Assignment 1

September 19, 1990

CMU 17-713 (Prof. Tygar)

Programming: This assignment is due by September 30. Your assignment is to write files `balance.h` and `balance.C` which can manipulate 2-3 trees. (As you recall, 2-3 trees are a basic data type which you learned about in your elementary data structures class. Just in case you sold your data structures textbook and forgot how to 2-3 trees work, refer to the attached handout which has a discussion of that data structure.)

These files should contain definitions and routines which support the following public functions. One should be able to compile a program (in a separate file) which uses these functions:

- `class prique` — a priority queue (which is actually realized as a 2-3 tree.) Be sure to include a constructor which initializes the priority queue to be empty and a destructor which does full garbage collection. **Extra Credit 1:** Include a second constructor which takes a single argument, a string of integers separated by a commas (with no whitespace) which loads the values into the new priority queue (with no duplicates).

- `>>` — executing `i >> p` enters the integer `i` into the priority queue `p` (deleting duplicates); executing `p >> i` returns the minimum element from the priority queue `p` into the integer variable `i` and deletes that element from the priority queue. (Whenever you delete a value, be sure to perform garbage collection.)

- `<<` — executing `p << i` enters the integer `i` into the priority queue `p` (deleting duplicates); executing `i << p` returns the minimum element from the priority queue `p` into the integer variable `i` and deletes that element from the priority queue. (Whenever you delete a value, be sure to perform garbage collection.)

- `*` — `*p` returns FALSE if the priority queue `p` is empty, and TRUE otherwise.

- `->` — `i->p` returns TRUE if the integer `i` is in the priority queue `p`, false otherwise.

- `~` — `~p` initializes a priority queue to be empty.

- **Extra Credit 2:** Make a template for the priority queue which can handle arbitrary data types (instead of just integers) given appropriate equality and inequality operators `==, >, >=, <, <=, !=`.

We will shortly put sample programs usings these definitions in (and a pmax executable) in the class directory, which is `/afs/andrew/scs/cs/17-713/ass1`. (Yeah, we know that is a stupid pathname. Don't ask ....) You may want to check from time to time for extra files.

Be sure to generate thorough tests! We may assume that untested features don't work.

Prepare a paper hand-in packet. It should contain your name*, a title indicating that it is assignment 1* (indicate if you attempted extra credit 1 or 2), a brief table of contents*, and a listing of all files (`balance.C`, `balance.h`, test source code files, output and input files for the tests). The pages should be numbered and

stapled. It is fine if you use pen or pencil for the items listed by an asterisk, and for the page numbering. Paper hand-in is to Annamarie Mackuliak — as usual. You may work on this assignment in pairs – if so, please indicate that clearly when you hand-in the program. (We usually expect that people working on this assignment in pairs will attempt one or both of the extra credit parts.)

An electronic hand-in is also required. You will find a directory named after your surname that you have access to in `/afs/andrew/scs/cs/17-713/handin1`. Put a copy of all files and a pmax executable compiled under Andrew there.