Imperial College of Science, Technology and Medicine	University of London
Computer Science (CS) / Software Engineering (SE)	BEng and MEng Examinations Part I
Department of Computing	Integrated Laboratory Course
Laboratory work is a continuously assessed part of the examinations and is a required part of the degree assessment. Laboratory work must be handed in for marking by the due date. Late submissions may not be marked.	

Exercise: 7	Working: Individual
Title: Loops in Java	
Issue date: 24th Nov 2003	Due date: 1st Dec 2003
System: Linux	Language: Kenya/Java

Aim

- To develop simple programs requiring **for loops** in **Kenya/Java**.

Introduction

The laboratory exercise is designed to increase your skill in writing programs with loops and to give practice in program testing.

The Problem

This exercise requires you to write three programs:

- **SWTriangle.k/SWTriangle.java** which displays a right angled triangle on the screen with the right angle at the "South West" corner.
- **NETriangle.k/NETriangle.java** which displays a right angled triangle on the screen with the right angle at the "North East" corner.
- **Table.k/Table.java** which prompts the user for two integer ranges that are then used to produce a multiplication table. The first and second integers give the range of the multiplier whilst the third and fourth integers give the range of the multiplicand. For example:

  Give lower and upper bounds of multiplier: 8 13
  Give lower and upper bounds of multiplicand: 99 103

  The program should then produce a meaningful header such as:

```
Multiplication table for [8..13] x [99..103]
```

and then print a newline, followed by the formatted table:

```
   |   99  100  101  102  103
---------------------------
 8 |  792  800  808  816  824
 9 |  891  900  909  918  927
10 |  990 1000 1010 1020 1030
11 | 1089 1100 1111 1122 1133
12 | 1188 1200 1212 1224 1236
13 | 1287 1300 1313 1326 1339
```

Each number in the table should be right justified in a number field. The size of the number field should such that it is just large enough to display the largest integer needed in the table. The number fields are separated from each other by a single space. The table headers are given above and to the left hand side of the table with the multiplier to the left and the multiplicand above. Each number in the multiplier header column should be (just) wide enough to display the largest multiplier. The table headers should be separated from the table body using '-' for the multiplicands and '|' for the multipliers.

Submit by Monday 1st Dec 2003

What to do

- **SWTriangle.k/SWTriangle.java** displays a right-angled triangle, made out of asterisks, of specified size, with the right angle in the bottom left-hand corner, e.g. for a size of 5:

```
*
* *
* * *
* * * *
* * * * *
```

Once again the height and width (of the base) should be the same, counting in asterisks.

- **NETriangle.k/NETriangle.java** displays a right-angled triangle of specified size with the right angle in the top right-hand corner, e.g. for a size of 5:

```
* * * * *
  * * * *
    * * *
      * *
        *
```

- **Table.k/Table.java** Design and write your program so that it prints the multiplication table as described above. When you design your program you may wish to include auxiliary methods to simplify your code.

  You will find the program easier to write if you divide the problem into smaller sub-problems, with a specialised method written to solve each sub-problem.

  Your program will need to perform tasks of the following general kind, depending on exactly how you wrote it.

  - Calculate the number of characters needed to write out an integer. Do not use logarithms to do this. Use a loop or recursive method instead.
  - Print out an integer in a field of a specified size, so that the integer is right justified and the field is padded with spaces as necessary.
  - Print out a specified number of spaces.

Unassessed

Write a program:
HollowSquares.k/HollowSquares.java prompts the user for a *start* size and a *finish* size, then prints out successively larger hollow squares (from the start size to the finish size inclusive) with their right-hand sides aligned, eg with 3 (the smallest possible size) and 5:

```
  * * *
  *   *
  * * *

 * * * *
 *     *
 *     *
 * * * *

* * * * *
*       *
*       *
*       *
* * * * *
```

Submission

- Before submitting, you should **test your programs on a wide range of user inputs**.

  Submit your programs:
  SWTriangle.k/SWTriangle.java, **NETriangle.k/NETriangle.java** and **Table.k** and **Table.java**. by typing the command:
  submit 7 at your Unix prompt.

Assessment

SWTriangle	1
NETriangle	1
Table	3
Design, style, readability	5
Total	10