

Sessió 8. Problemes PARCIAL

Estructura de Dades
Curs 2020-2021

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

Contingut

1. Exercicis de Punters
2. Exercicis ArrayStack

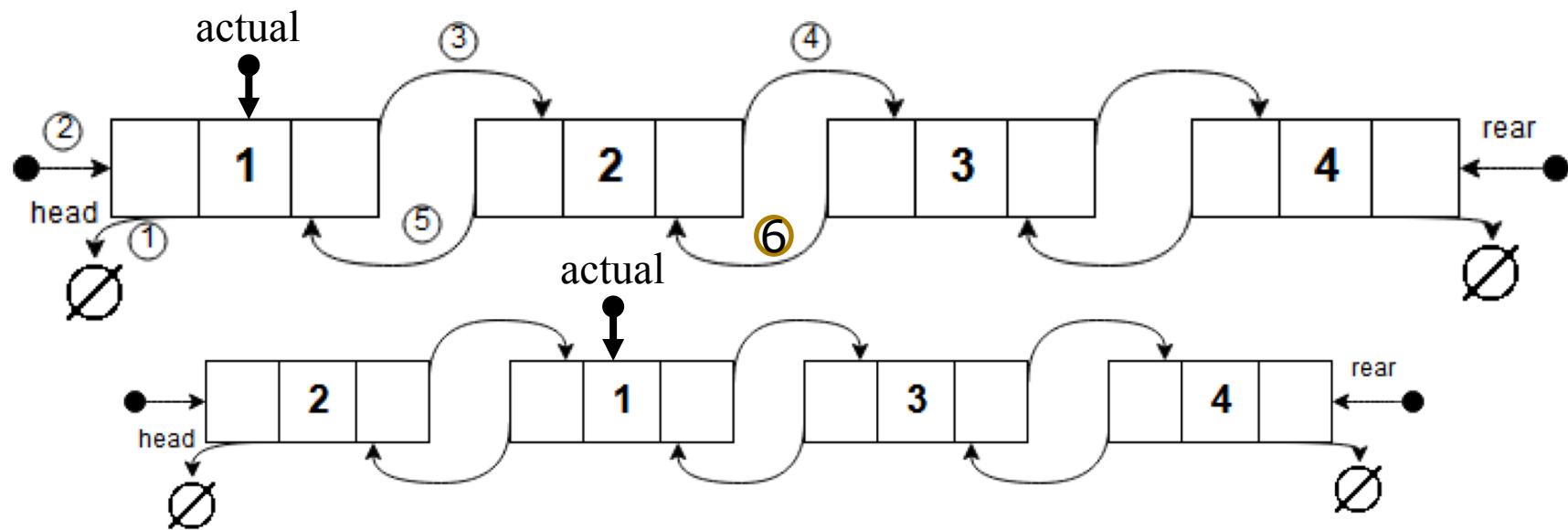


Exercicis de Punters

Punters 1

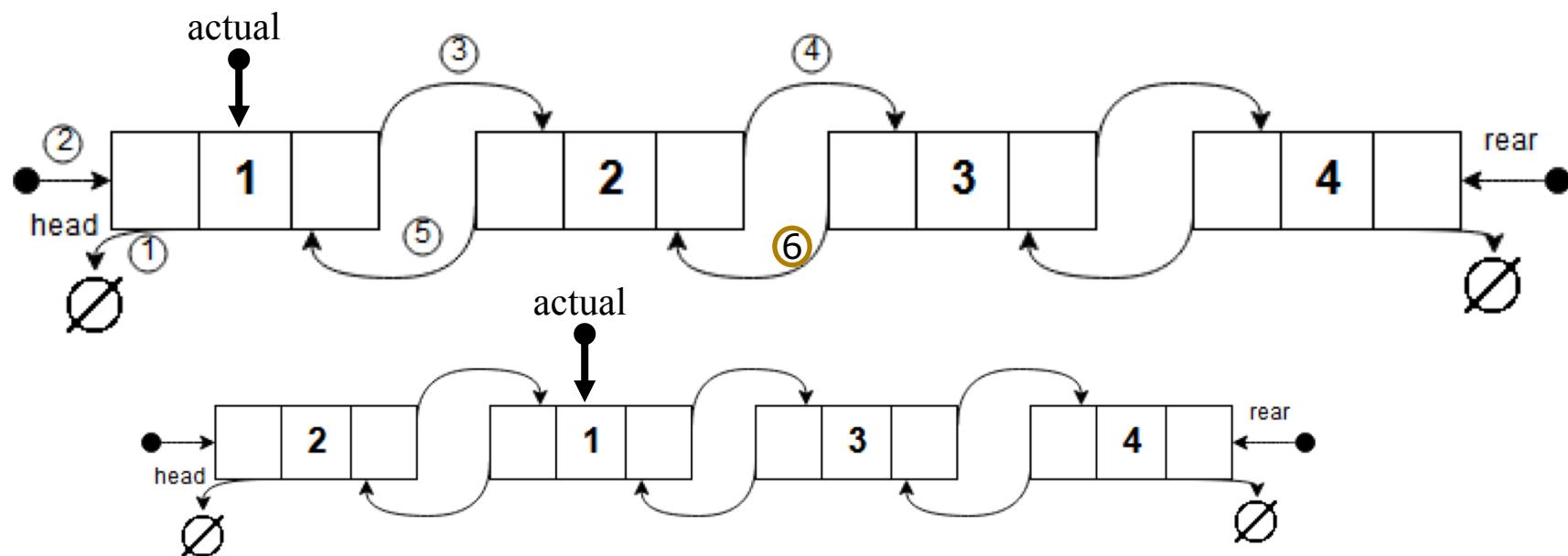
Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (els nodes amb atributs públics: **next**, **prev**, **elem**), volem intercanviar les posicions dels nodes amb elements 1 