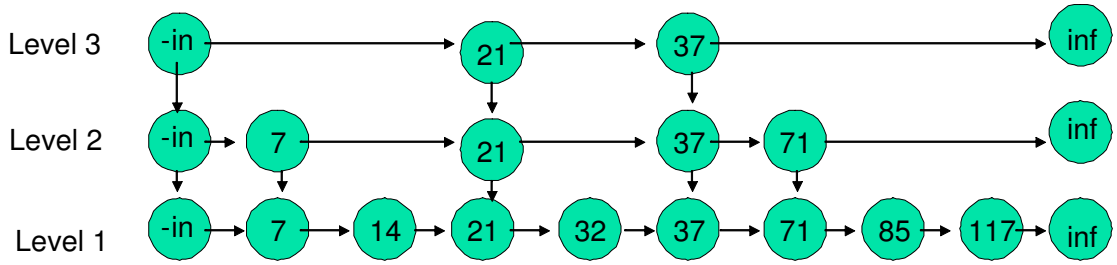
Cs352 — Homework #7
Visualizing SkipList (Version 3)

November 18, 2003

Due Time: 11/28 (9:00PM). Submission in pairs is allowed and encouraged.

Turnin ID: cs352_assg7

Turnin Specification: You should turnin ONE Makefile, TWO .c (or .cpp) files (one is for SL module, the other is for graphics module) and at most ONE .h file. The .c (or .cpp) and .h files can have whatever filenames you like, but by running the command "make", an executable named "GSkipList" should be produced. (PS: Please don't turnin directories. Turnin files instead.)



Create an OpenGL program that works with the SkipList data structure you created in hw4, and shows in an OpenGL screen the SkipList. After each operation on the SkipList (such as insertion or deletion of an element) you should update the SkipList displaying window to reflect the new skiplist. Further instructions:

1. You can assume that all values stored in the SL are positive integers between 0 and 999 (inclusive), and that no more than 20 different keys are stored in the skip list at the same time at any point of time when running the program. You can also assume that no grading testcase will generate a skip list of more than 10 levels at any point of time. (All the above is just to guarantee that you always have enough space in the window to display all your nodes.)

2. Whenever encountering an error case (e.g., inserting the same key twice, deleting a non-existing key), output the error message (the same as we specified in hw4) to the text console (the same way we did in hw4) and the graphics window doesn't change.

3. Draw the SL on the window using as much space as possible. Once an element is inserted or deleted, the other elements should not change their location on the screen, unless necessary (see below). The horizontal distance between elements with consecutive keys should follow the following rule. The distance between any pair of consecutive elements cannot be more than twice the distance between any other pair of consecutive elements. Once this condition is violated, you should re-render the whole SL, so that the horizontal distance between the consecutive elements is the same (i.e. equally spaced).

4. Each element can be represented either as a circle or as a square (your choice). Inside the element the numeric value of the key of this element must appear. Arrows should connect each element to the elements it points to in the SL.

5. Use two modules (each implemented in a single .c file respectively. That's why you are to turnin TWO .c files) in your project — one contains the graphics part and the other contains the functions that maintain the SL. Make sure to submit the Makefile as well.

6. The syntax of the commands of manipulating the SL and how the input is taken by the program is the same as specified in hw4.

7. Cells representing the elements of the SL should not intersect with each other. If they do, the SL should be re-rendered the same ways as described above with equal gaps between the cells. If even then they still intersect, an error message "Intersect\n" should be displayed on the text terminal (but the element should still be inserted without re-rendering the SL, so there is intersection in the SL in this case).