Azar Muradli **1. assignment/0. task** 25th March 2023
EZEEEL
ezeeel@inf.elte.
huGroup 4

Task

Implement the bag type which contains integers. Represent the bag as a sequence of (element, frequency) pairs. Implement as methods: inserting an element, removing an element, returning the frequency of an element, returning the number of elements which occur only once in the bag (suggestion: store the number of these elements and update it when the bag changes), printing the bag.

Bag type

Set of values

Bag = {items: Item^n , N | ($\forall$i$\in$[1..n],$\forall$j$\in$[1..n]: a[i].element $\neq$ a[j].element) $\wedge$
($\forall$i$\in$[1.. n] : a[i].frequency>0}
Item =(element $\in \mathbb{Z}$, frequency $\in$ N)

Operations

1. Inserting an element
Inserting an element into a bag. If the element exists in the bag, then increase its frequency. If the element does not exist, just insert a new pair with frequency 1.

Formally:
A : { *items*$\in$Item^n,Item=(element x frequency), element $\in \mathbb{Z}$, frequency $\in$ N, n $\in$ N ,e $\in$ Z, index $\in$ N,numOfOneOccur $\in$ N }
Pre = (*items=items' $\wedge$ e=e' $\wedge$ index=index'*)
Post = (*Pre* $\wedge$ *Exists:* $\exists$i $\in$1..n]: items[i].element = e $\rightarrow$ index := i
Exists $\rightarrow$ items[index].frequency = items[index].frequency + 1
items[index].frequency = 2 $\rightarrow$ numOfOneOccur = numOfOneOccur – 1
Otherwise $\rightarrow$ items[n+1].element = e
items[n+1].frequency = 1
numOfOneOccur = numOfOneOccur + 1
)

2. Removing an element

If element exists in bag, and its frequency is more than 1, then reduce its frequency. If its frequency is 1, then remove an element from bag. If bag doesn't contain that element, throw an exception

A : { *items*$\in$Item^n,Item=(element x frequency), element $\in \mathbb{Z}$, frequency $\in$ N, n $\in$ N e $\in$ Z, index $\in$ N,numOfOneOccur $\in$ N }

Pre = (*items=items'* ∧ *e=e'* ∧ *index=index'*)
Post = (*Pre* ∧ *Exists:* ∃i ∈1..n]: items[i].element = e → index := i
Exists → items[index].frequency = 1 → items.remove(items[index])
numOfOneOccur = numOfOneOccur -1
otherwise → items[index].frequency = items[index].frequency -1
items[index].frequency = 1 → numOfOneOccur =
numOfOneOccur + 1
Otherwise → throw BagDoesntContainThisElement
)

3. Frequency

Getting the frequency of the element. If the element does not exist, then return zero. If the element exists, return the frequency of that element.

A : { *items*∈Item^n,Item=(element x frequency), element ∈ ℤ , frequency ∈ N, n ∈ N ,e ∈ Z, index ∈ N, res ∈ N
}

Pre = (*items=items'* ∧ *e=e'* ∧ *index=index'*)
Post = (*Pre* ∧ *Exists:* ∃i ∈1..n]: items[i].element = e → index := i,
Exists → res := items[index].frequency
Otherwise → res:=0
)

4. getNumOfOneOccur

Getting the number of elements with frequency one.

A : { $items \in Item^n$,Item=(element x frequency), element $\in \mathbb{Z}$, frequency $\in$ N, n $\in$ N , numOfOneOccur $\in$ N }

Pre = -

Post = ($\exists$i $\in$1..n]: numOfOneOccur = $\sum_{i=1}^{n}$ 1)
items[i].frequency = 1

Representation

Bag type is represented by a list with elements of type Item, where Item is a class containing the element and its frequency.

