PWr 2013-02-28
Data Structures and Algorithms, summer semester 2012/2013
Laboratory – **List 2**

1. Implement **finite** queue with **only one** organizing index. Implement operation:
   a. **void** `init(Queue& q, int size)` – which initialize the queue `q`.
   b. **void** `enqueue(Queue& q, int value)` – which put a `value` on the end of queue `q`. If there is no place – do nothing;
   c. **void** `dequeue(Queue& q, int &value)` – which remove and return under `value` an element from the front of a queue `q`. If there is no element – do nothing;
   d. **bool** `isEmpty(Queue& q)` – return **true** if queue `q` is empty, otherwise - **false**;
   e. **bool** `isFull(Queue& q)` – return **true** if queue `q` is full, otherwise - **false**;
   f. **void** `show(Queue& q)` – show elements of a queue q starting from the front. The values are written in one line, separated by one space. If a list is empty – the line is empty. The line ends with newline character.

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

2. Write a program with all operation presented on the lecture for a **one-way unsorted linked** list:
   a. **void** `init(List& l)` – which initialize the list `l`.
   b. **void** `insertHead(List& l, int elem)`- insert an element `elem` as a head (first element) in a list `l`.
   c. **void** `deleteHead(List& l)`- remove a head (first element) from a list `l`.
   d. **void** `insertTail(List& l, int elem)`- insert an element `elem` as a tail (last element) in a list `l`.
   e. **void** `deleteTail(List& l)` - remove a tail (last element) from a list `l`.
   f. **int** `findValue(List& l, int value)` - find first element in list `l` with value and return its position (starting from 0). If there is no such element, return -1;
   g. **void** `deleteValue(List& l, int value)` – remove from list `l` first element which is equal to `value`. If there is no such element, do nothing.
   h. **int** `atPosition(List& l, int pos)` – find in list `l` an element on specified position `pos`. Return a value on this position. If a position does not exist, return -9999
   i. **void** `showListFromHead(List& l)` - show elements of list `l` starting from the head. The values are written in one line, separated by one space. If a list is empty – the line is empty. The line ends with newline character.
   j. **void** `clearList(List& l)` - remove all elements from list l.

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

For 10 points present solutions for this list till 2013-03-12.

For 7 points present solutions for this list till 2013-03-19.
After 2013-03-19 the list is closed.
Appendix 1 (for a queue).
The solution will be automated tested with tests from console of presented below format. The test assumes, that there are up to X different queues, which there are created as the first operation in the test. Each queue can be initialized with different size, and it will be done in consecutive operations.

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

If a line has a format:
`ST` *n*
your program has to create *n* queues (without initialization). The queues are numbered from 0 like an array of queues. Default current queue is a queue with number 0.

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

If a line has a format:
`IN` *n*
your program has to call `init(q, n)` for current queue `q`. There is *n>=1*. For any queue this operation will be called once.

If a line has a format:
EN *x*
your program has to call `enqueue(q,x)` for current queue `q`.

If a line has a format:
DE
your program has to call `dequeue(q,x)` for current queue `q` and write on output one line with value `x`.

If a line has a format:
EM
your program has to call `isEmpty(q)` for current queue `q`, and then depending on return value, write on output one line with text “true” or “false”.

If a line has a format:
FU
your program has to call `isFull(q)` for current queue `q`, and then depending on return value, write on output one line with text “true” or “false”.

If a line has a format:
SH
your program has to call `show(q)` for current queue `q`, which write one line on output with values separated by one space, e.a. string “4 6 1”

If a line has a format:

HA
your program has to end the execution.

For example for input test:

```
ST 2
IN 3
EN 1
EM
EN 3
EN 4
FU
EN 5
DE
SH
CH 1
IN 5
EM
FU
CH 0
EN 6
SH
HA
```

The output have to be:

```
false
true
1
3 4
true
false
3 4 6
```

Appendix 2 (for a linked list).
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:
`ST` *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.

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.

If a line has a format:
`IH` *x*
your program has to call `insertHead(l,x)` for current list `l`.

If a line has a format:
`DH`
your program has to call `deleteHead(l,x)` for current list `l`.

If a line has a format:
`IT` *x*
your program has to call `insertTail(l,x)` for current list `l`.

If a line has a format:
`DT`
your program has to call `deleteTail(l,x)` for current list `l`.

If a line has a format:
`FV` *x*
your program has to call `findValue(l,x)` for current list `l`, and write on output returned value.

If a line has a format:
`DV` *x*
your program has to call `deleteValue(l,x)` for current list `l`.

If a line has a format:

`AT` *`x`*
your program has to call `atPosition(l,x)` for current list `l`, and write on output returned value.

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

If a line has a format:
`CL` *`x`*
your program has to call `clearList(l)` for current list `l`.

If a line has a format:
`HA`
your program has to end the execution.

For example for input test:

```
ST 2
IN
IH 1
IH 2
IT 3
SH
FV 2
AT 0
DH
DT
FV 2
AT 2
HA
```

The output have to be:

```
2 1 3
0
2
-1
-9999
```