UNIVERSITAT DE BARCELONA

Sessió 5. Teoricopràctica

Estructura de Dades
Curs 2020-2021

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona

Contingut

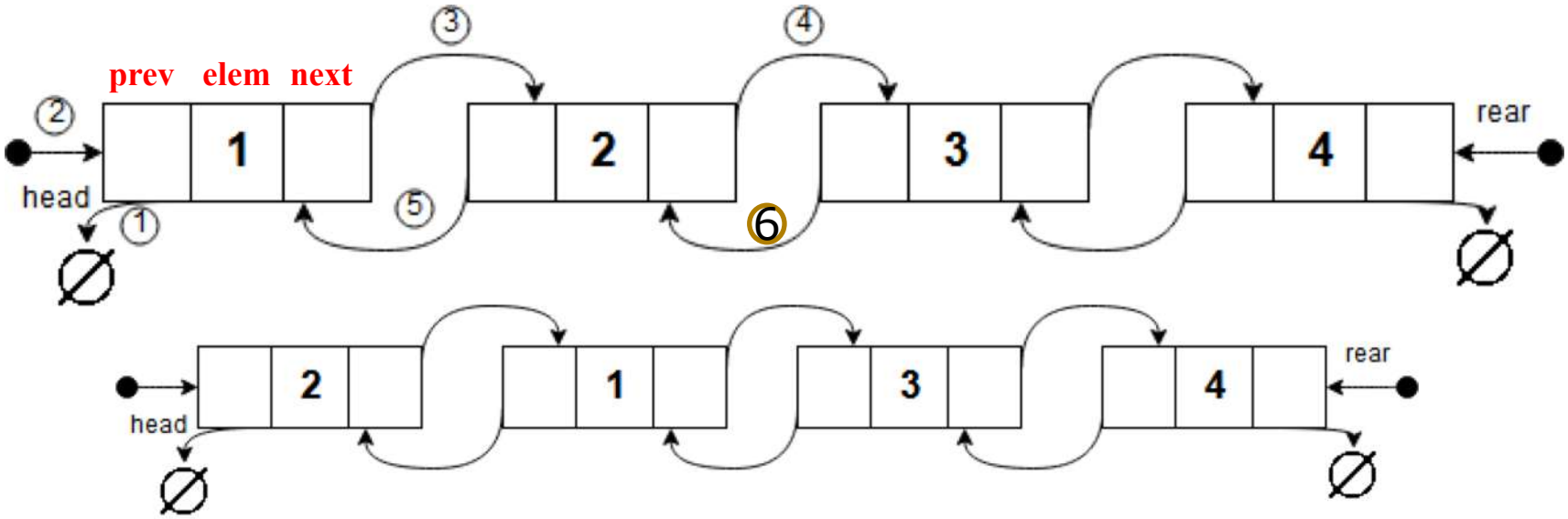
1. Exercicis de Punters
2. Pila Dinàmica
3. Problema Palíndrom

Exercicis de Punters

Punters 1

Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (els nodes tenen els atributs públics: **next, prev, elem**). MIREU EL DIBUIX

