
Bonus-Exercise

This is a bonus-exercise. Points earned in this exercise will benefit you for your final grade of the "Basispruefung". Maximal achievable bonus is 0.25 on top of your grade and it will be added **after** the correction of the exam, but before averaging with the "Informatik I" grade. (Meaning it is a real bonus).

Academic integrity

Only submit a solution you did yourself and you understand. To ensure this we reserve the right to invite people for an oral debriefing of the exercise. We select people both randomly and those that got suspiciously similar code to other people. (This will be checked by automated tools).

Asking questions

Before sending a mail with a question to this exercise check out the homepage Inf II Homepage. We might update the exercise in case it contains bugs and collect all frequently asked questions with their answers on the website. In case your question was not answered by the information given on the homepage please send your question to the email

`Informatik2_baug@lists.inf.ethz.ch`

Using the judge

We will use the same system, as we already used for the first five exercises. Those that successfully submitted a homework so far can ignore this section. If you are one of the few people that did not use the judge system so far follow the instructions below to create an account within the system:

- Open https://judge.inf.ethz.ch/team/?cid=60 in a Web Browser and login with your student account.
- On the next site you will be asked for an “Enrollment Key“ - enter “informatik“ as the key.
- A detailed instruction on how to use the judge can be found here: Assignment 1

1 Write a function that ... (16 pts)

In the first and hopefully easiest part of the bonus exercise you are asked to complete functions according to a specification.

1.1 ... raises the mean of two numbers to its third power

```
//pre: two doubles a and b
//post: takes the mean of (a,b) and then returns the 3th power of this mean
public static double meanthirdpower(double a, double b) { ...}
```

1.2 ... checks if an integer is a factor of another integer

```
//pre: two positive non zero integers fact and n
//post: true if fact is a factor (german: Teiler) of n, false if not
public static boolean isfactor (int n, int fact){ ... }
```

1.3 ... conditionally returns either a sum or product

```
//pre: two intergers a and b
//post: if a is a even number it returns the sum of (a,b)
//      otherwise it returns the product of the two
public static int addormultiply(int a, int b){ ... }
```

1.4 ... returns the largest number contained in an integer array

```
//pre: an integer array
//post:
//      0 if the passed reference is null
//      0 if it is empty
//      otherwise the largest number in this array
public int maxinArray (int [] array) { ... }
```

Files and Submission

Link for the skeleton1.1 and link for the submission1.1

Link for the skeleton1.2 and link for the submission1.2

Link for the skeleton1.3 and link for the submission1.3

Link for the skeleton1.4 and link for the submission1.4

2 Recursive Factorial Function (14 pts)

In this exercise you are asked to repair the given recursive function. The function is supposed to compute the factorial value of any number in the range 0 to 20 and print the calculation in the order it is actually computed. Remember the definition of the factorial function:

$$n! = \begin{cases} 1 & n = 0 \\ n \cdot (n-1)! & \forall\, n > 1 \end{cases}$$

The factorial is currently computed using the following (incorrect) fac function in the main class. Your task is to correct the function.

```
// pre: assume n is greater or equal 0, but smaller than 20.
// post: return n! where n!=n*(n-1)! and 0!=1.
public static long fac(int n){
    System.out.println(n);
    long t = n*fac(n-1);
    if (n < 0)
        return 1;
    return t;
}
```

There seem to be at least two problems:

- The function crashes with a stackoverflow - which in case of a recursion means that it most likely contains an infinite loop and the only reason it stops is because it runs out of memory.
- You want the to print the results bottom up -> e.g. for input 4 the output should be 1, 1, 2, 6 and 24 (in that order).

To test your implementation here are some inputs and the expected output

input	output
0	1
1	1
	1
2	1
	1
	2
3	1
	1
	2
	6

Files and Submission

Download the skeleton code for this exercise from http://lec.inf.ethz.ch/baug/informatik2/2015/ex/ex07/02/Main.java Submit your solution to the judge at https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB15721

3 Objects: Sliding Window (16 pts)

Goal of this part of the exercise is to write a sliding window object that can return the maximum and minimum of $n > 0$ provided integer values. We do not provide the interface of the object. Rather we describe its behavior phenomenologically and with example code in the following. Your task is to implement the corresponding class such that the object's behavior is as expected.

