ADT List

Domain:

L = { l | l is a list with elements of type *TElem*, each element having a unique position of type *TPosition* in l}

Operations:

- **init(l)**
  pre: true
  post: $l \in L, l = \phi$
- **element(l, p, e)**
  pre: $l \in L, p \in TPosition, valid(p)$
  post:
  $e \in TElem$
  $e = the\ element\ on\ position\ p\ in\ l$
  @throws exception if *p* is not valid
- **position(l, e)**
  pre: $l \in L, e \in TElem$
  post:
  $$p = \begin{cases} position \leftarrow p \in TPosition, \\ first\ position\ of\ e\ from\ l, \\ \quad if\ e \in l \\ \perp, otherwise \end{cases}$$
- **modify(l, p, e)**
  pre: $l \in L, p \in TPosition, valid(p), e \in TElem$
  post: the element from position *p* from l' = e
  @ throws exception if *p* is not valid
- **addFirst(l, e)**
  pre: $l \in L, e \in TElem$
  post: e was added to the beginning of l
- **addEnd(l, e)**
  pre: $l \in L, e \in TElem$
  post: e was added to the end of l
- **addAfter(l, p, e)**
  pre:
  $l \in L, p \in TPosition, valid(p), e \in TElem$
  post: e was inserted in l' after position *p*
  @throws exception if *p* is not valid
- **addBefore (l, p, e)**
  pre:
  $l \in L, p \in TPosition, valid(p), e \in TElem$
  post: e was inserted in l' before position p
  @ throws exception if *p* is not valid
- **remove(l, p, e)**
  pre: $l \in L, p \in TPosition, valid(p)$
  post: $e \in TElem$, element *e* from position *p* was removed from *l'*.
  @ throws exception if *p* is not valid
- **search(l, e)**
  pre: $l \in L, e \in TElem$
  post: $search = \begin{cases} true, if\ e\ is\ in\ l \\ false, otherwise \end{cases}$
- **isEmpty (l)**
  pre: $l \in L$
  post: $isEmpty = \begin{cases} true, if\ l = \phi \\ false, otherwise \end{cases}$
- **size(l)**
  pre: $l \in L$
  post: $size = n, n \in Natural$
  n =the number of elements of *l*
- **destroy(l)**
  pre: $l \in L$
  post: l was "destroyed" (allocated memory was freed)
- **iterator(l, it)**
  pre: $l \in L$
  post: $it \in I$, *it* is an iterator on list *l*

ADT Sorted MultiMap

Domain

SMM = {smm | smm is a Sorted Multimap with pairs *TKey*, *TValue*, where we can define a relation R on the set of all possible keys}

Operations:

- **init (smm, R)**
  pre: R – relation on the set of all possible keys
  post: $smm \in SMM, smm = \phi$
- **destroy(smm)**
  pre: $smm \in SMM$
  post:smm was destroyed (allocated memory was freed)

- **add(smm, k, v) –** can be called put or insert
  pre: $smm \in SMM, k \in TKey, v \in TValue$
  post: the pair <k,v> was added into smm.

- **remove(smm, k, v)**
  pre: $smm \in SMM, k \in TKey, v \in TValue$
  *post:*the pair <k,v> was deleted from smm (if it was in smm).

- **search(smm, k, l)**
  pre: $smm \in SMM, k \in TKey$
  $l \in L,$
  post: $\rightarrow \begin{cases} true \text{ and } l \text{ is the list of values associated with } c, \\ \quad if \ c \text{ is in } smm \\ false \text{ and } l = \phi, otherwise \end{cases}$
- **iterator(smm, it)**
  pre: $smm \in SMM$
  post: $it \in I$, *it* is an iterator over *smm*

Other possible operations:

- **keySet(smm, m)**
  pre: $smm \in SMM$
  post: $m \in M$, *m* is the set of all keys from smm

- **valueBag(smm, b)**
  pre: $smm \in SMM$
  post: $b \in B, b$ is the collection of all values from smm