UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN
FUNDAMENTOS DE INFRAESTRUCTURA TECNOLÓGICA - ISIS1304
Trabajo práctico 3

Este trabajo vale 5% de la nota del curso.
Debe ser elaborado individualmente.
No se permite ningún tipo de consulta sobre el trabajo con otras personas.
Se debe entregar por BNe a más tardar el 20 de octubre de 2024

A. OBJETIVOS

- Practicar el lenguaje C y desarrollar un programa de baja complejidad.
- Conocer las operaciones de C para el manejo de bits.
- Aplicar lo anterior haciendo una rotación de bits en un vector de unsigned char.

B. DESCRIPCIÓN DEL TRABAJO

El objetivo del trabajo es desarrollar un programa en C que lea un vector de unsigned char (bytes) de tamaño `n` para después realizar una rotación a la izquierda de tamaño *k* bits usando el vector completo como operando.

Nota: en este artículo puede ver la definición de la operación de rotación: https://es.wikipedia.org/wiki/Operador_a_nivel_de_bits

El programa toma todo el vector como si fuera un solo número, un byte tras otro, y numerando los bits de izquierda a derecha:

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	...
v[0]								v[1]								v[2]				

Y hace una rotación a la izquierda de `k` bits. Es decir, transforma una secuencia de m bits:

$b_0,\ldots b_{k-1}, b_k,\ldots b_{m-1}$

en: $b_k,\ldots b_{m-1}, b_0,\ldots b_{k-1}$

El vector de char puede tener cualquier tamaño; de hecho, el programa debe empezar preguntándole dicho tamaño (`n`) al usuario, para después proceder a leer los `n` elementos que lo componen. También debe pedir el valor de `k`.

Al final imprime el vector, todos los elementos en la misma línea, en hexadecimal (ver formato `%X` de `printf`).

Ejemplo:

El usuario indica que n = 8, y después ingresa los valores: 255, 255, 255, 254, 0, 0, 0 y 12. Adicionalmente indica que k = 4.

El vector de `char (tamaño 8)` es:

Valor en hexa	FF	FF	FF	FE	00	00	00	0C

Después de la rotación, el vector debe `ser`:

Valor en hexa	FF	FF	FF	E0	00	00	00	CF

Imprime:

```
FFFFFFE0000000CF
```

Algoritmo

Para hacer la rotación usaremos un método basado en inversiones de los bits como se describe a continuación.

Supongamos que tenemos una secuencia de m bits: b_0,... b_{k-1}, b_k,... b_{m-1}

1- Se invierte toda la secuencia; obtenemos: b_{m-1},... b_k, b_{k-1},... b_0
2- Se invierten los m-k bits del comienzo; obtenemos: b_k,... b_{m-1}, b_{k-1},... b_0
3- Se invierten los k bits del final; obtenemos: b_k,... b_{m-1}, b_0,... b_{k-1}

Estructuras de datos

- Unas variables para almacenar el tamaño del vector (`n`) y el número de corrimientos (k).
- Un apuntador a `unsigned char`: para apuntar al vector de bytes.
- Por supuesto puede usar más variables escalares, pero NO puede usar más vectores.

Estructura del programa

El programa debe tener tres procedimientos: el `main` y otros dos procedimientos encargados de efectuar la rotación.

`intercambiarBits:`

- Tiene tres parámetros: un apuntador al vector de bytes, y dos enteros `x` y `y`, que designan los bits que se van a intercambiar.
- Intercambia de lugar a b_x y b_y. Es decir, pasa de: b_0,... b_x,... b_y,... b_{m-1}
  a: b_0,... b_y,... b_x,... b_{m-1}
- Atención: solo se Intercambian b_x y b_y; no los bits intermedios entre ellos (para eso, ver procedimiento siguiente).

