PWr 2013-03-15
Algorithms and Data Structures
Laboratory – **List 4**

1. Write a program with below operations for a two-way **ordered** cycled list with sentinel. The list will be of elements of type `Element`, with fields `key` and `value`.
    a. **`void`** `init(List2W& l)` – which initialize the list `l`.
    b. **`void`** `insertElem(List2W & l, Element elem)`- insert an element elem in a list `l`, depending on field `key` in type `Element`. If there is an element with this same key, insert the new one on last proper position.
    c. **`bool`** `findKey(List2W & l, int key, Element &elem)` - find first element in list `l` with value and assign to elem this element. Return `true` if element has been found, otherwise – `false`;
    d. **`void`** `removeAllKeys(List2W & l, int key)` – remove from list `l` all elements which key is equal to `key`. If there is no such element, do nothing.
    e. **`void`** `showListFromHead(List2W & l)` - show elements of list `l` starting from the head. The values are written in one line, after EVERY element (also the last) write a comma ‘,’. Every element have to be written in a format `key`(`value`). If a list is empty – the line is empty. The line ends with newline character. Example: “1(5),2(2),6(3),”
    f. **`void`** `showListFromTail(List2W & l)` - show elements of list `l` starting from the tail. Format like for `showListFromHead()`.
    g. **`void`** `clearList(List2W & l)` - remove all elements from list l.
    h. **`void`** `addList(List2W & l1, List2W & l2)` - move all elements from list `l2` to list `l1`. After adding list l1 has to be still ordered. If in list l2 elements have the same key, in result list, this elements have to be after element from list l1 with the key. After this operation the list `l2` is empty. If `l1` and `l2` are the same list – do nothing.

    Format of a stream on judgment system is presented in appendix 1. Prepare 2-3 interesting tests using this format.

For **10 points** present solutions for this list till **2013-03-26**.
For **7 points** present solutions for this list till **2013-04-09**.
After 2013-04-09 the list is closed.

Appendix 1

The solution will be automated tested with tests from console of presented below format. The test assumes, that there are up to X different lists, which there are created as the first operation in the test. Each list can be initialized separately.

If a line starts from '`#`' sign, the line have to be ignored.

If a line has a format:
`GO` *`n`*
your program has to create *n* lists (without initialization). The lists are numbered from 0 like an array of lists. Default current list is a list with number 0. This operation will be called once as the first command.

If a line has a format:
`CH` *`n`*
your program has to choose a list of a number `n`, and all next functions will operate on this list. There is *n>=0*.

If a line has a format:
`IN`
your program has to call `init(l)` for current list `l`. For any list this operation will be called once, before using the list.

If a line has a format:
`IE` *`k v`*
your program has to call `insertElement(l,x)` for current list `l`, and element x with field `key` equals `k`, and field `value` equals `value`.

If a line has a format:
`FK k`
your program has to call `findKey(l,k, el)` for current list `l`, and if the function return **`true`**, write on the output returned value `el` in format "key(value)". Otherwise write "false" with new line character.

If a line has a format:
`RK` *`k`*
your program has to call `removeAllKeys(l,k)` for current list `l`.

If a line has a format:
`SH`
your program has to call `showListFromHead(l)` for current list `l`.

If a line has a format:
`ST`
your program has to call `showListFromTail(l)` for current list `l`.

If a line has a format:

`CL`
your program has to call `clearList(l)` for current list `l`.

If a line has a format:
`AD n`
your program has to call `addList(l, l2)` for current list `l` and for list l2 which is the *n*'th list in the array of lists

If a line has a format:
`HA`
your program has to end the execution, writing as the last line "END OF EXECUTION". Every test ends with this line.

For example for input test:

```
GO 2
IN
IE 1 4
IE 4 1
IE 3 7
SH
CH 1
IN
IE 7 0
IE 0 1
IE 3 9
SH
AD
SH
CH 0
SH
HA
```

The output have to be:

```
1(4),3(7),4(1),
0(1),3(9),7(0),
0(1),1(4),3(9),3(7),4(1),7(0),

END OF EXECUTION
```