i 2 seguit els passos descrits a la imatge. (**No volem moure els continguts, volem moure els nodes**)

- 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 punter actual.**



Punters 1 (SOLUCIÓ)

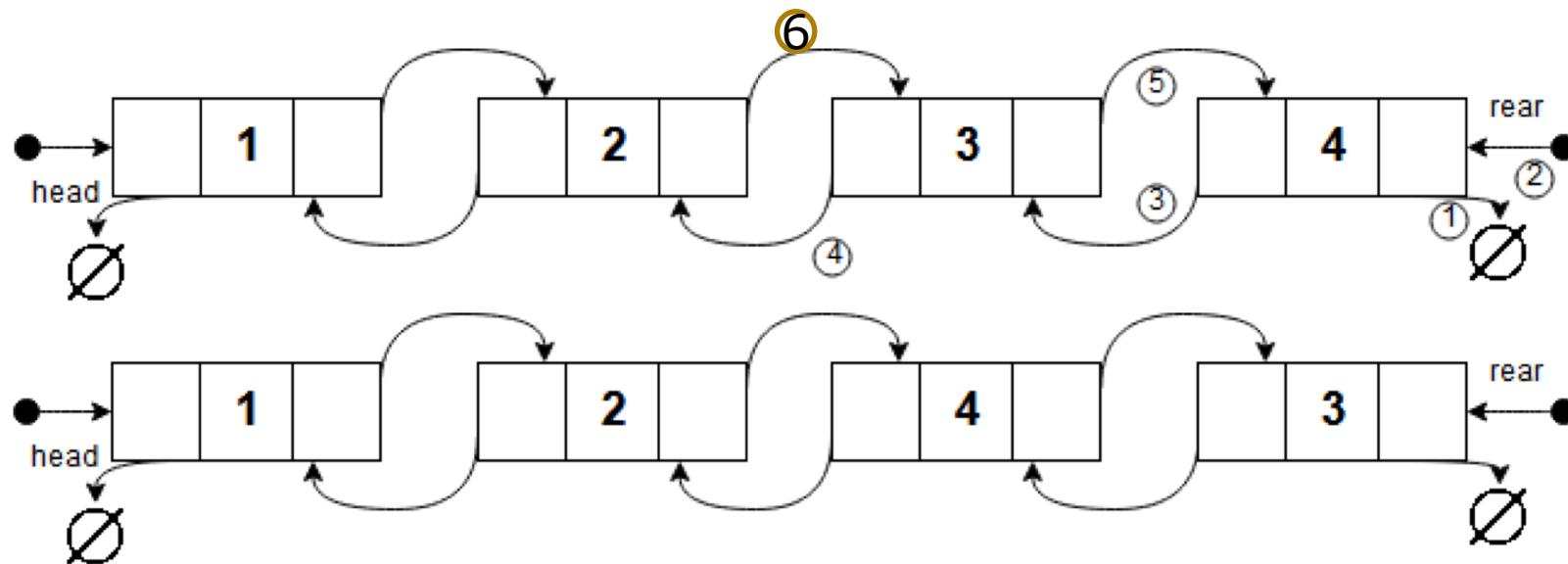
1. `actual->prev=actual->next;`
2. `head= actual->next;`
3. `actual->next= actual->next->next; o actual->next = actual->prev->next`
4. `actual->prev->next=actual;`
5. `actual->prev->prev=nullptr;`
6. `actual->next->prev=actual;`