Task Description

Provide a class `SlidingWindow` with the following properties:

- A **sliding window** `w` can be instantiated with a **window size** n in the following way:

```
// assume n of type int, n > 0
SlidingWindow w = new SlidingWindow(n);
```

- A sliding window provides an **input method** `Put` that accepts integer numbers and has no return value

```
int value;
w.Put(value);
```

- Assume that a number of $m > 0$ input values have already been provided via `w.Put`. A sliding window provides a method `Max` that returns the maximum of the most recent $\min(m, n)$ inputs. That means that given 5 inputs and a sliding window of size 4 the oldest element will be discarded, and with a sliding window of size 8 all 5 elements will be taken into account. Example:

```
SlidingWindow w = new SlidingWindow(4); // windows size 4
w.Put(1); w.Put(5); w.Put(2):
System.out.println(w.Max()); // values: 1 5 2 => output 5
w.Put(3); w.Put(4); w.Put(1);
System.out.println(w.Max()); // values: 2 3 4 1 => output 4
```

- Assume that a number of $m > 0$ input values have already been provided via `w.Put`. A sliding window provides a method `Min` that returns the minimum of the most recent $\min(m, n)$ inputs.

To test your implementation, enter the window size followed by a list of numbers followed by `end`. For example, input

```
3 1 2 3 4 end
```

should lead for each new element to an output of the form [min max]:

```
[1 1]
[1 2]
[1 3]
[2 4]
```

Files and Submission

Download the skeleton code for this exercise from http://lec.inf.ethz.ch/baug/informatik2/2015/ex/ex07/03/Main.java Submit your solution to the judge at https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB15731

4 Palindrome (12 pts)

A palindrome is a sequence of characters that from left to right provides the same text as from right to left.

(a) Given a string, determine if it is a palindrome - we only consider strings containing lower case letters only. For example, "racecar" is a palindrome. Please implement the method 'isPalindrome' inside the class Solution.

```
//pre: a String consisting of only lower case letters
//post: true if the string is a palindrom or if its empty
//       false otherwise
public boolean isPalindrome(String s)
    int left=0, right=s.length()-1;
    // implement it
    return true;
}
```

Note: For the purpose of this problem, we define an empty string as valid palindrome.

(b) Given a string S, find the longest palindromic substring in S. (you can assume that there is a unique biggest substring) For example, the longest palindromic substring of "forgeeksskeegfor" is "geeksskeeg"

We have already implemented a method $findPalindrome$. Given a string s, position left and right, it will search for a palindrom within the string. It takes a string and starts to expand it outwards from positions $left$ and $right$ as long as the characters on those positions are the same. This process is sketched below

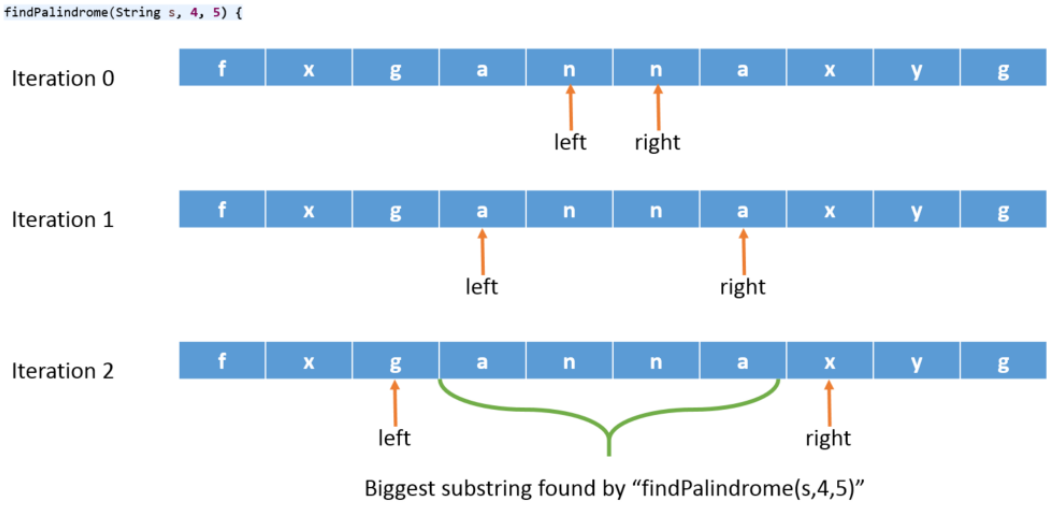


Figure 1: sketch of the simple collision detection

Your task is to implement the function $longestPalindrome$ using the provided $findPalindrome$ function. Note the a palindrome might have an uneven number of letters - e.g. "anxna" - to find such a palindrome you would call $findPalindrome$ with $left = right$.

Files and Submission

Link for the skeleton and link for the submission of part a and the submission of part b

5 Lottery Simulation (15 pts)

In this exercise, we develop a lottery simulation. In each draw of the lottery, 6 numbers between 1 and 49 are selected. Within a draw, all numbers are unique (no duplicates). Players can participate in the lottery by guessing 6 numbers between 1 and 49. A player wins the lottery if all his 6 guesses are correct.

