Estructuras de Daos y algoritmos: Laboratorio 1
Profesor Cátedra: Demian Schkolnick
Profesor Laboratorio: Yerko Ortiz

"If you can't solve a problem, then there is an easier problem you can solve: find it."

How to solve it - George Pólya

Josefo y el tesoro

Josefo y sus amigos han encontrado un tesoro, pero sus amigos se niegan a compartir el tesoro en partes iguales por lo que han decidido jugar un juego de eliminación, Josefo y sus amigos formaran un circulo donde cada persona estará enumerada desde el 1 hasta N; dado el circulo de personas las reglas del juego que han definido son las siguientes:

- Siempre se empieza eliminando desde la posición 2.
- Siempre se elimina a la persona subsiguiente, es decir si se elimino al 2, luego se elimina al 4, luego al 6 y así sucesivamente.
- La cantidad de personas en el circulo es finita (N).
- El ganador es la última persona restante en el circulo.

Dado que Josefo y sus amigos saben que existe una posición ganadora para cada posible cantidad de personas en el circulo, han puesto la regla de que cada persona tiene derecho a escoger su posición en el circulo, solo que el más rápido en escoger se queda con la posición. Diseñe e implemente un algoritmo que permita a Josefo encontrar la posición ganadora con una cantidad N de personas en el circulo. Este algoritmo debe simular el proceso de eliminación utilizando listas enlazadas, y retornar el número del ganador para N personas.

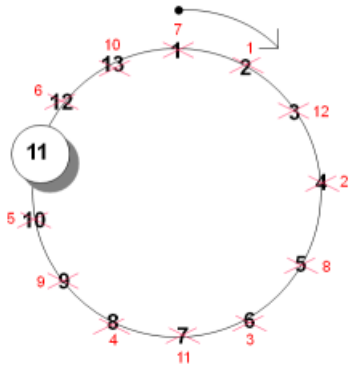

Figure 1: Ejemplo de eliminación para N=13 (los números en rojo corresponden al orden de eliminación).

Entrada y salida

- El input de su método ha de ser un número entero N.
- El output de su método ha de ser el número ganador, para que Josefo pueda reclamar esa posición y ganar el tesoro.

Hint: Para hacer simulación del proceso de eliminación inserte N nodos enumerados de 1 a N en la lista y posteriormente simule eliminando nodos con las reglas descritas previamente.

Ahab el inversor de listas

El capitán Ahab cansado de perseguir a Moby Dick, ha encontrado un nuevo hobbie: aprender algoritmos y estructuras de datos. En su afán de aprender se encuentra actualmente aprendiendo listas enlazadas, el ya conoce cómo invertir una lista enlazada de manera iterativa y recursiva, pero ahora se encuentra estancado invirtiendo listas en grupos de 3. La idea de esto es que dada una lista enlazada con N nodos, se inviertan cada subgrupo de 3 nodos dentro de la lista.

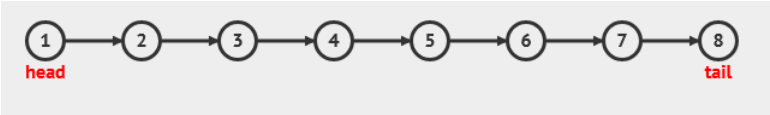

Figure 2: Ejemplo de lista enlazada.

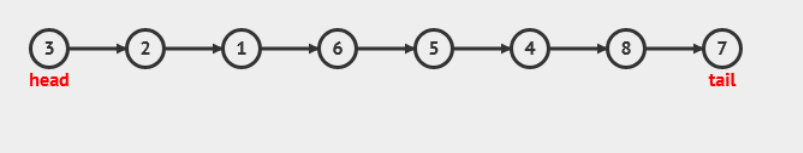

Figure 3: Ejemplo de lista enlazada invertida en grupos de 3.

Diseñe e implemente un método para una lista enlazada, que invierta en grupos de 3, la lista empezando desde el nodo head.

Entrada y salida

- El input de su método ha de ser la cabeza de una lista enlazada.
- El output de su método ha de ser la cabeza de la lista enlazada, con los nodos invertidos.

Hint: Recuerde que en el laboratorio se estudiará o estudió(depende del día que lea esto) cómo invertir listas, en base a esto puede que le sea más fácil resolver este problema.

Alice y Bob perdidos

Alice y Bob han decidido tomarse unas vacaciones de sus trabajos en criptografía y protocolos de seguridad, para viajar a la isla de las listas enlazadas. En esta isla ofrecen de premio una cena para dos personas para todas las personas que sean capaces de implementar un algoritmo que pueda decidir si una lista enlazada tiene un ciclo o no. Se dice que una lista enlazada tiene un ciclo si y solo si, dentro de la lista enlazada existe una sublista enlazada circular tal como en la siguiente imagen:

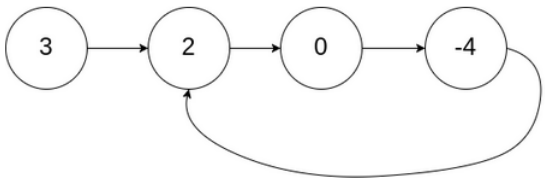

Figure 4: Ejemplo de lista enlazada con un ciclo.

Diseñe e implemente un algoritmo que dada la cabeza de una lista enlazada, sea capaz de decidir si dentro de la lista enlazada existe un ciclo o no.

Entrada y salida

- El input de su método ha de ser la cabeza de una lista enlazada que puede contener un ciclo o no.
- El output de su método ha de ser un booleano que asevere si existe un ciclo dentro de la lista o no.

Hint: Para crear un ciclo en una lista enlazada basta con que el puntero next de alguno de los nodos apunte hacía cualquiera de los nodos **anteriores** a el, en el laboratorio igualmente se explicará o explicó, cómo crear un ciclo.

Bonus: Punteros dobles

Esta pregunta es completamente opcional.
Una lista enlazada doble requiere dos punteros en la definición de nodo, un puntero next para ir al siguiente nodo y un puntero prev para ir al nodo anterior. ¿Es posible crear una lista enlazada doble, donde las referencias del nodo siguiente y nodo anterior se almacenen dentro de un solo puntero en lugar de dos?, en caso que sea posible argumente el por qué.

Hint: No es necesario implementar nada, con la explicación basta y sobra. La respuesta puede ir como le acomode más (pdf, word, markdown, etc). La explicación debe ser lo suficientemente buena como para que cualquiera de sus compañeros la entienda, pero lo suficientemente acotada de forma que no utilice más de tres párrafos en explicar. (Puede poner ejemplos gráficos si así lo desea)

Evaluación:

- Cada problema vale dos puntos, de un total de 6 puntos.
- El problema bonus puede agregarle un punto extra.
- La copia entre pares se sancionará con la nota mínima.
- La fecha de entrega es para el viernes 16 de Septiembre a las 23:59.
- En CANVAS se creará una entrega para que pueda adjuntar su código.
- Deben trabajar en grupos de dos personas máximo, en caso de trabajar solos notificar al profesor.
- En la entrega deben ir los nombres de los integrantes del grupo, en caso de trabajar en parejas.