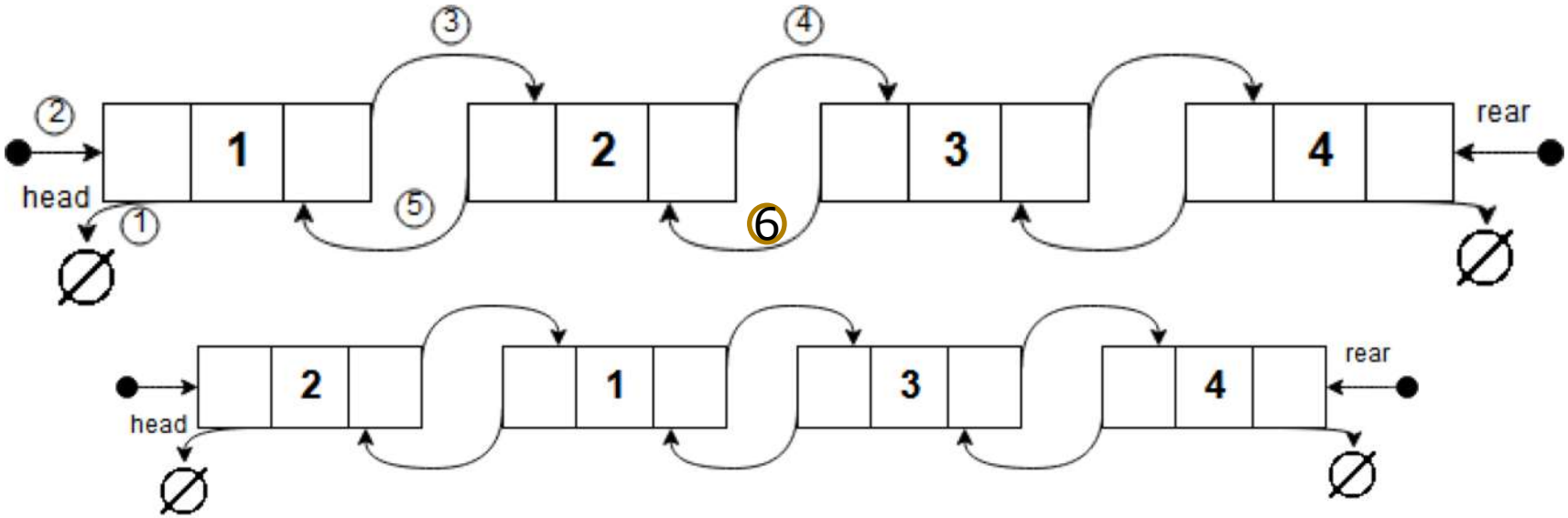
- Volem imprimir per pantalla el node 1 usant el punter head
- Volem imprimir per pantalla el node 2 usant el punter head
- Volem imprimir per pantalla el node 3 usant el punter head
- **No heu d'usar punters auxiliars.**



Punters 1 (Solució)

Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (els nodes tenen els atributs públics: **next, prev, elem**).

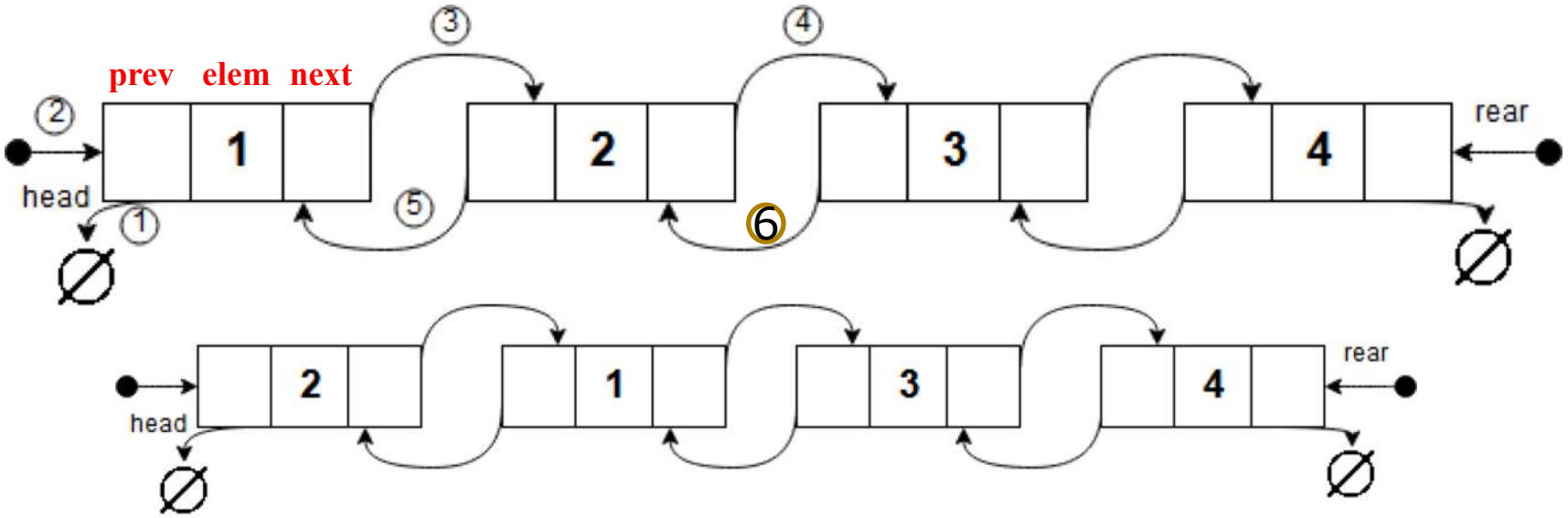
- Volem imprimir per pantalla el node 1 usant el punter head
  - **cout << head->elem**
- Volem imprimir per pantalla el node 2 usant el punter head
  - **cout << head->next->elem**
- Volem imprimir per pantalla el node 3 usant el punter head
  - **cout << head->next->next->elem**



