Assignment 3

Due. Oct. 26 (Wednesday), 17:30.

Programming style (10 marks).

1. (10 marks) Chapter 4, Programming 8, p. 168.
   Write a C++ program that checks whether the bracketing operators (parentheses, brackets, and curly braces) in a string are properly matched. As an example of proper matching, consider the string

   ```
   { s = 2 * (a[2] + 3 ); x = (1 + (2)); }
   ```

   If you go through the string carefully, you discover that all the bracketing operators are correctly nested, whith each open parenthesis matched by a close parenthesis, each open bracket matched by a close bracket, and so on. On the other hand, the following strings are all unbalanced for reasons indicated:

   ```
   (([])     The line is missing a close parenthesis.
   )(        The close parenthesis comes before the open parenthesis.
   {(})      The bracketing operators are improperly nested.
   ```

   The reason that this exercise fits in this chapter is that one of the simplest strategies for implementing this program is to store the unmatched operators on a stack.

2. (10 marks) Chapter 4, Programming 13, p. 169
   Write a program to simulate the following experiment, which was included in the 1957 Disney film, *Our Friend the Atom*, to illustrate the chain reactions involved in nuclear fission. The setting for the experiment is a large cubical box, the bottom of which is completely covered with an array of 625 mousetraps, arranged to form a square grid 25 mousetraps on a side. Each of the mousetraps, is initially loaded with two ping-pong balls. At the beginning of the simulation, an additional ping-pong ball is released from the top of the box and falls on one of the mousetraps. That mousetrap springs and shoots its two ping-pong balls into the air. The ping-pong balls bounce around the sides of the box and eventually land on the floor, where they are likely to set off more mousetraps.

   In writing this simulation, you should make the following simplifying assumptions:

   - Evey ping-pong ball that falls always lands on a mousetrap, chosen randomly by selecting a random row and column in the grid. If the trap is loaded, its balls are released into the air. If the trap has already sprung, having a ball fall on it has no effect.
   - Once a ball falls on a mousetrap–whether or not the trap is sprung–that ball stops and takes no further role in the simulation.
   - Balls launched from a mousetrap bounce around the box and land again after a random number of simulation cycles have gone by. That random interval is chosen independently for each ball and is always between one and four cycles.

   Your simulation should run until there are no balls in the air. At that point, your program should report how many time units have elapsed since the beginning, what percentage of the traps have been sprung, and the maximum number of the balls in the air at any time in the simulation.

3. (8 marks) Chapter 5, Programming 10, p. 200.
   The `strutils.h` library contains a function `IntegerToString`. You might have wondered how the computer actually goes about the process of converting an integer into its string representation. As it turns out, the easiest way to implement this function is to use the recursive decomposition of an integer. Write a recursive implementation of `IntegerToString` so that it operates recursively without using any of the iterative constructs such as `while` and `for`.