using given conditions, in order to solve problems. Teachers should introduce the abstract concepts of DB including field and record through data. Step 3: Students should be able to select the fields, by themselves, they can make the DB with similar content to that of the presented problem.

Stage 2: Skill And Extension Knowledge Acquisition - ON THE COMPUTER Step 1: Learn the lowest DB software interface to make the DB table, and implement basic inquiries, sort and searching. Step 2: Expand the number of records, using the example data that was learned through manual work, and input the data into the DB software. Step 3: Learn various commands of the DB and how to use those commands including inquiries, sorts and searching with which students can operate the DB data file that they had input or were provided.

This work was supported by the Brain Korea 21 Project.

**Using Recursion as a Tool to Reinforce Functional Abstraction**

Raja Sooriamurthi, University of West Florida, USA (sraja@cs.uw)

A good design lays the firm foundation of any engineering effort. The most severe impediment towards attaining a good design is managing the complexity of the problem. While complexity can't be avoided the best we can do is learn how to cope with it. Two tools that students are taught to help slay the dragon of complexity are (1) modularity (divide and conquer) which helps to better manage it and (2) abstraction which helps to reduce it.

When teaching introductory programming apart from the syntactic and semantic rules of a programming language, students need to be introduced to good design strategies. Functional abstraction is a cornerstone strategy in good software design. In this poster we present a template-based introduction to recursion which emphasizes the notion of functional abstraction and assists students in cultivating their intuitions. The irony and the key behind this pedagogical strategy, for a topic that is normally considered hard, is to encourage students not to think too hard. In the form of a Zen style paradox, at the introductory level, we have found that the harder you think the harder it becomes to comprehend the charm of recursion!

**"Network Protocols and Services" - a Non-Specialist Approach to Teaching Networking**

David Stratton, University of Ballarat, Australia

Networking, especially as represented by the Internet, is becoming increasingly embedded in all aspects of computing. Whilst this tendency increases the need for specialised network education it also creates an expectation that all graduates of computing degrees will have a broad and basic understanding of networking concepts. This poster describes a first year core unit that addresses this requirement by presenting network concepts in a non-technical yet "hands-on" manner.

The approach of Network Protocols and Services (NPS) is firmly at the qualitative end of the spectrum of existing approaches to teaching networks. Established approaches are reviewed and classified according to their year-level, their dependence on specific hardware and the extent to which programming knowledge is assumed. NPS also goes out of its way to give students direct experience of networking concepts - one half of the class time is spent in practicals that run in a standard university PC lab.

Two distinctive aspects of NPS, the use of packet tracing and of network simulation, are described in some detail. Packet tracing is performed using a demonstration version of a commercial packet analyser and networks are simulated using a locally developed package. The overall structure of the Unit is described and the design and use of the particular tools is related to the pedagogical aims of each section of the course.

**Program Paper-Slide-Show**

Minoru TERADA (terada@sanpo.t.u-tokyo.ac.jp), University of Tokyo, Japan

The `paper-slide-show' is a traditional form of Japanese children's entertainment in which the narrator tells a story and illustrates it by showing a sequence of painted pictures.

We designed and implemented a source code animation system based on the metaphor of the paper-slide-show. Our system offers an entry-level code animation with which student can trace the behavior of his program by himself.

Our system is

1. Automatic: No modification of the source code is necessary. The only requirement is to use the debugging option when compiling.
2. Passive: Once the student has started the animation from the command line, all he has to do is to watch his code be animated automatically.