Objective: To use recursion in problem solving. This assignment is due at mid-night on February 28. The first and second questions have to be completed. Question three is a bonus question. You do not have to complete question three, but you will receive extra bonus marks if you do.

1 Base 8 Representation

Write a program to convert a non-negative integer decimal number with based 10 to the representation of base 8. Base 8 representation is composed of only 8 possible numbers 0 – 7. For example, the base 8 representation of number 10 is 12 (since $1\times8 + 2 = 10$). To obtain the base 8 representation of a decimal number, we use the following recursive procedure:

\[
\begin{align*}
\text{if the number is less than 8} & \quad \text{output the number} \\
\text{else} & \quad \text{find the base 8 representation for the number divided by 8} \\
& \quad \text{output the mod of the number with 8}
\end{align*}
\]

Write a recursive program to implement the base 8 conversion based on the above procedure.

2 Sierpinski Triangle

Write a recursive program to draw the Sierpinski triangle. Fig. 1 shows Sierpinski triangles of different orders. As shown in Fig. 1 (a), the first order Sierpinski triangle is just a normal triangle. Sierpinski triangle can be constructed recursively: the order $n$ triangle is composed of three order $(n-1)$ triangles whose outer vertices are from the three outer vertices and three middle edge points of the order $n$ triangle. Your recursive function should include parameters to represent three triangle vertices $x_1, y_1, x_2, y_2, x_3, y_3$ and the order $n$.

![Figure 1: Sierpinski Triangles.](image)

(a) Order 1.  (b) Order 2.  (c) Order 3.  (d) Order 8.

Use StdDraw in your implementation.
3 Recursive Random Tree (Bonus Question)

Using Recursion, we can construct trees as shown in Fig. 2. Random factors are included so that each time you run the program it will generate a different tree.

There are different schemes to grow a tree. The following is just one of these methods. The tree starts from a point, for instance, the middle point at the bottom of the canvas. It grows a trunk along a direction with a certain length. Then, the tree splits into two directions at the end of the trunk; along each direction we can grow another tree using the same scheme. This is the recursive structure of a tree.

To make the tree look like a real tree, we need to make the sub-branches shorter and thinner. This can be achieved by multiplying a factor in $(0, 1)$ to the branch length and thickness from the previous level. The factor can be chosen something like $0.95 + \delta$, where $\delta$ is a small random number. You can also introduce randomness to control whether a splitting will occur.

The two branching directions are determined by 2 random angles $\theta_1$ and $\theta_2$, so that, if the trunk growing direction is $\alpha$, the two branches will follow the directions of $\alpha + \theta_1$ and $\alpha + \theta_2$. Use small $\theta_1$ and $\theta_2$ if you want the tree to grow mostly upwards. The last step is to draw a red “leave” (in fact a filled circle) at each of the branch terminal.

![Recursive Tree](image)

Figure 2: Recursive Tree.

4 What to Submit

Submit the Java programs. Pay attention to good Java programming style. Include comments at the beginning of each program to show your name and give some explanations about what your program does. Upload your Java files to webCT before the submission deadline. There will be 4 days grace period. But late submission may involve 10% point deduction for each day. Submissions later than 4 days are not accepted.