Programmeren (FEB22012)

Exercise 9

Deadline for submission: 2011-10-02 23:59 CET

This exercise is done **in pairs**. Submit your solution only under the account of one of the pair members, and remember to add comments containing names and student numbers of both pair members in all submitted source files. In case you do not feel like coding this with another human being, you can also do it alone. Be aware of the following:

- Submit a ZIP file (so no .tgz, rar, and any other compression format).
- Follow the instructions carefully, when you are asked to create a function then you are supposed to deliver a function file and not a script file.
- Make sure that your code has one script called `main.m`, which contains the code to "start" your program. This script should run with no errors, otherwise you are at risk of receiving 0 points for the assignment.

Introduction

In this assignment you will implement linked lists in Matlab and solve a classical problem using your implementation of a circular single-linked list. To achieve this, we need to apply the object-oriented extensions of Matlab.

Creating classes in Matlab is achieved similarly as writing function files (if you use the GUI, you can select "class" for adding them from the menu). Classes are accessed using handles that are similar to Java object references. For using classes without methods, we can define them to contain simply the fields (properties), e.g. for class `ExampleClass` with a single field `exampleField`:

```
classdef ExampleClass < handle
  properties
    exampleField;
  end
end
```

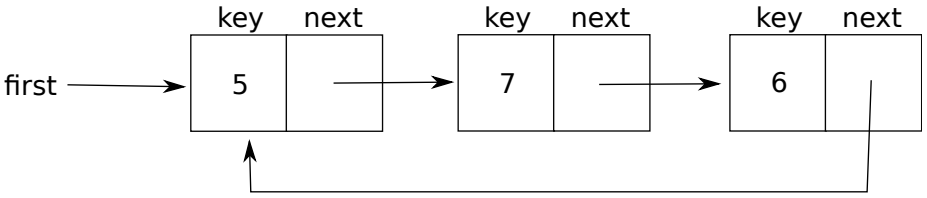
Multiple field variables can be defined by separating them with commas. Now we could instantiate this class and set the field value with:

```
>> t = ExampleClass();
>> t.exampleField = 3;
```

Note that in this exercise we will not actually program according to the object oriented paradigm, but just use the OO-extensions of Matlab to enable dynamic memory allocation and pass by reference. All the linked list operations will be implemented as separate methods.

Exercise, part 1: implementing circular linked list

In this part, you will implement linked lists in Matlab using objects. Start by copy-pasting the linked list code from the lecture notes and modifying it so, that you implement a *circular linked list* (see Figure below). Note that in a circular linked list when the "first" node is deleted, the "first" reference of the LinkedList object has to be updated as well. Implement also an additional function for computing the length of the list.



After changing the implementation, make sure that you have the following methods for operating with lists:

1. `initLinkedList(value)`, that initializes a new linked list
2. `insertIntoBeginning(L, value)`, that inserts value into the beginning of the list
3. `deleteNodeAfter(prevNode)`, that deleted the node coming after `prevNode`
4. `listLength(L)`, that gives the number of elements in the linked list `L`

Hint: you should keep track of the first node in your list, otherwise you do not know where the list starts. Make sure all your methods work and that you know how to iterate through the list nodes in a circular manner.

Exercise, part 2: Joseph's problem with linked lists

You are now going to solve the so-called "Joseph's problem" using a circular linked list. Although there exists a analytical solution for the problem, we are going to provide a numerical solution for an adapted version of the original problem.

The problem that we consider is as follows: There are n people standing in a circle, of whom one is designated as the "first" one. This first person is executed (e.g. shot). After that, k next persons (going clockwise) are skipped, and again the following person is executed. This continues until k people are

left. For example, for $n = 5$ and $k = 1$ first index 1 is shot, 2 skipped, 3 shot, 4 skipped, 5 shot (at which point 2 and 4 are left), 2 skipped and 4 shot. At this point there is $k = 1$ person left, so the algorithm can terminate (result = [2] survives).

Create a function `safeIndexJoseph(n, k)` which returns the k safe indices to stand in when n people are standing in a circle and the executing proceeds with skipping schedule k. You have to do this using the circular linked lists you implemented in part 1. Create a main script that prints out the safe indices for $n \in \{5, \ldots, 20\}$, $k \in \{1, \ldots, 5\}$.