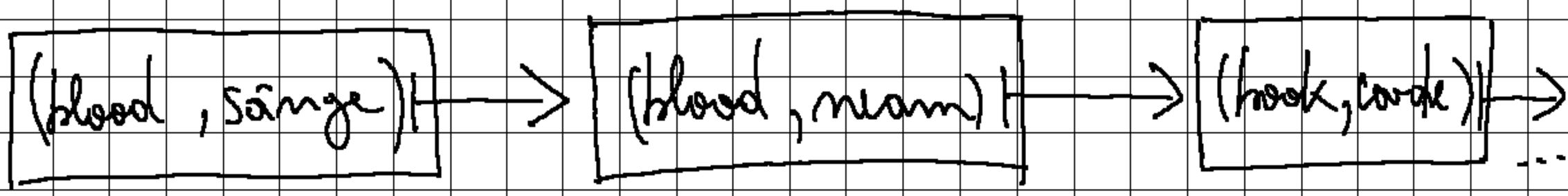
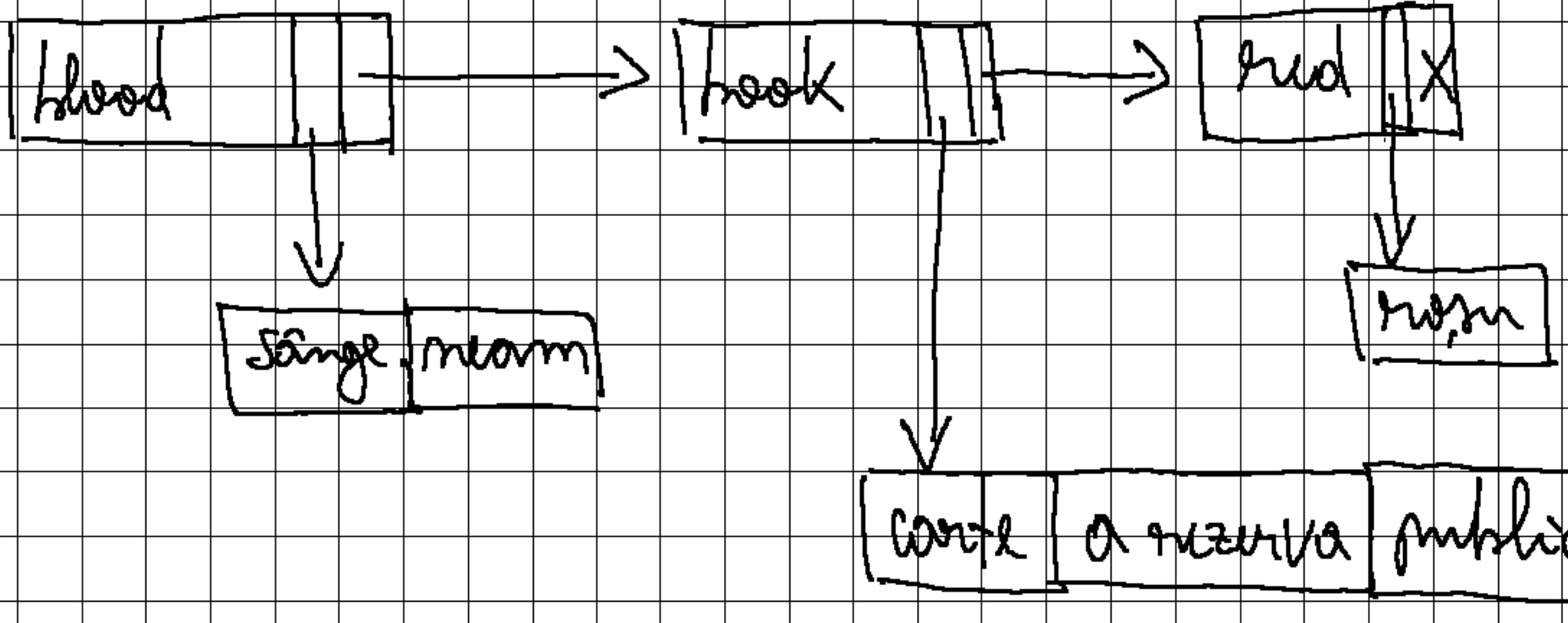
implement a SortedMultiMap using a singly linked representation

Eg: book: carte, a rezerva, publicatie
blood: sânge, neam
red: roșu

types (:



types 2:



SMMap:

head: ↑SLNode

R: relation

$$R(k_1, k_2): \begin{cases} \text{true}, \text{ if } k_1 \leq k_2 \\ \text{false}, \text{ otherwise} \end{cases}$$

SLNode:

info: TElem

next: ↑SLNode

TElem:

k: TKey

v: T

```
subalg init (Smm, R):
    Smm.head ← NIL
    Smm.R ← R
endsubalg

Subalg distroy (Smm):
    p ← Smm.head
    while p != NILL:
        Smm.head ← [p].next
        distroy ([p].info.vL)
        free (p)
        p ← mm.head
    endwhile
endsubalg

Subalg searchNode (Smm, k, kNode, prevNode)
                            ↓       ↓         ↓
                          book   (↑)book   ↑blood
                          apple   NIL       NIL
                          green   NIL      ↑book
                          red    (↑)red    ↑book
```

```
currNode ← Smm.head()
kNode ← NIL
prevNode ← NIL
while currNode execute
    if Smm.R([currNode].info.K, K)  AND
       ([currNode].info.K ≠ K) then
        prevNode ← currNode
    endif
    if [currNode].info.K == K
        kNode ← currNode
    endif
    currNode ← [currNode].next
endwhile
endSubalg
```

```
Subalg  Search (Smm , k , l)
    SearchNode (Smm , k , kNode, prevNode)
    if kNode = Nil then
        Init(l)
    else
        l ← ⌈kNode⌉.info.nl
    endif
endSub.

Subalg  addNode (Smm , k , v)
    SearchNode (Smm, k, kNode, prevNode)
    if kNode = Nil then
        allocate(p)
        if prevNode = Nil
            ⌈p⌉.next ← Smm.head
            Smm.head ← p
        else
            ⌈p⌉.next ← ⌈prevNode⌉.next
            ⌈prevNode⌉.next ← p
        endif
        init (⌈p⌉.info.nL)
        kNode ← p
    endif
```

```
    addEnd( [kNod].info.vL , v)
  endSubalg

Subalg  remove (Smm , k , v)
  searchNode( Smm, k , kNoode , prevNoode
    if  kNode == NiL then
      return false
    endif
    if search ([kNode].info.vL , v) then
        return false
    endif
    p ← position ( [kNode].info.vL , v)
    remove ( [kNode].info.vL , p , v)
    if isEmpty([kNode].info.vL)
      if prevNode = NiL then
          Smm.head ← [kNode].next
      else
          [prevNode].next ← [kNode].next
      endif
      destroy ([kNode].info.vL)
      free (kNode)
    endif
  endSubalg
```

```
SMM Iterator:
    smm: SMMap
    currentN: ↑SLNode
    itL: IteratorList

    subalg next(itr)
    if itr.currentN = NIL then
        throw exception
    endif
    next(itr.itL)
    if valid(itr.itL) then
        currentN ← currentN.next
        if current != NIL then
            itr.itL ← iterator([itr.currentN].info.vL)
        endif
    endsubalg
```