Punters 2

Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (els nodes tenen els atributs públics: **next, prev, elem**). MIREU EL DIBUIX

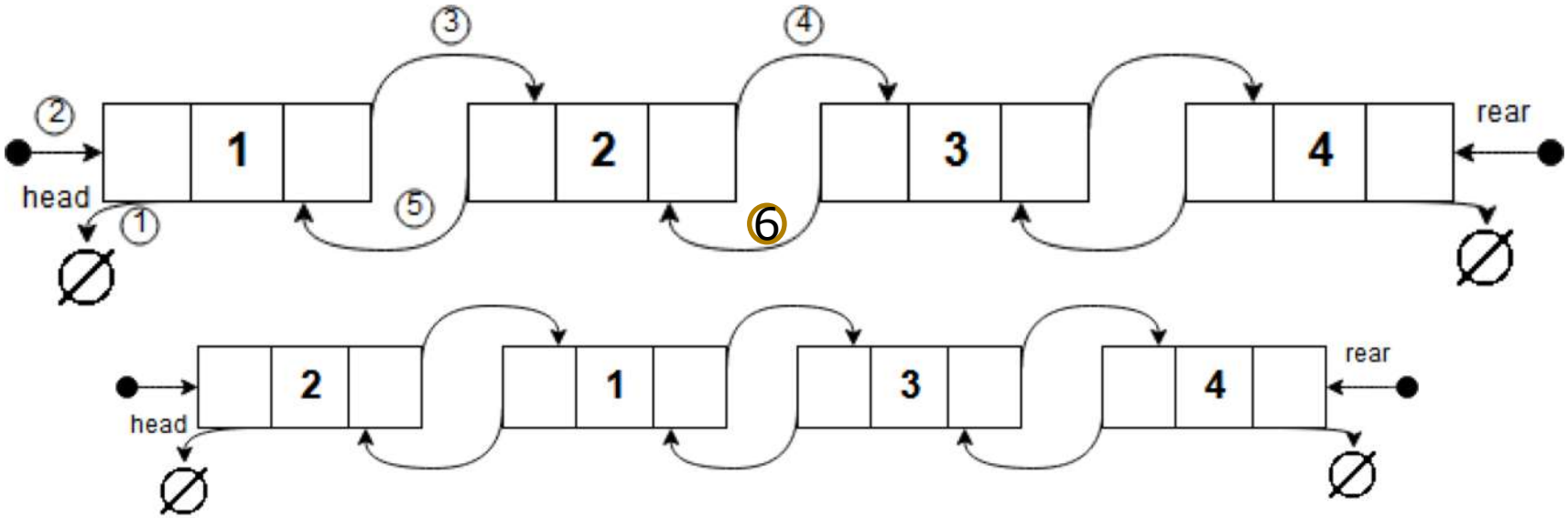
- Volem imprimir per pantalla el node 1 usant el punter rear
- Volem imprimir per pantalla el node 2 usant el punter rear
- Volem imprimir per pantalla el node 3 usant el punter rear
- **No heu d'usar punters auxiliars.**



Punters 2 (Solució)

Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (els nodes tenen els atributs públics: **next, prev, elem**).

