Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Übungen zur Vorlesung Informatik II (D-BAUG) FS 2017

D. Sidler, F. Friedrich

http://lec.inf.ethz.ch/baug/informatik2/2017

Problem set # 4 **13.3.2017 – 21.3.2017**

Problem 4.1. Recursive Function Analysis

For the following two recursive functions:

(a)
```
boolean f (const int n) {
    if (n == 0) {
       return false;
    }
    return !f(n-1);
}
```

(b)
```
void g (int n) {
    if (n == 0) {
       System.out.printf("*");
       return;
    }
    g(n-1);
    g(n-1);
}
```

i. Formulate pre- and postconditions.

ii. Show that the function terminates (under preconditions of i.).

iii. Determine the number of functions calls as (mathematical) function of parameter n.

Submission link: https://codeboard.ethz.ch/inf2baugex04t01

Problem 4.2. Money

In how many ways can you own CHF 10? Despite its somewhat philosophical appearance, the question is a mathematical one. Given some amount of money, in how many ways can you partition it using the available denominations (bank notes and coins)? Today's denominations in CHF are 1000, 200, 100, 50, 20, 10 (banknotes), 5, 2, 1(coins). Note we ignore values below CHF 1. The amount of CHF 4, for example, can be owned in four ways: $(2, 2), (2, 1, 1), (1, 1, 1, 1)$.

Solve the problem for a given input amount, by writing the following function.

```
// PRE: billsAndCoins is array of bill and coins values, start is an index into
    this array, the array is sorted such that billsAndCoins[i] >
    billsAndCoins[i+1] > .. > 0
// POST: return value is the number of ways to partition amount
//      using billsAndCoins beginning at index start
public static int partition(int amount,
                            int[] billsAndCoins,
                            int start);
```

To allow you to focus on implementing the recursive function `partition`, we provide you with the template code that implements all functionality except the said function. The input is the amount in CHF as `int` and the expected output is the number of ways that amount can be owned as `int`.

ETH Codeboard link: https://codeboard.ethz.ch/inf2baugex04t02

Once you have implemented the partition function, you can test your program by un-commenting the annotation `@RunTests`. If you pass the test you can submit your program.

Problem 4.3. Throwing Eggs.

Suppose you are in a skyscraper with 100 floors and you want to find the lowest floor f such that an egg breaks when you throw it out the window. So you get some eggs to conduct your experiments. If you throw an egg and it doesn't break, you may re-use it. However, if it breaks, it can't be thrown again. You make the reasonable assumption that if you throw an egg from floor i and it breaks, an egg thrown from a higher floor $j > i$ will also break. So if an egg doesn't break when thrown from floor i, it also doesn't break when thrown from a lower floor $k < i$. Your task is to find a strategy to

find the lowest floor from which a thrown egg breaks such that you need as few attempts as possible (you don't care about the number of eggs you break). However, when you run out of eggs, you should be able to name the requested floor.

(a) What would be your strategy if you would have an arbitrary number of eggs?

(b) What would you do if you only had one egg?

(c) What would be your strategy if you only had two eggs?

Submission link: `https://codeboard.ethz.ch/inf2baugex04t03`