`invertirSecuencia:`

- Tiene tres parámetros: un apuntador al vector de bytes, y dos enteros `x` y `y`, que designan el rango de bits que se van a invertir ($x \leq y$).
- Invierte los bits en el rango indicado usando para ello el procedimiento `intercambiarBits`. Es decir, transforma: b_0,... b_x, b_{x+1},... b_{y-1}, b_y,... b_{m-1}
  en: b_0,... b_y, b_{y-1},... b_{x+1}, b_x,... b_{m-1}
- Este procedimiento Intercambia b_x y b_y y además los bits intermedios entre ellos.

`main:`

- Le pide al usuario el tamaño del vector (`n`) para inicializar la respectiva variable (validar que `n > 0`).
- Declara e inicializa el apuntador al vector de `unsigned char`. La inicialización consiste en ponerlo a apuntar a un vector del tamaño pedido. El vector se crea dinámicamente usando la función de C `calloc`.
- Inicializa el vector: le pide al usuario que teclee el valor de cada elemento.
- Le pide al usuario el número de bits (k) de la rotación (validar $0 < k < 8*n$).
- Invoca el procedimiento `invertirSecuencia` para realizar las inversiones necesarias según lo indicado por el algoritmo.

- Imprime el resultado (el vector de `unsigned char`) en hexadecimal.

Restricciones y consideraciones

- Las estructuras de datos se declaran e inicializan en el `main`.
- El programa solo debe constar del `main` y los dos procedimientos de cálculo.
- La rotación se efectúa sobre el mismo vector; no puede usar vectores auxiliares.
- No puede usar librerías externas que resuelvan directamente el problema; solo las básicas de C (entrada/salida, manejo de cadenas, etc.).
- Los números hexa deben estar en letras mayúsculas.
- El vector puede tener cualquier tamaño.
- Los programas se calificarán únicamente usando el ambiente de visual de las máquinas virtuales. Si el programa no compila en este ambiente, se considerará que no corre (así compile en otros ambientes).

C. CONDICIONES DE ENTREGA

Entregar el código fuente junto con el ejecutable en un archivo `.zip`. El nombre del archivo debe ser: `TP3_`*`código_apellido_nombre`*`.zip`

Al comienzo del archivo fuente escriba su nombre, código y correo.

Si su programa no funciona o si su solución tiene particularidades, puede enviar un archivo `.docx` o `.pdf` explicando por qué cree que no funciona o qué fue lo que hizo.

El trabajo es individual. No debe haber consultas con otros estudiantes.

Se puede solicitar una sustentación sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota.

El trabajo se debe entregar por BNe a más tardar el **20 de octubre de 2024 hasta las 11:50 pm. No se recibirán trabajos entregados con posterioridad**.

D. CRITERIOS DE CALIFICACIÓN PARA LOS PROGRAMAS

La calificación consta de dos partes:

- Ejecución (50%). Se harán 5 pruebas: los tres casos de prueba y otras 2 adicionales. Para cada caso, se revisará si la salida es correcta o no según los requerimientos establecidos en el enunciado. Cada prueba vale 10%.
- Inspección del código (50%). Se consideran tres aspectos:
  - 10% - legibilidad (nombres dicientes de variables, comentarios, indentación)
  - 20% - manejo de bits (uso correcto y eficaz de los operadores de bits de C: >>, &, etc. para resolver el problema)
  - 20% - Correcto y eficaz manejo de la estructura de datos (declaración, creación, recorrido, manejo de los elementos, lectura, impresión).

No cumplir las condiciones de entrega acarreará una penalidad del 10% de la nota.

E. CASOS DE PRUEBA

Caso 1:

Entrada: n = 8; valores = 255, 255, 255, 254, 0, 0, 0 y 12; k = 3

Salida:

```
FFFFFFF000000067
```

Caso 2:

Entrada: n = 8; valores = 255, 255, 255, 254, 0, 0, 0 y 12; k = 63

Salida:

```
7FFFFFFF00000006
```

Caso 3:

Entrada: n = 1; valores = 145; k = 1

Salida:

```
23
```