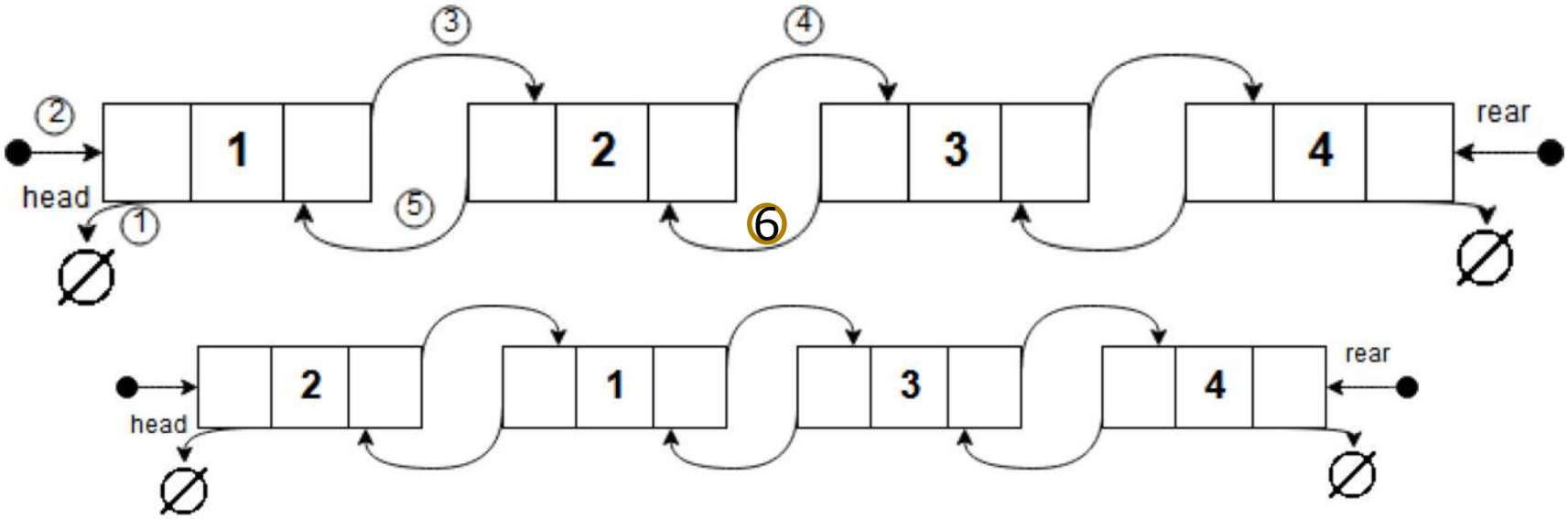
- Volem imprimir per pantalla el node 1 usant el punter rear
  - **cout << rear->prev->prev->prev->elem**
- Volem imprimir per pantalla el node 2 usant el punter rear
  - **cout << rear->prev->prev->elem**
- Volem imprimir per pantalla el node 3 usant el punter rear
  - **cout << rear->prev->elem**



