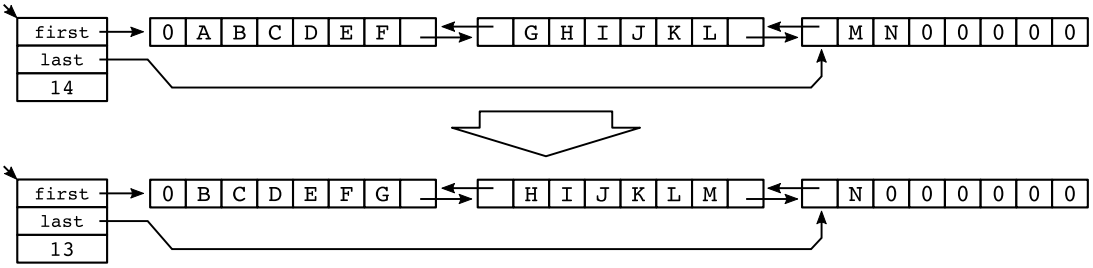
```
typedef struct s_nodeVeclis {
    nodeVeclist_t* prev,
    void*          data[6],
    nodeVeclist_t* next
} nodeVeclis_t;
```

Esta estructura permite almacenar un vector de longitud variable en tramos de 6 elementos. El campo `size` indica la cantidad de datos almacenados en total. Mientras que `first` y `last` apuntan al primer y último nodo. Ejemplo:

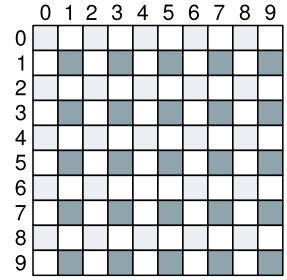


(20p) a. Implementar la función `void appendLast(veclis_t* vl, void* data)`, que agrega un nuevo dato al final del vector.

(20p) b. Implementar la función `void removeFirst(veclis_t* vl, func_t delete)`, que borra el primer elemento utilizando la función `delete` y luego acomoda los siguientes datos haciendo que ocupen el lugar liberado del vector. Ejemplo borrando de una estructura de 14 elementos.



Ej. 2. (30 puntos)



Considerar una matriz de 10×10 bytes, interpretados como números con signo. Identificar dos casos posibles:

(A) Los datos pintados en gris claro, cumplen que sus dos indices de coordenadas son pares.
(B) Los valores en gris oscuro cumplen que sus dos coordenadas son números impares.

(30p) a. Construir una función que tome un puntero a una matriz y obtena la sumatoria de todos los valores (A) menos la sumatoria de todos los valores (B).

Nota: Se debe procesar minimamente 5 datos simultaneamente, es decir, una fila.

Ej. 3. (30 puntos)

Se tiene una opción de compilación que intercala un llamado a una función entre cada par de instrucciones. Utilizando esta opción se busca programar a la función intercalada para que realice la siguiente tarea. Cada vez que se llama debe llamar a otra función que determine si la función intercalada debe ser modificada por otra o debe se remplazada por instrucciones `nop`.

```
strlen:
    mov eax, 0
    ciclo:
    mov cl, [rdi]
    cmp cl, 0
    je fin
    inc eax
    jmp ciclo
    fin:
    ret
```

```
strlen:
    CALL LOG
    mov eax, 0
    ciclo:
    CALL LOG
    mov cl, [rdi]
    CALL LOG
    cmp cl, 0
    CALL LOG
    je fin
    CALL LOG
    inc eax
    CALL LOG
    jmp ciclo
    fin:
    CALL LOG
    ret
```

```
strlen:
   0: e8 xx xx xx xx
   5: b8 00 00 00 00
ciclo:
   a: e8 xx xx xx xx
   f: 8a 0f
  11: e8 xx xx xx xx
  16: 80 f9 00
  19: e8 xx xx xx xx
  1e: 74 0e
  20: e8 xx xx xx xx
  25: ff c0
  27: e8 xx xx xx xx
  2c: eb dc
fin:
  2e: e8 xx xx xx xx
  33: c3
```

```
strlen:
   0: e8 yy yy yy yy
   5: b8 00 00 00 00
ciclo:
   a: 90 90 90 90 90
   f: 8a 0f
  11: 90 90 90 90 90
  16: 80 f9 00
  19: 90 90 90 90 90
  1e: 74 0e
  20: e8 yy yy yy yy
  25: ff c0
  27: e8 yy yy yy yy
  2c: eb dc
fin:
  2e: e8 yy yy yy yy
  33: c3
```

En el ejemplo, la función intercalada se llama `LOG`, que en el código binario esta representada por la letra `x`. En el último paso del ejemplo la función fue remplazada por `y` para algunos casos y por `0x90` en otros. Estos últimos son instrucciones `NOP` de un byte.

Para determinar si la función debe ser remplazada se debe llamar a la función `int getNewFunc(void* eip)`, que toma el puntero a la próxima instrucción y retorna un *offset* a la función por remplazar o cero si la función debe ser remplazada por `NOP`s.

En el caso de remplazar el llamado a `LOG` por una nueva función, está debe ser ejecutada luego de ser remplazada, como si nunca se hubiera llamando a la función `LOG`.

(30p) a. Implementar la función pedida. Tener en cuenta que la misma debe conservar el estado de todos los registros.

Nota: Utilizar las instrucciones `sahf` y `lahf` de ser necesario. Considerar que `getNewFunc` respeta convención C.