A registrar's program to manage registration and scheduling, including printing student registration status, class lists for instructors, room use sheets, and process add-drop cards.

A program to teach arithmetic to a little brother or sister (i.e., a program for computer-aided instruction). This could include a scoring system, for example, and perhaps some analysis of errors to determine what kinds of problems to concentrate on. This is not a well-defined project and the group must be careful to decide what is to be handled.

A system to perform differentiation and simplification on symbolic expressions. Warning - simplification is quite tricky. I recommend encouraging groups to use existing algorithms first and then augment their programs. One group attempted this project and did a reasonable job with a pretty printing expression display and with function definitions but got bogged down on simplification. They had a hard time believing that simplification is in the eye of the beholder and the next step of the problem solving.

A database system, perhaps to maintain SCRIBE bibliography databases. This would make it convenient to add new entries to the database, along with key words, and also allow searching of the database, either by key word or by other fields, or with a special query language. Keep the on-line database alphabetized.

A database system for maintaining voting records of Federal and/or State Senators and Representatives. The system should keep all districts and their current representative along with their party, data of election, and a set of user-determined votes. There should be a way to score votes; for example, to determine how a person rates on a set of issues about the environment or the ERA or the space program. In the congressional record, votes recorded include yes, no, did not vote, paired for/against, unknown, etc. The user should be able to specify how these correspond to 1 or 0 or whatever for computing scores. There should be various ways to print out this information (again, perhaps with a query language), deal with people winning and losing elections, resigning, dying, etc.

A prototype simulation of a small, low-cost computing engine that one might find for sale in stores to the average consumer in the foreseeable future. It could be for children or adults; it might perform a common routine task now performed manually, or it might be a game. The Texas Instruments "Speak-and-Spell" ~(a registered trademark of Texas Instruments~) game would be one example.

A modest version of a game of Adventure or Dungeons and Dragons. The computer would serve as the dungeon-master. Alternatively, a game where the computer is a competitor or the board manager for a group of people. Depending on the game, an attractive display may be an important part of the project. Parker Brothers' "Clue" is an interesting game: you have to keep track of what other players know as well as what you know. Other possibilities are mastermind, checkers, bridge, backgammon, chess, and Go. Keep in mind that writing the programs to handle these games may not be nearly as fun or easy as playing them; certain of last year's students have threatened murder if anyone mentions the Rivets game to them again. Mastermind is perhaps too simple a project,

checkers is a bit more complex but still pretty straightforward if existing approaches are used, and adventure games can be made reasonably difficult.

An on-line recipe file and meal planner. This could compute the overall nutrition value of a meal, summarize the necessary ingredients, and estimate cost. It might also be able to suggest a meal given "I want to use up this ground beef" or "I'm out of eggs and lettuce."

A teacher's program to detect similarities in programs (i.e., a "cheat detector"), for programs of about the size and complexity required for week long homework assignments.

A program version comparator and updater. This is like the previous suggestion, but the emphasis is on determining differences rather than similarities. One might use this system after a day's editing, to get a summary of all changes. Suppose that two people edit the same program, adding different fancy features. This system might determine the differences between them and automatically or (more likely) interactively merge them to produce a new program with the features of both.

An automated bank teller. This would keep track of accounts, and let users perform various transactions: shuffle money between accounts, make deposits and withdrawals, pay bills (manually or automatically), include a display similar to the ones at the real money machines, etc. The system should be resistant to abuse and should be reliable (able to keep everyone's money straight in spite of system crashes). This is a simple project unless it includes features like multiple tellers and simultaneous deposits and withdrawals, bank manager programs, and good backup systems.

A string handling package. This would provide dynamic allocation of variable-length character strings, with either explicit or automatic de-allocation. In addition to managing storage, the package should include facilities for operating on strings (copying, concatenating, comparing, sorting, searching, matching, etc.). This project is straightforward but can involve a lot of work, especially if fancy pattern matching is attempted. One group did a nice job on a moderately sized string package.

A complex string-matching package, expanded into a spelling and diction checker. (The string matching might be based on regular expressions a la Aho). It could check a document for commonly misspelled words, as well as instances of such redundant fragments as "-ing behavior", "added increment", and so on.

A personal finances manager. Something to balance the checking account, keep track of bills, match receipts to credit card statements, do taxes, provide summary reports in various categories, and so on. One group did a modest version of this.

A calendar maintainer. This would schedule your time, send reminders when necessary (through the computer mail system), perhaps coordinate with other people's calendars for scheduling meetings, and so on. Printing parts of the calendar on demand is a must ("What's up for today?" or "Let's see next week at a glance." or "What in the world was I doing last Tuesday?" or "On what day of the week for the next four weeks is there an hour free starting at 10:00?").

A verification condition generator.

A tree (or graph) structured bulletin-board system, with the root allowed in any user's directory. Any user could post notices for others to read. Notices (especially replies) could be associated with other notices (hence the graph structure). A user might want to ask such questions as "which notices of mine have had answers attached that I haven't read yet?" Some branches of such a bulletin board might serve as a structured on-line documentation system. This would have to have a nice user interface; it should also lead the beginner by the hand (possibly by guiding them through a documentation tree!) A stripped-down version of the user interface for hard-copy terminals should be a part of the core system. One group tried a similar project but was overly ambitious and had trouble completing the project.

Write a program for analyzing the NFL playoff situation, which has a complex set of rules for resolving various situations that can arise. One portion of the project would be an evaluator to determine the playoff choices given the end-of-season scenario. The rules should be modularly implemented so that they can be changed (as they frequently are). The user interface could vary from something simple to a system capable of answering questions like "If Pittsburgh loses to Houston, what must occur for Pittsburgh to enter the playoffs?" (expressed in some more formal notation). Incremental improvements might include something like the capability to guess, based on the season record so far, how some upcoming games might be decided or what games might be crucial.