Punters 3

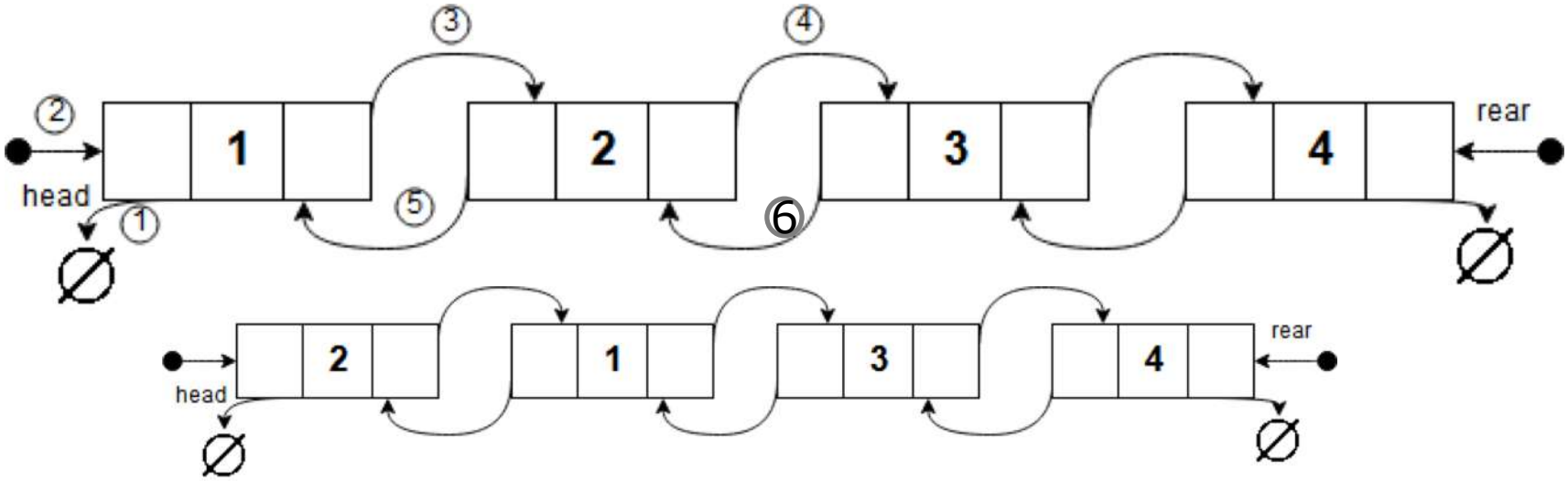
Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (els nodes amb atributs públics: **next, prev, elem**), volem intercanviar les posicions dels nodes amb elements 1 i 2 seguint els passos descrits a la imatge. (No volem moure els continguts, **volem moure els nodes**)

- Escriu la resposta donant les instruccions a cada pas. No es pot modificar l'ordre dels passos.
- **No heu d'usar punters auxiliars. Tot es fa a partir del head.**



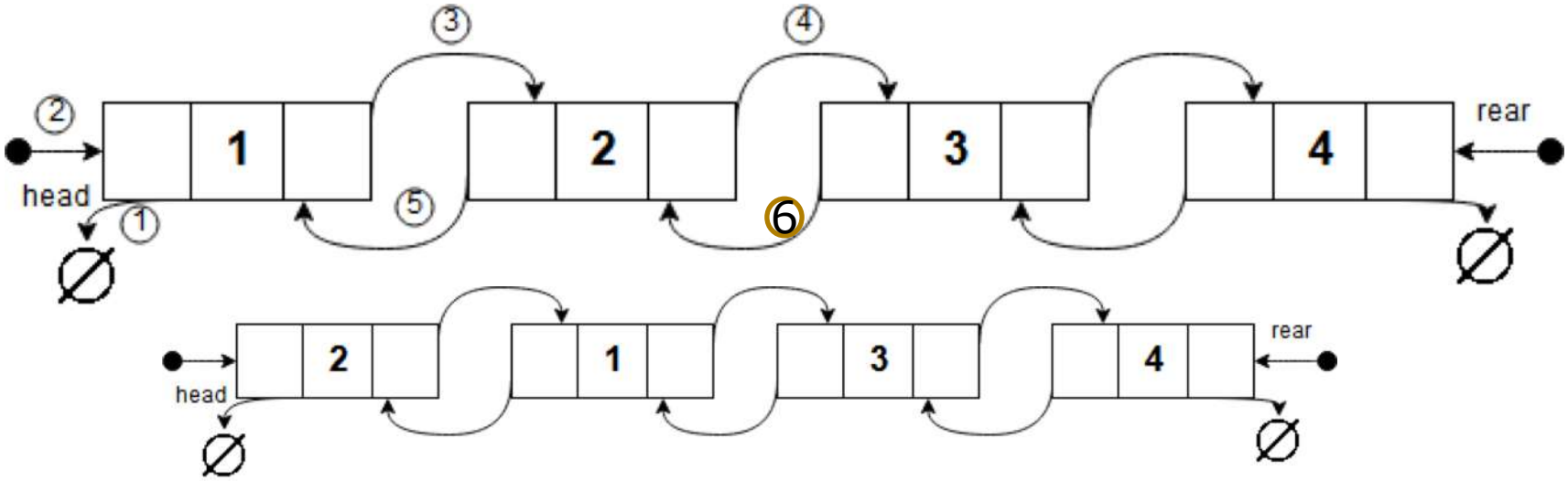
Punters 2 (feu la SOLUCIÓ)