Punters 2

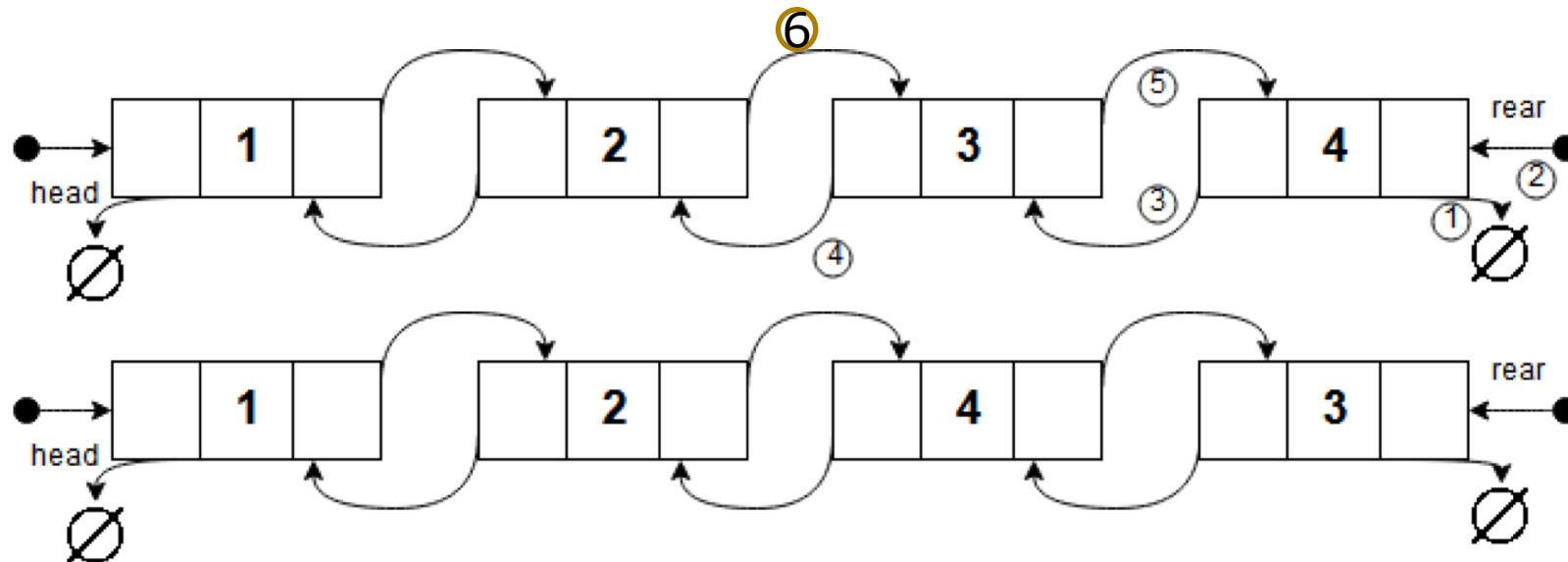
Sigui la implementació d'una seqüència d'elements amb nodes doblement encadenats (amb atributs públics: **next**, **prev**, **elem**), volem intercanviar les posicions dels nodes amb elements 3 i 4 seguint els passos descrits a la imatge. **(No volem moure els continguts, volem moure els nodes)**

- 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 rear.**



Punters 2 (SOLUCIÓ)