Bag = Item1 (element, frequency), Item2 (element, frequency), Item3 (element, frequency), Item4(element,frequency)

Bag = (element1,frequency1),(element2,frequency2),(element3,frequency3),(element4,frequency4)

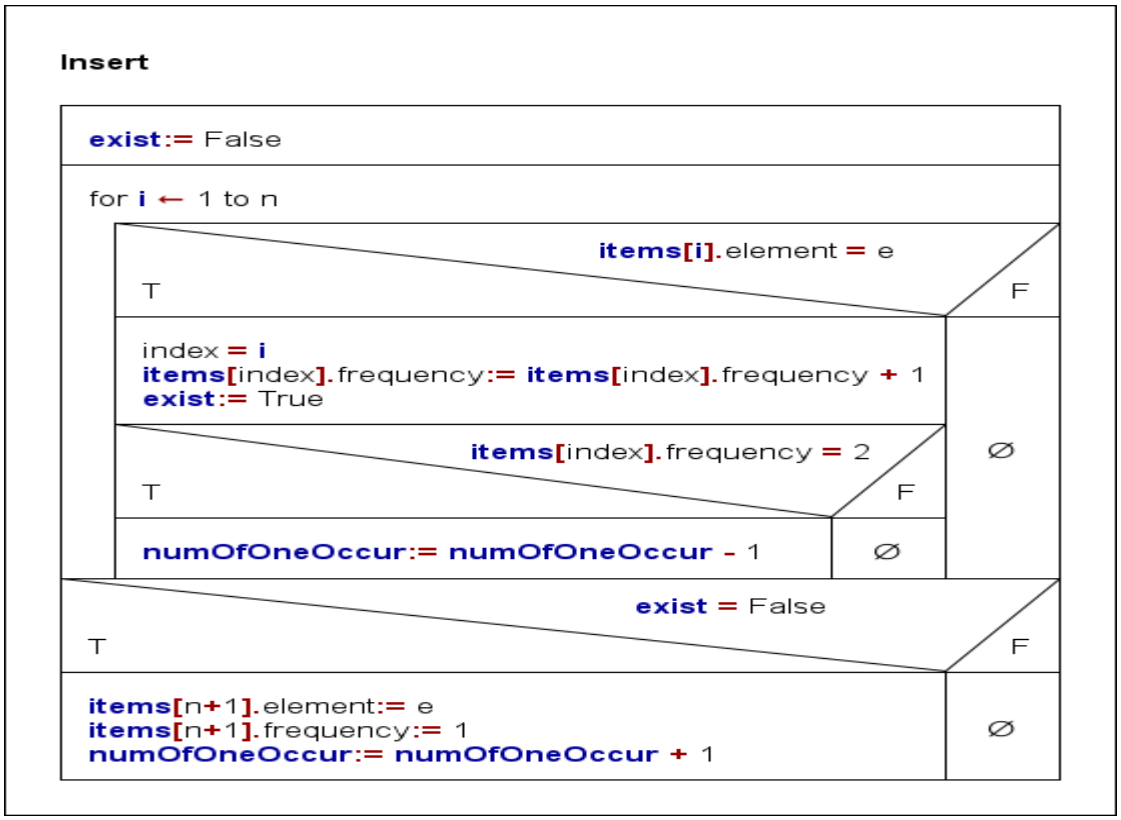
Frequency can only be positive numbers, and this number can be as big as user want there is no restriction.

Additionally, the bag can't have two items with the same elements.

Implementation[1]