1.
2.
3.
4.
5.
6.



Punters 2 (SOLUCIÓ)

1. head->prev = head->next;
2. head = head->next;
3. head->prev->next = head->next;
4. head->next = head->prev;
5. head->prev = nullptr;
6. head->next->next->prev= head->next;



Pila Dinàmica

LinkedStack.h

- Definiu la classe LinkedStack
  - Pel primer node de la pila, tindrem una variable anomenada front
  - Penseu si us cal algun altre atribut
- Definiu la classe Node
  - Els nodes són simples, només tenen l'apuntador al següent element
- Definiu totes dues classes en C++

SOLUCIÓ LinkedStack.h i Node.h

```
#include <iostream>
#include <stdexcept>

using namespace std;

#include "Node.h"
using namespace std;

template <class Element>
class LinkedStack{
    private:
        Node<Element> *front;
        int  num_elements;
    public:
        LinkedStack();
        ~LinkedStack();
        int size() const;
        bool empty() const;
        const Element& top() const;
        void push(const Element & e);
        void pop();
};
```

```
template <class Element>
class Node
{
    private:
        Element element;
        Node<Element> *next;
    public:
        Node(Element e);
        ~Node();
        const Element& getElement() const;
        Node<Element> * getNext()const;
        void setNext(Node<Element> * elem);
};
```

Constructor

- Implementeu el constructor de la classe LinkedStack

Constructor (SOLUCIÓ)

```
template <class Element>
LinkedStack<Element>::LinkedStack(){
    this->front = nullptr;
    this->num_elements = 0;
}
```

El this->
és opcional

Print

- Implementeu un mètode print de la classe LinkedStack
  - void print();

```
template <class Element>
void LinkedStack<Element>::print()
{
  // AQUí el vostre codi
}
```

Print (SOLUCIÓ)

```
template <class Element>
void LinkedStack<Element>::print()
{
   if  (this->empty()) cout << "Stack =[]" << "  size " << size() << endl;
   else
   {
      cout << "Stack=[" ;

      Node<Element> *aux_pointer = this->front;
       while (aux_pointer != nullptr )
       {
          cout << " " << aux_pointer->getElement()  ;
          aux_pointer = aux_pointer->getNext();
       }
      cout << " ] size " << size() << " " << endl;
   }
}
```

Problema Palíndrom

Problema Cap_i_Cua

- Donat el següent main, feu la funció detecta_palindrom per detectar si conjunt de valors és cap i cua (palíndrom) amb l'ús de piles. No es permet cap altre estructura de dades.

```
int main() {
    try{
        LinkedStack<int> pila;

        int valors[5]  = {4, 2, 2, 2, 4};

        for (int i = 0 ; i < 5; i++) {
            pila.push(valors[i]);
        }
   if (detecta_palindrom(pila))
           cout << "la seqüencia de numeros SI es palindroma" << endl;
       else cout << "la seqüencia de numeros NO es palindroma" << endl;
}
```

Aquesta exercici queda com a proposta per fer durant aquesta setmana. ES RESOLDRÀ A LA PROPERA SESSIÓ