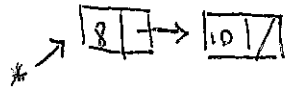


```
#pragma once
#include <iostream>
using namespace std;

template <class T> class LinkedList{
private:
  struct Node{
    Node * next;
    T value;
    Node(T value, Node * next = NULL){
      this->next = next;
      this->value = value;
    }
  };
  Node * head;
public:
  LinkedList(){
    head = NULL;
  }
  ~LinkedList(){
    if(head){
      Node * lead = head->next;
      Node *follow = head;
      while(lead){
        delete(follow);
        follow = lead;
        lead = lead->next;
      }
      delete(follow);
    }
  }
  void push(T v){
    head = new Node(v, head);
  }
  T pop() {
    T item = head->value;
    Node * temp = head;
    head = head->next;
    delete(temp);
    return item;
  }

  void pop(T item){
    if(!head) return;
    if(head->value == item){
      T item = pop();
      return;
    }
    Node * lead = head->next;
    Node * follow = head;
    while(lead != NULL){
      if(lead->value == item){
        Node * temp = lead->next;
        delete(lead);
        follow->next = temp;
        return;
      }
      lead = lead->next;
      follow = follow->next;
    }
  }

  T get(int index){
    if(!head) throw(1);
    int i = 0;
    Node * next = head;
    while(i < index){
      i++;
      if(next->next){
```

```
          next->next;
        }else{
          throw(1);
        }
      }
      return next->value;
    }

    friend ostream& operator<<(ostream& os, const LinkedList& list){
      Node * next = list.head;
      while(next != NULL){
        os << next->value << " ";
        next = next->next;
      }
      return os;
    }
  };
```

First	First
In	In
First	Last
Out	Out

V | N	Linked List	FIFO	O(n)
		FILO	O(1)

vs

P | V | N	Doubly Linked List	FIFO	O(1)
		FILO	O(1)

DL List Queue

push

1. make new node
   p : NULL, v : value, n : rear
2. rear → prev = new node
3. rear = new node

dequeue

1. front = front → next
2. delete (front → next)
3. front → next = NULL

front, rear queue
=
head, tail LList

with priority queue

add!

1. NewNode (cur, x, next)
2. cur → next = New Node
3. NewNode → next → prev = NewNode