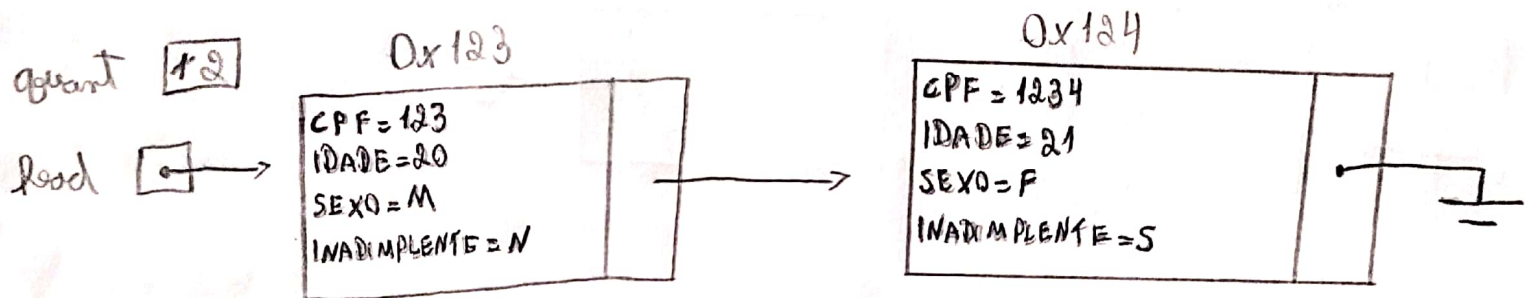


Método Insert

```

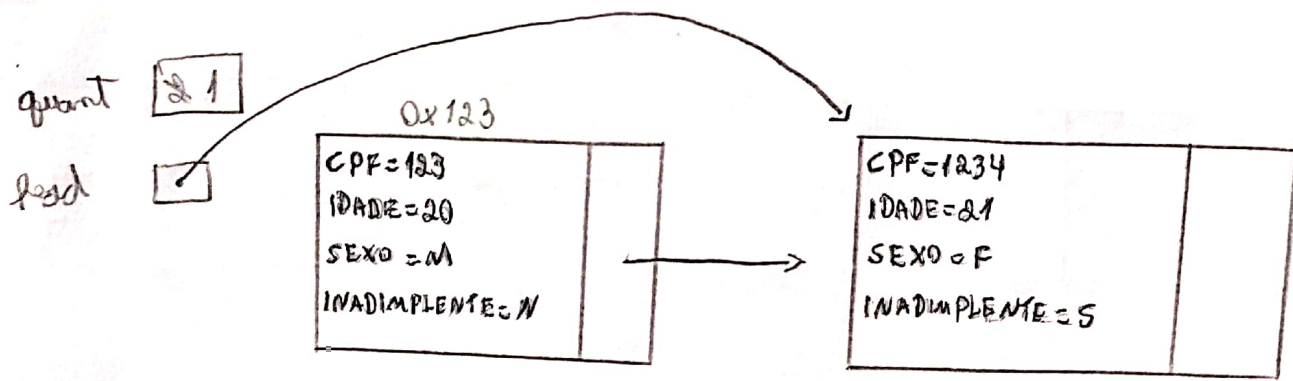
1 void Encodedata::insert() {
2     Pessoa p;
3     p.preencherCampos();
4     Node * novo = new Node();
5     novo -> setItem(p);
6     novo -> setProx(head);
7     head = novo;
8     quant++;
9 }
    
```



Método Remove

```

1 void Encodedata::remove() {
2     head = head -> getProx();
3     quant--;
4 }
    
```

