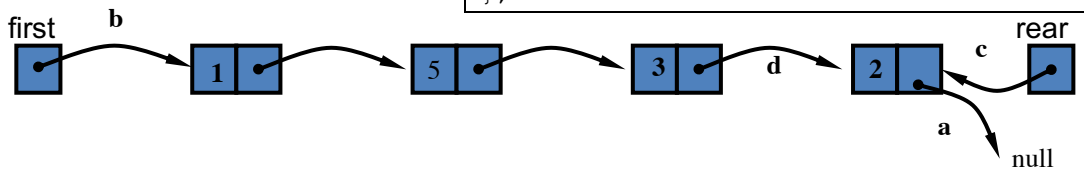
Problemes **PARCIAL 2017-2018** **Temps: 30 minuts**

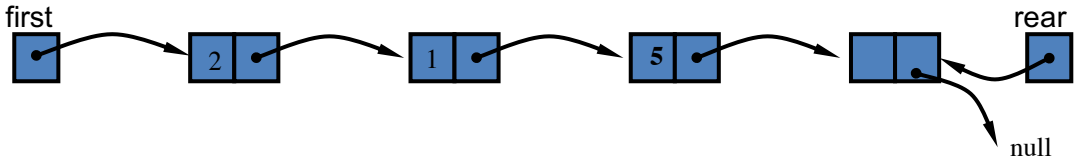
1. **Partim d'una cua enllaçada (`LinkedQueue`) amb una representació amb enllaços simples (`SingleNode`). Aquesta `LinkedQueue` té un punter al primer element de la cua anomenat `first` i un punter al darrer element de la cua anomenat `rear`. Implementeu el mètode de la classe LinkedQueue anomenat `void priorityLast()`, que permet moure el darrer element de la cua a la primera posició de la `LinkedQueue`. L'especificació de la classe `SingleNode` és la següent:**

```
class SingleNode
{   private:
        int element;
        SingleNode *next;
    public:
        SingleNode(int e);
       ~SingleNode();
        const int & getElement() const;
        SingleNode * getNext()const;
        void setNext(SingleNode * elem);
};
```

Inici



Final



Per a resoldre el problema, se us dona una part del codi implementat. Ompliu els forats que falten.

```
1. void LinkedQueue::priorityLast()
2. {
3.    If (this->empty() && this->size() < 3) throw out_of_range("LinkedQueue exception");
      ó   if (this->first == null) o (this->rear = null ) // les dije en el examen que no tenia size()
4.     else
5.     {
6.       Node<Element> * tmp;
7.       tmp = first;  // guarda a tmp una adreça
8.       while (tmp->getNext()!= rear) //Bucle per buscar la posició pel tmp
      ó while (tmp->getNext()->getNext() != null      {
                    tmp = tmp->getNext();
9.    }
10.      rear->setNext(first); o tmp->getNext()->setNext(first);// moure punter a
11.      first = rear o first = tmp->getNext() // moure punter b
12.      rear = tmp;          // moure punter c
13.      tmp->setNext(null); // moure punter d
14.    }
15.   }
```

2. Donades dues piles ordenades A i B (el mínim al top de la pila), implementa un mètode en C++ que a partir de les dues piles ordenades creï una nova pila ordenada (el mínim al top de la pila) combinant tots els elements de les dues piles anteriors. Només pots usar les operacions del TAD `Stack`. No es permet l'ús d'altres estructures de dades, només de piles. En aquest cas, teniu una part del codi però no acaba de funcionar. L'objectiu és que arregleu els errors en aquest codi.

```
1  Stack<int> mergeSortedStacks(Stack<int> A, Stack<int> B)
2  {
3  Stack<int> sol;
4  while (not A.empty() or not B.empty()){
5          if (A.empty()) { sol.push(A.top()); A.pop();  }
6          else if (B.empty()){ sol.push(B.top(); B.pop(); }
7          else {
8                  if (A.top() >= B.top()){  // Correcte, no modificar aquesta linia
9                          sol.push(A.top());
10                         A.pop();
11                 }
12                 else {
13                         sol.push(A.top());
14                         A.pop();
15                 }
16         }
17 }
18 if (not sol.empty())
19 {
20         while (not sol.empty()){ A.push(sol.pop()); sol.top(); }
21 }
22 return A;
23 }
```

Indiqueu a sota la solució per fer funcionar aquest programa correctament. Reescriu a sota <u>NOMES les línies que s'han de modificar.</u>

```
1  Stack<int> mergeSortedStacks(Stack<int> A, Stack<int> B)
2  {
3  Stack<int> sol;
4  while (!A.empty() || !B.empty())
5          if (A.empty() { sol.push(B.top()); B.pop();  }
6          else if (B.empty()){ sol.push(A.top(); A.pop(); }
7          else {
8                  if (A.top() >= B.top()){  // Correcte, no modificar aquesta linia
9                          sol.push(B.top());
10                         B.pop();
11                 }
12                 else {
13
14
15                 }
16         }
17 }
18 if (!sol.empty())
19 {
20         while (!sol.empty()){ A.push(sol.top()); sol.pop();}
21 }
22 return
23 }
```