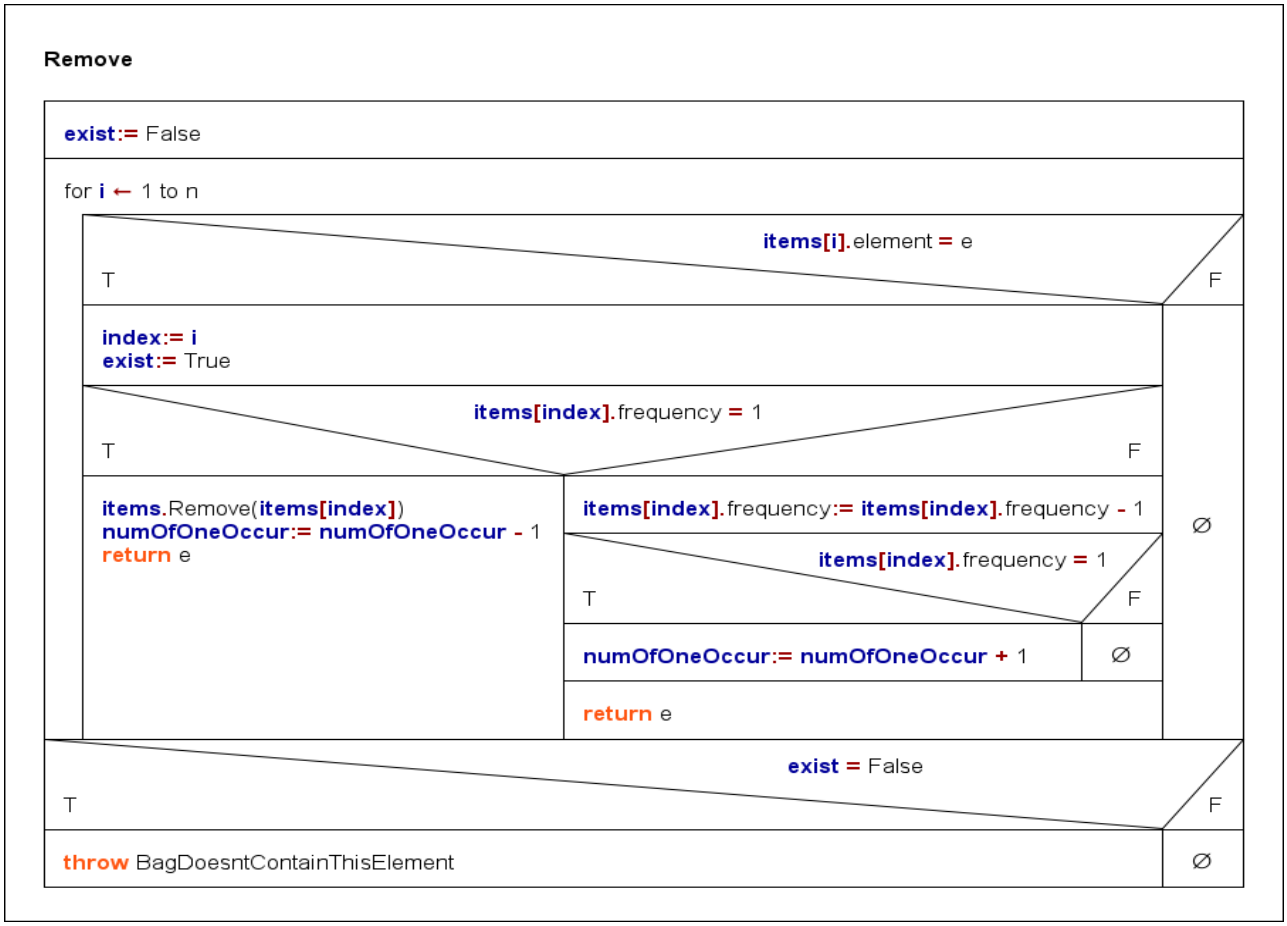
1. Inserting an element

Inserting an element into a bag. If the element exists in the bag, then increase its frequency. If the element does not exist, just insert a new pair with frequency 1.



