Map - representation on a hash table - collision resolution with coalesced chaining

Assume:
- we memorize the keys only
- the keys are integer numbers

Example:
- Map with the following keys: 5, 18, 16, 15, 13, 31, 26
- HT with m= 13 positions and a hash function with the division method

k	5	18	16	15	13	31	26
h(k)	5	5	3	2	0	5	0

Representation:

Map:
    elems: TKey[]
    next: Integer[]
    firstEmpty: Integer
    m: Integer
    h: TFunction

subalgorithm init(map) is:
    map.m = 1
    @allocate map.next m positions
    @allocate map.elems m positions
    @initialize map.h
    map.next[0] = -1
    map.elems[0] = -1
    map.firstEmpty = 0
end-subalgorithm
Complexity: Theta(1)

function search(map, k) is:
//returns true if k is in the map and false otherwise
    Pos <- map.h(k)
    While pos != -1 AND map.elems[pos] != k execute:
        Pos <- map.next[pos]
    End-while
    If pos = -1 then

```
            Search <- false
        Else
            Search <- true
        End-if
end-function
```
Complexity: O(n) (Theta(1) on average)

Remove:

Example 1:

Insert: 11, 8, 3

	0	1	2	3	4
elems	3	11		8	
next	-1	-1	-1	0	-1

firstEmpty = 2

Remove 11

	0	1	2	3	4
elems	3			8	
next	-1	-1	-1	0	-1

firstEmpty = 1

Example 2:
Insert: 56, 8, 11, 12

	0	1	2	3	4
elems	11	56	12	8	
next	-1	0	-1	-1	-1

firstEmpty = 4

Remove 11

	0	1	2	3	4
elems		56	12	8	
next	-1	-1	-1	-1	-1

firstEmpty = 0

Example 3:
Insert: 11, 20, 56

	0	1	2	3	4
elems	20	11	56		
next	-1	2	-1	-1	-1

firstEmpty = 3

Remove 11

	0	1	2	3	4
elems	20	56			
next	-1	-1	-1	-1	-1

firstEmpty = 2

Example 4:

Insert: 11, 20, 56, 57

	0	1	2	3	4
elems	20	11	56	57	
next	-1	2	3	-1	-1

firstEmpty = 4

Remove 11

	0	1	2	3	4
elems	20	56	57		
next	-1	2	-1		-1

firstEmpty = 3

Example 5:

Insert: 56, 11, 12, 1

	0	1	2	3	4
elems	11	56	12	1	
next	3	0	-1	-1	-1

firstEmpty = 4

Remove 11

	0	1	2	3	4
elems		56	12	1	
next	-1	3	-1	-1	-1

firstEmpty = 0

```
function remove(map,k) is:
//return true of k was removed and false otherwise
        pos ← map.h(k)
        prevpos ← -1
        //find the key
        while pos != -1 AND map.elems[pos] != k execute
                prevpos ← pos
                pos ← map.next[pos]
        end-while
        if pos = -1 then
                //key is not in the map
                remove ← false
```

```
        else
                over ← false
                repeat
                        p ← map.next[pos]
                        pp ← pos
                        while p != -1 AND map.h(map.elems[p]) != pos execute
                                pp ← p
                                p ← map.next[p]
                        end-while
                        if p = -1 then
                                over ← true
                        else
                                map.elems[pos] ← map.elems[p]
                                pos ← p
                                prevpos ← pp
                        end-if
                until over
                if prevpos = -1 then
                        index ← 0
                        while index < map.m AND prevpos = -1 execute
                                if map.next[index] = pos then
                                        prevpos ← index
                                end-if
                                index ← index + 1
                        end-while
                end-if
                if prevpos != -1 then
                        map.next[prevpos] ← map.next[pos]
                end-if
                map.next[pos] ← -1
                map.elems[pos] ← -1
                if pos < map.firstEmpty then
                        map.firstEmpty ← pos
                end-if
                search ← true
        end-if
end-function
```