CURTIN UNIVERSITY (CRICOS number: 00301J)

Department of Computing, Faculty of Engineering and Science

Data Structures and Algorithms (COMP1002)

PRACTICAL 3 - RECURSION

AIMS

- To practice recursion
- To implement Towers of Hanoi

BEFORE THE PRACTICAL:

- Read this practical sheet fully before starting.

ACTIVITY 1: TOWER IMPLEMENTATION

Implement the Towers of Hanoi algorithm as a java method. Now write a main method to test this. You will also need to implement the `moveDisk(src, dest)` method. This can be as simple as an output statement printing "Moving top disk from peg source to peg destination" where source and destination are 1, 2 or 3.

ACTIVITY 2: TOWER IMPROVEMENT

Modify your Towers of Hanoi algorithm so that it additionally displays the import parameters, the (possible) value of temp, and the line or statement number of the recursive call.

To make this readable, you must indent each level of recursion. You MUST use spaces, not tabs. Do NOT use a loop to add spaces, use recursion.

Hint: You will need to add another import parameter.

Use the format given in Appendix 1 of the worksheet as a guide. Your output does not have to be identical, but it should (a) be accurate and (b) give a clear indication of what the program is doing.

SUBMISSION DELIVERABLE:

Your classes (all that are required for this program) are due at 12:00 noon on the Tuesday after your tutorial. Also include any other relevant classes that you may have created. Please only submit the *.java files, although if you have a Makefile, including that would be very welcome.

You do not need to submit your test harness if everything is fine. If there are doubts about some part of your program though, the test harness could be quite useful but it won't be available if you haven't submitted it. I will not allow you to bring code from your account during the testing since this adds uncertainty – we'll work with what you've submitted.

SUBMIT ELECTRONICALLY VIA BLACKBOARD, under the *Assessments* section.

If you finish early, use the rest of the practical to start the next worksheet, because that will be due later on.

MARKING GUIDE

Your submission will be marked as follows:

- [5] Your recursion is implemented properly.
- [2] The base case(s) have been properly defined.
- [2] Your output is clear (or looks to be clear, given the code).
- [1] You are adding spaces correctly through the use of recursion.

APPENDIX 1: SAMPLE OUTPUT FOR N=5

```
towers(5, 1, 3)
       n=5, src=1, dest=3, temp=2
(3)    towers(4, 1, 2)
           n=4, src=1, dest=2, temp=3
(3)        towers(3, 1, 3)
               n=3, src=1, dest=3, temp=2
(3)            towers(2, 1, 2)
                   n=2, src=1, dest=2, temp=3
(3)                towers(1, 1, 3)
                       n=1, src=1, dest=3
(1)                    Move top disk from peg 1 to peg 3
(4)                Move top disk from peg 1 to peg 2
(5)                towers(1, 3, 2)
                       n=1, src=3, dest=2
(1)                    Move top disk from peg 3 to peg 2
(4)            Move top disk from peg 1 to peg 3
(5)            towers(2, 2, 3)
                   n=2, src=2, dest=3, temp=1
(3)                towers(1, 2, 1)
                       n=1, src=2, dest=1
(1)                    Move top disk from peg 2 to peg 1
(4)                Move top disk from peg 2 to peg 3
(5)                towers(1, 1, 3)
                       n=1, src=1, dest=3
(1)                    Move top disk from peg 1 to peg 3
(4)        Move top disk from peg 1 to peg 2
(5)        towers(3, 3, 2)
               n=3, src=3, dest=2, temp=1
(3)            towers(2, 3, 1)
                   n=2, src=3, dest=1, temp=2
```

```
(3)                towers(1, 3, 2)
                       n=1, src=3, dest=2
(1)                    Move top disk from peg 3 to peg 2
(4)                Move top disk from peg 3 to peg 1
(5)                towers(1, 2, 1)
                       n=1, src=2, dest=1
(1)                    Move top disk from peg 2 to peg 1
(4)            Move top disk from peg 3 to peg 2
(5)            towers(2, 1, 2)
                   n=2, src=1, dest=2, temp=3
(3)                towers(1, 1, 3)
                       n=1, src=1, dest=3
(1)                    Move top disk from peg 1 to peg 3
(4)                Move top disk from peg 1 to peg 2
(5)                towers(1, 3, 2)
                       n=1, src=3, dest=2
(1)                    Move top disk from peg 3 to peg 2
(4)    Move top disk from peg 1 to peg 3
(5)    towers(4, 2, 3)
           n=4, src=2, dest=3, temp=1
(3)        towers(3, 2, 1)
               n=3, src=2, dest=1, temp=3
(3)            towers(2, 2, 3)
                   n=2, src=2, dest=3, temp=1
(3)                towers(1, 2, 1)
                       n=1, src=2, dest=1
(1)                    Move top disk from peg 2 to peg 1
(4)                Move top disk from peg 2 to peg 3
(5)                towers(1, 1, 3)
                       n=1, src=1, dest=3
(1)                    Move top disk from peg 1 to peg 3
(4)            Move top disk from peg 2 to peg 1
(5)            towers(2, 3, 1)
                   n=2, src=3, dest=1, temp=2
(3)                towers(1, 3, 2)
                       n=1, src=3, dest=2
(1)                    Move top disk from peg 3 to peg 2
(4)                Move top disk from peg 3 to peg 1
(5)                towers(1, 2, 1)
                       n=1, src=2, dest=1
(1)                    Move top disk from peg 2 to peg 1
(4)        Move top disk from peg 2 to peg 3
(5)        towers(3, 1, 3)
               n=3, src=1, dest=3, temp=2
(3)            towers(2, 1, 2)
                   n=2, src=1, dest=2, temp=3
(3)                towers(1, 1, 3)
                       n=1, src=1, dest=3
(1)                    Move top disk from peg 1 to peg 3
(4)                Move top disk from peg 1 to peg 2
(5)                towers(1, 3, 2)
                       n=1, src=3, dest=2
(1)                    Move top disk from peg 3 to peg 2
(4)            Move top disk from peg 1 to peg 3
(5)            towers(2, 2, 3)
                   n=2, src=2, dest=3, temp=1
```

```
(3)                towers(1, 2, 1)
                       n=1, src=2, dest=1
(1)                    Move top disk from peg 2 to peg 1
(4)                Move top disk from peg 2 to peg 3
(5)                towers(1, 1, 3)
                       n=1, src=1, dest=3
(1)                    Move top disk from peg 1 to peg 3
```