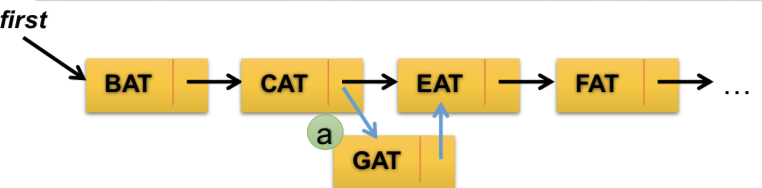
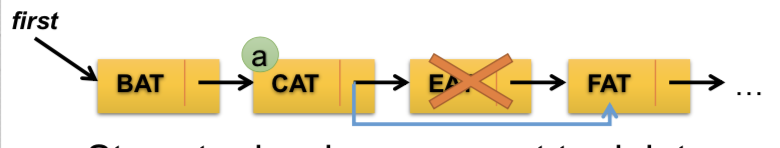
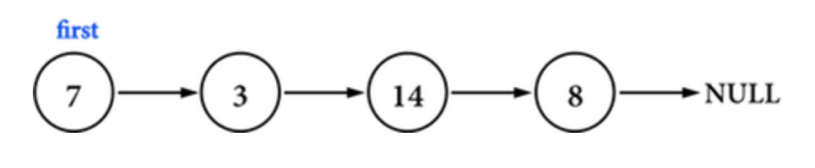
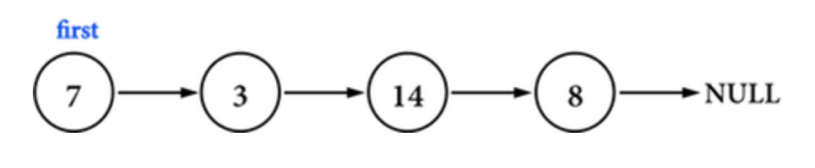
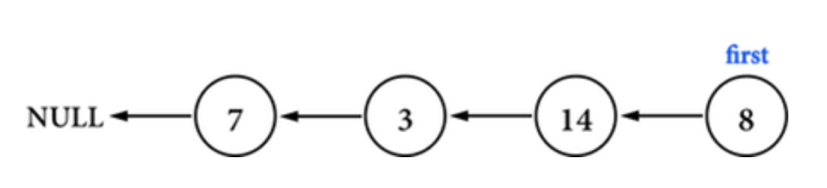
1. Write a pseudo code to insert an element with value “GAT” into between element “CAT” and “EAT” in a linked list as illustrated below. (Your code should manipulate pointers to achieve the goal.)



1. Write a pseudo code to delete the element with value “EAT” in a linked list as illustrated below. (Your code should manipulate pointers to achieve the goal.)



1. Write a pseudo code to clear the whole linked list as illustrated below.   
   (Your code should manipulate pointers to achieve the goal.)  
   
2. Write a pseudo code to reverse the linked list A into B as illustrated below. (Your code should manipulate pointers to achieve the goal.)

 AB

1. Can one use a linked list to implement a circular queue? If yes, please describe how to do it.
2. Given the following Node structure of a singly linked list and an example of a “circular” (if there are cycle in the list, it is considered circular here) linked-list, please give an efficient pseudo code about how to determine whether a singly linked list is circular or not.

**public class ListNode {**

**ListNode next;**

**int val;**

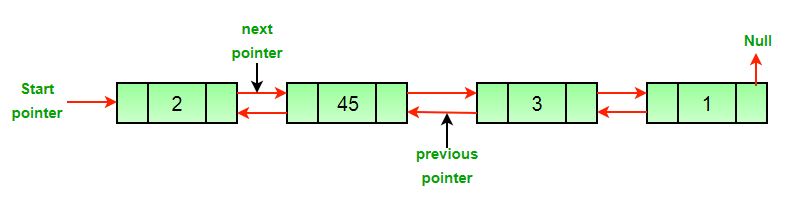
**ListNode(int value) { this.val = value; }**

**}**

A “circular” linked list example:



1. Please analyze the time complexities of the search function for a linked list and an array.
2. Please analyze the time complexity of the “insert” and “delete” function of a linked list. Consider cases when the mentioned functions occur at the head, middle and the end of a linked-list.)
3. Write a pseudo code to “swap” two elements in a doubly linked list as illustrated below.



For example, node with index 0 stores the number 2, node with index 1 stores 45, node with index 3 stores 3, and node with 4 stores 1. If we want to swap node 0 (i.e. node with index=0) and node 1, the swap function call should be swap(0, 1), and the function would perform swapping in the double linked-list.

11. Please complete the TODO part.

class Node{

    private:

        int data;

        Node \*pre;

        Node \*next;

    public:

        // Constructor

        Node():data(0),pre(0),next(0){}

        Node(int data):data(data),pre(0),next(0){}

        Node(int data, Node \*pre):data(data),pre(pre),next(0){}

        Node(int data,Node \*pre,Node \*next):data(data),pre(pre),next(next){}

        // functions

        Node\* getPre(){

            return pre;

        }

        Node\* getNext(){

            return next;

        }

        int getData(){

            return data;

        }

        void setData(int data){

            this->data=data;

        }

        void setPre(Node \*pre){

            this->pre=pre;

        }

        void setNext(Node \*next){

            this->next=next;

        }

};

class DoubleLinkedList{

    private:

        Node \*firstNode;

        Node \*lastNode;

    public:

        DoubleLinkedList():firstNode(0),lastNode(0){}

        DoubleLinkedList(Node \*firstNode):firstNode(firstNode),lastNode(firstNode){}

        void printAll(){

            //TODO : print all the elements in linked list

        }

        void push\_back(Node \*node){

            //TODO : push new element in the last of linked list

        }

};

12. The Josephus Problem (Josephus permutation) is a problem in theoretical math where we call “counting-out game” here. Suppose there are ***N*** people standing in a circular waiting queue to be executed. The “counting-out” proceeds around the circle in a fixed direction, and each round we “count out” (or “execute” in some version) a person in the circle. The person “after” the “counted-out” person will be considered survived and then execute the person who stands “after” him (“after” means the next one toward the fixed direction we mentioned before.) The elimination starts at person indexed #1 and proceeds around the circle (which is becoming smaller and smaller as the executed people are removed), until only the last person remains, who will be given freedom. Write a pseudo code to trace the process and status of each round, and output the number of the last survived person. (For simplicity, you can name these persons based on the indices from 1 to N ) (Hint: Using a circular linked-list is a straightforward manner.)

You may refer to this video for more animation of this problem: <https://www.youtube.com/watch?v=uCsD3ZGzMgE>