Object-Oriented Programming 2

Rolf Haenni & Annett Laube

Exercises Recursion

1. Recursion in Java

Implement the recursive solutions for the `printInteger(n,b)` and `hanoi(n)` problems in Java. Use the Eclipse debugger to inspect the execution of corresponding method calls.

2. String Operations

Write recursive methods for the following string-related problems:

- `boolean isPrefix(String s1, String s2)`: checks if s_2 is a prefix of s_1
- `boolean isSubString(String s1, String s2)`: checks if s_2 is a sub-string of s_1
- `boolean count(String s1, String s2)`: compute the number of times that s_2 appears in s_1 (without overlaps).

3. Subset-Sum Problem

Given an array of integers $a[0], \ldots, a[n-1]$, check if there is subset of the given integers that sums up to a target value t. Write a recursive method

```
boolean hasSubset(int[] a, int t)
```

which returns `true` if this is the case and `false` otherwise. Examples:

- `hasSubset([2, 4, 7], 0)` ⇒ `true`
  `hasSubset([2, 4, 7], 2)` ⇒ `true`
  `hasSubset([2, 4, 7], 4)` ⇒ `true`
  `hasSubset([2, 4, 7], 7)` ⇒ `true`

`hasSubset([2, 4, 7], 6)` ⇒ `true`
`hasSubset([2, 4, 7], 9)` ⇒ `true`
`hasSubset([2, 4, 7], 11)` ⇒ `true`
`hasSubset([2, 4, 7], 13)` ⇒ `true`

- `hasSubset([2, 4, 7], 1)` ⇒ `false`
  `hasSubset([2, 4, 7], 3)` ⇒ `false`
  `hasSubset([2, 4, 7], 5)` ⇒ `false`
  etc.

Hint: write an auxiliary recursive method `recHasSubset(int pos, int[] a, int t)`, which is called by `hasSubset` for $pos = 0$. The value *pos* indicates the position in the array where the search for a subset starts.