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



Exercicis ArrayStack

Classe ArrayStack

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

using namespace std;

class ArrayStack {
private:
    enum { DEFAULT_MAX_STACK = 10};

    int _stack[DEFAULT_MAX_STACK]; // stack
    int _top; // _top

public:
    ArrayStack(); // constructor

    int size () const; // return the size
    bool isEmpty() const; // return TRUE if the stack is empty

    void popEspecial(); // pop the secondElement of the stack
    void popFinal(); // pop the first element introduced to the stack
    void pushEspecial(const int&e); // push the element in the secondPosition
    void pushFinal(const int &e); // push the element at the end
};

ArrayStack::ArrayStack() { this->_top = -1; }

int ArrayStack::size () const { return (_top+1); }

bool ArrayStack::isEmpty() const { return (_top<0); }
```

El darrer element
de la pila es troba
a la posició 0

Noteu que a
l'especificació de
ArrayStack no estan els
mètodes push, pop, top,
i per tant,

Només podeu usar els
mètodes que hi ha a la
classe

popEspecial

- Implementeu el mètode **popEspecial** de l'ArrayStack que permet eliminar de la pila el segon element començant pel top.
- Per exemple, si la pila té [1,2,3,4] on 4 és l'element del top de la pila, després de cridar al mètode popEspecial, la pila quedarà com [1,2,4].
- Tingueu present que no sempre la pila estarà plena
- Llanceu les excepcions que creieu oportunes.

popEspecial

```
void ArrayStack::popEspecial()
{
    if (this->isEmpty()) throw out_of_range("popEspecial(): empty stack");
    else if (_top<1) throw out_of_range("popEspecial(): nomes 1 element");
    else
    {
        int top_actual = _stack[_top];
        --_top;
        _stack[_top] = top_actual;
    }
}
```

popFinal

- Implementeu el mètode `popFinal` de l'ArrayStack que permet eliminar de la pila el darrer element (és a dir, el primer element que s'havia afegit a la pila).
- Per exemple, si la pila té [1,2,3,4] on 4 és l'element del top de la pila, després de cridar al mètode `popFinal`, la pila quedarà com [2,3,4].
- Tingueu present que no sempre la pila estarà plena
- Llenceu les excepcions que creieu oportunes.

popFinal

```
void ArrayStack::popFinal()
{
    if (this->isEmpty())  throw out_of_range("popFinal(): empty stack");
    else
    {
        for (int i= 0; i < _top; i++)
        {
            _stack[i] = _stack[i+1];
        }
        --_top;
    }
}
```

pushEspecial

- Implementeu el mètode **pushEspecial** de l'ArrayStack que permet inserir un nou element després del top de la pila.
- Per exemple, si la pila té [1,2,3,4] on 4 és l'element del top de la pila, després de cridar al mètode pushEspecial(5), la pila quedarà com [1,2,3,5,4].
- Tingueu present que no sempre la pila contindrà elements i que té un màxim de capacitat
- Llenceu les excepcions que creieu oportunes.

pushEspecial

```
void ArrayStack::pushEspecial(const int& e)
{
    if (size() == this->DEFAULT_MAX_STACK)
        throw out_of_range("pushEspecial(): full stack");
    else if (this->isEmpty())
        throw out_of_range("pushEspecial(): empty stack");
    else {
        int top_actual = _stack[_top];
        _stack[_top] = e;
        _stack[+_top] = top_actual;
    }
}
```

pushFinal

- Implementeu el mètode **pushFinal** de l'ArrayStack que permet inserir un nou element al fons de la pila.
- Per exemple, si la pila té [1,2,3,4] on 4 és l'element del top de la pila, després de cridar al mètode `pushFinal(5)`, la pila quedarà com [5,1,2,3,4].
- Tingueu present que no sempre la pila contindrà elements i que té un màxim de capacitat
- Llenceu les excepcions que creieu oportunes.

pushFinal

```
void ArrayStack::pushFinal(const int& e)
{
    if (size() == this->DEFAULT_MAX_STACK)
        throw out_of_range("ArrayStack::pushFinal(): full stack");
    else {
        for (int i = _top; i>=0;i--){
            _stack[i+1] = _stack[i];
        }
        _stack[0]= e;
        ++_top;
    }
}
```