(a) The function 'generateOneDraw()' creates 6 unique random numbers between 1 and 49. In this exercise, we want to simulate the lottery over a period of 5 weeks (1 draw per week). Implement the function 'generateAllDraws()', which creates 5 draws and stores them in the static variable 'draws'.
To test your implementation the following input

```
1 end
```

should lead to this output:

```
1 33 5 36 10 13
37 9 47 5 16 4
3 35 19 30 4 20
7 44 36 20 1 46
32 39 25 49 18 48
```

(b) We want to be notified if we have a winner, i.e. a player which has 6 correct guesses for one of the 5 draws. Implement the function 'winnerExists()', which returns true, if there is a player with 6 correct numbers. A correct guess is defined as a person having six correct numbers for one of the draws - the order of the numbers does not matter. Remember that numbers are unique within a draw.
Guesses are stored as a two dimensional array (see the bottom of the skeleton for details). You can assume only valid guesses are read in (unique numbers in the correct range).
To test your implementation the following input

```
2
3
2 34 27 42 39 10
34 7 19 22 10 23
48 32 25 49 18 39
end
```

should lead to this output:

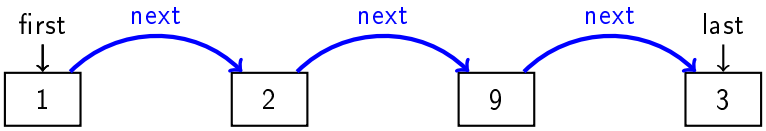
```
There is at least one winner
```

Files and Submission

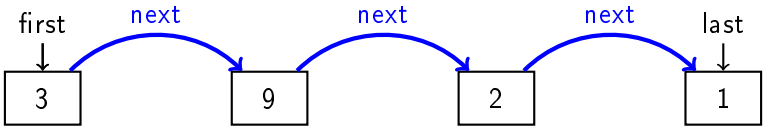
Link for the skeleton and link for the submission of part a and the submission of part b

6 Reverse List (15 pts)

In this exercise your task is to reverse a linked list. For example if the initial list looks like this



then the reversed list (after calling reverse()) should look like this.



Task

Implement the method `reverse` of the class `List`.

Hint: you may want add a helper method in order to insert an element at the first position of the list. Additionally you may want to add a helper method to instantly remove all elements from a list.

To test your implementation, enter a list of numbers followed by `end`. For example, input

```
1 2 3 end
```

should lead to this output:

```
1 -> 2 -> 3 ->
3 -> 2 -> 1 ->
```

Files and Submission

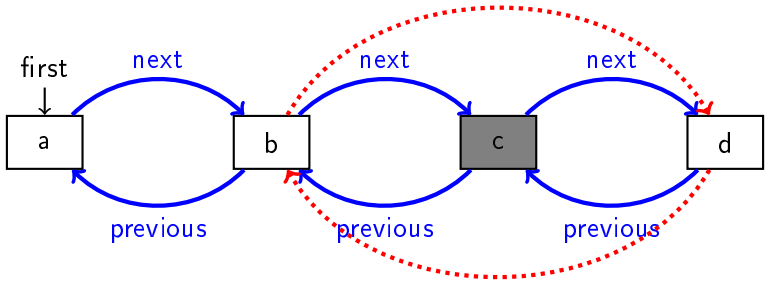
Download the skeleton code for this exercise from http://lec.inf.ethz.ch/baug/informatik2/2015/ex/ex07/06/Main.java Submit your solution to the judge at https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB15761

7 Remove an Element from a Double Linked List. (12 pts)

In this exercise your task is to remove an element from a double linked list. A double linked list is a list where each element has a reference to the next element and the previous element of the list.

```
class Node{
    private int data;          // holds the value of the node
    private Node next;         // references to next
    private Node previous;     // and previous node
    ...
}
```

Double linked lists are known for particular efficient removal and insertion of elements. In the figure below, the marked element is removed by resetting the next pointer of its previous element and the previous pointer of its next element accordingly.



Task

Implement the method `remove(Node n)` of the class `List`.

Hint: Before implementing think of invariants and special cases.

To test your implementation, enter a list of numbers containing a number 0 followed by `end`. For example, input

```
1 2 3 0 1 2 3 end
```

should lead to this output:

```
3 <-> 2 <-> 1 <-> 0 <-> 3 <-> 2 <-> 1
3 <-> 2 <-> 1 <-> 3 <-> 2 <-> 1
```

Note that in this implementation insertion happens at the beginning of the list, hence the reverse order.

Files and Submission

Download the skeleton code for this exercise from http://lec.inf.ethz.ch/baug/informatik2/2015/ex/ex07/07/Main.java Submit your solution to the judge at https://judge.inf.ethz.ch/team/websubmit.php?cid=60&problem=IB15771