

Pregunta 1 (40%)

Una cola circular de enteros se almacena en un archivo binario, por ejemplo con nombre *cola.bin*. El comando *./encolar* agrega un elemento al final de la cola y el comando *./desencolar* extrae el elemento que está en primer lugar. La tabla de abajo muestra la estructura del archivo de una cola con los elementos 100, 36 y 48, siendo 100 el primer elemento.

Posición en el arreglo y su contenido									
	P	C	0	1	2	3	4	5	6
Contenido del archivo	2	5			100	36	48		

El archivo almacena enteros de 4 bytes en formato binario *little-endian*, el mismo de la arquitectura de trabajo. El primer entero es *P* (primero), el segundo es *C* (cola) y luego viene un arreglo circular de tamaño *N* con el contenido de la cola. *P* es la posición del elemento que está en primer lugar en la cola y *C* la posición en donde se escribirá el próximo elemento. Para desencolar un elemento se accede al contenido de la posición *P* y luego se suma 1 a *P*. Para encolar un entero, se escribe en la posición *C* y se suma 1 a *C*. La cola es de tamaño fijo y es circular porque si *P* o *C* exceden el tamaño del arreglo, vuelven a la posición 0. Por ejemplo si se agrega un elemento y *C* es *N*-1, *C* queda en el valor 0. La cola está vacía cuando $P \equiv C$. Puede almacenar un máximo de *N*-1 elementos (6 en el ejemplo).

Programe la función *main* para el comando *./desencolar* que recibe como parámetro el nombre del archivo que almacena la cola. Si la cola no está vacía, debe mostrar el elemento desencolado en la salida estándar y el código de retorno del comando debe ser 0. El cuadro de la derecha muestra ejemplos de ejecución. El código de retorno del comando debe ser distinto

```
$ ./desencolar cola.bin
100
$ ./desencolar cola.bin
36
$ ./desencolar cola.bin
48
$ ./desencolar cola.bin
cola vacía
$
```

de 0 si (i) la cola está vacía, escribiendo "cola vacía" en la salida estándar de errores, o (ii) si no se puede acceder al archivo, diagnosticando el error con la función *perror*. No necesita diagnosticar otras situaciones de error. **No programe** el comando *./encolar*.

Restricciones: No puede leer todo el archivo. Para acceder al archivo Ud. solo puede usar las funciones *fopen*, *fread*, *fwrite*, *ftell*, *fseek* y *fclose*. Determine el valor de *N* calculando el tamaño del archivo con *fseek* y *ftell*.

Pregunta 2 (30%)

La función *f* de la derecha está programada en assembler Risc-V. La tabla de abajo muestra la ejecución parcial de la función *f*. **Prosiga llenando la tabla** con la ejecución de *lw a5,5(a5)* hasta que se ejecute la instrucción *ret*. No incluya lo que ya muestra este enunciado. Cada vez que se modifique *a0*, *t0*, *t1*, *t2* o la memoria, debe indicarlo en la columna correspondiente. En el caso de los saltos condicionales, tacharemos la instrucción para marcar cuando este no ocurre.

```
f:  mv    a5,a0
    li    a0,0
L2: mv    a4,a5
    lw    a5,4(a5)
    addi  a0,a0,1
    bne   a5,zero,L2
    sw    a1,4(a4)
    ret
```

		dirección de memoria				100	104	200	204	300	304	400	404
instrucción		registros				contenido de memoria							
		a0	a1	a4	a5								
		300	200			2	400	4	0	1	100	3	0
mv	a5,a0				300								
li	a0,0	0											
mv	a4,a5			300									
lw	a5,4(a5)				100								
addi	a0,a0,1	1											
bne	a5,zero,L2												
mv	a4,a5			100									
...	prosiga ...												

Pregunta 3 (30%)

Traduzca la función *g* del cuadro de la derecha a assembler Risc-V.

```
int *g( int *a, int n,
        int x ) {
    int *t= &a[n];
    int s= 0;
    while (a<t) {
        s += *a;
        if (s>=x) {
            *a= 0;
            return a;
        }
        a++;
    }
    return NULL;
}
```