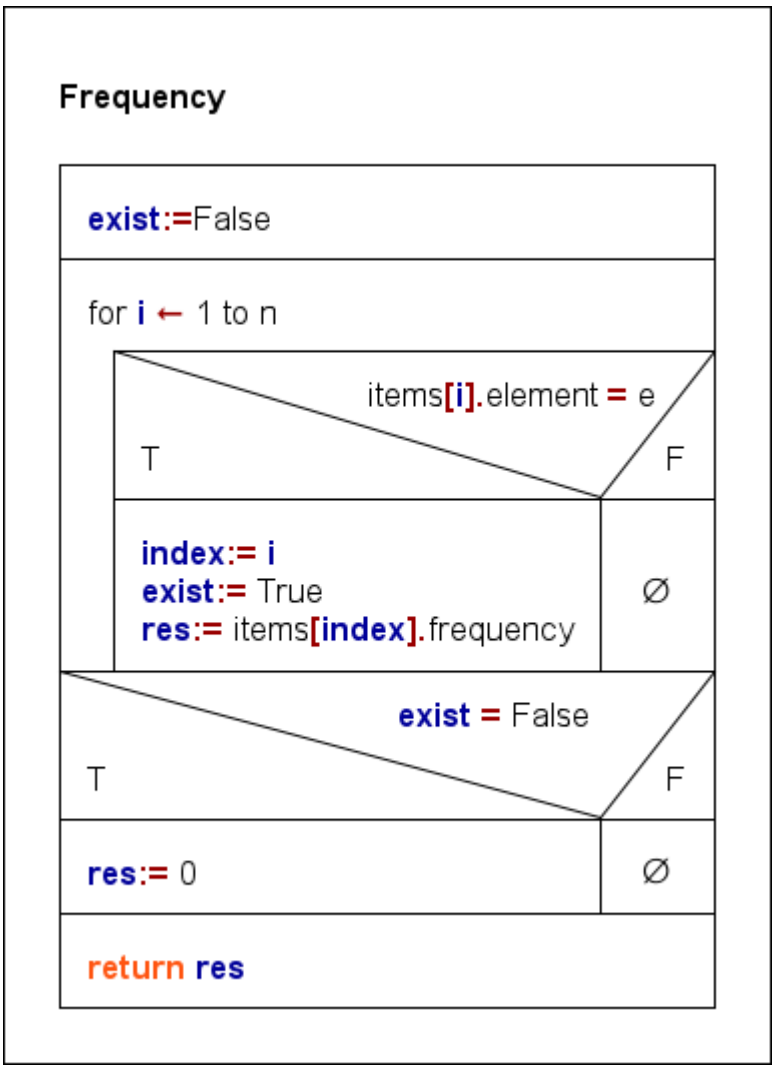
2.Removing an element

If element exists in bag, and its frequency is more than 1, then reduce its frequency. If its frequency is 1, then remove an element from bag. If bag doesn't contain that element, throw an exception



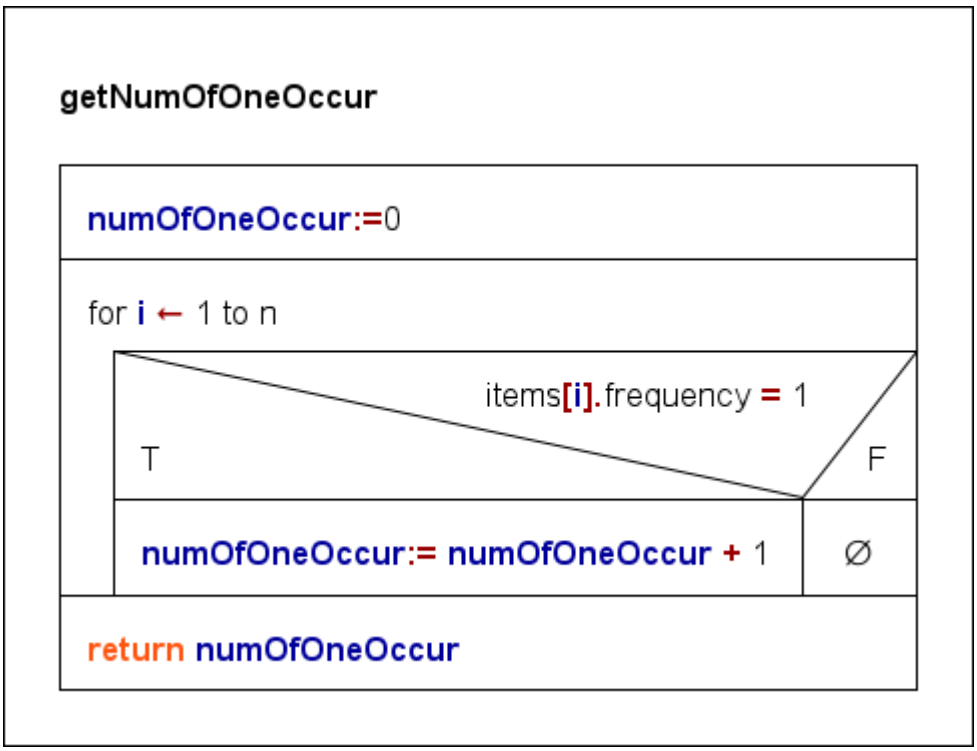
3. Frequency

Getting the frequency of the element. If the element does not exist, then return zero. If the element exists, return the frequency of that element.



getNumOfOneOccur

Getting the number of elements with frequency one.



[1] To implement an operation, a program has to be given (not necessarily structogram).

Testing

Testing the operations (black box testing)

1. Inserting an element
    a. Inserting new element
    b. Inserting existing element
    c. Checking frequency of element
    d. Checking the number of elements with one occurrence

2. Removing an element
    a. Removing existing element
    b. Removing non-existing element
    c. Checking exception

3. Checking Frequency
    a. Getting frequency of existing element
    b. Getting frequency of a non-existing element

4. Checking getNumOfOneOccur
    a. Checking number of elements with one occurrence

Testing based on the code (white box testing)
1. Generating and catching exceptions.