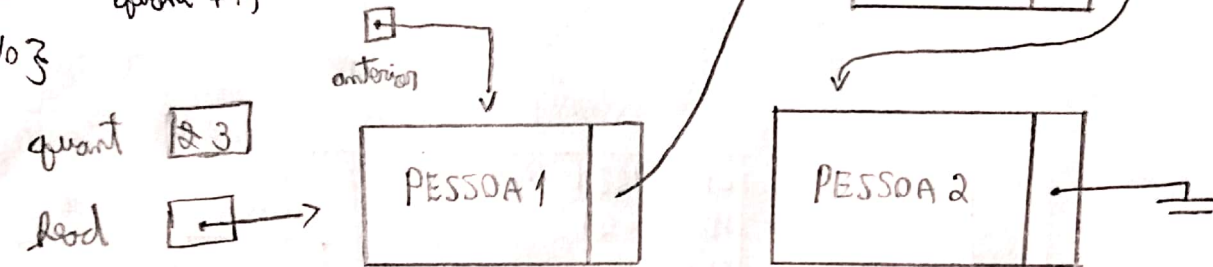


Método Insert na Lista

```

1 void Encadeada::insertPessoa (int n){
2     Pessoa p;
3     p.preencherEmpres();
4     Node* nove = new Node();
5     nove -> setItem (p);
6     Node* anterior = this -> getElemento (n-1);
7     nove -> setProx (anterior -> getProx());
8     anterior -> setProx (nove);
9     quant ++;
10 }

```

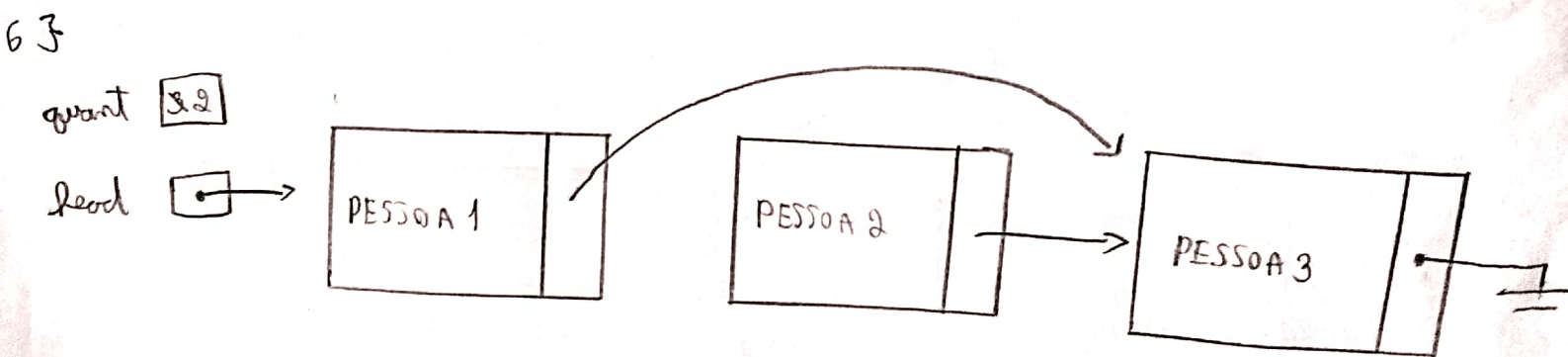


Método Remove na Lista

```

1 void Encadeada::removePessoa (int n){
2     Node* anterior = this -> getElemento (n-1);
3     Node* suiva = anterior -> getProx();
4     anterior -> setProx (suiva -> getProx());
5     quant --;
6 }

```



1 Node * Encodeda :: get Elementa (int n) {

2 Node * p = root;

3 int i = 1;

4 while (i <= n-1 && p->getProx() != NULL) {

5 p = p->getProx();

6 i++;

7 }

8 if (i == n) {

9 return p;

10 } else {

11 return NULL;

12 }

13 }

$\rightarrow i \leq 4$

i = 1

$\rightarrow i \leq 4$

i = 2

$\rightarrow i \leq 4$

i = 3

$\rightarrow i \leq 4$

i = 4

$\rightarrow i \leq 4 \rightarrow \text{false}$

i = 5

p [0x123]

p [0x124]

p [0x125]

p [0x126]

p [0x127]

p [0x127]

\rightarrow Sai do while e retorna +

quant [6]

0x123

0x124

0x125

0x126

0x127

0x128

head [0x123]

[] [0x124]

[] [0x125]

[] [0x126]

[] [0x127]

[] [0x128]

[] [NULL]