CSCI 301
Computer Science 2

Project EXAMPLE: TESTS of the formatting program.

The following script file tests and illustrates the text formatting program format. It shows the input file, run, and output file of three runs of the program. In the third example, the input file is the program's source code, so it is not reproduced here.

```
csh> cat in1.dat
This is a very small test input file
containing only a few words on just a few lines.
It should be an easy test for the formatting program.
csh> format
Enter input file name: in1.dat
Enter output file name: out1.dat
Enter an integer value between 30 and 80: 30
csh> cat out1.dat
This is a very small test
input file containing only a
few words on just a few lines.
It should be an easy test for
the formatting program.
csh>
csh>
csh>
csh> cat in2.dat
The subject known as graph theory is a branch of mathematics
enjoying a special alliance with computer science in both its practical and
theoretical aspects. First, the language, techniques, and theorems of graph
theory may be applied to systems as diverse as data
structures and parse trees. Second, graph theory itself is rich in
problems which challenge our ability to solve by computer.
Indeed, not many graph-theoretic problems appear to have algorithms that
solve them in polynomial time. Many of the first
problems shown to be NP-complete were problems in graph theory.

From Minimum Spanning Trees, chapter 22 of The New Turing Omnibus, by
A. K. Dewdney. New York: Computer Science Press, 1993, p.146.
csh> format
Enter input file name: in5.dat
Enter input file name: none.dat
Enter input file name: in2.dat
Enter output file name: out2.dat
Enter an integer value between 30 and 80: 15
Enter an integer value between 30 and 80: 90
Enter an integer value between 30 and 80: 42
csh> cat out2.dat
The subject known as graph theory is a
branch of mathematics enjoying a special
alliance with computer science in both its
practical and theoretical aspects. First,
the language, techniques, and theorems of
graph theory may be applied to systems as
diverse as data structures and parse
trees. Second, graph theory itself is rich
```

```
in problems which challenge our ability to
solve by computer. Indeed, not many
graph-theoretic problems appear to have
algorithms that solve them in polynomial
time. Many of the first problems shown to
be NP-complete were problems in graph
theory. From Minimum Spanning Trees,
chapter 22 of The New Turing Omnibus, by
A. K. Dewdney. New York: Computer Science
Press, 1993, p.146.
csh>
csh>
csh> format
Enter input file name: format.cxx
Enter output file name: out3.dat
Enter an integer value between 30 and 80: 60
csh> cat out3.dat
// 22C:30/115 // Computer Science III // Spring, 2001 //
format.cxx // This program reads an input file of text and
writes an output file of the // same text, formatted into
lines no longer than a maximum length. The // names of the
input and output files and the maximum line length are //
read from the terminal. Functions open the files, and
continue prompting // for file names until names are entered
than can be successfully opened. // Another function reads
the maximum line length, which must fall within // bounds
set by two program constants. // The program reads and
writes words from the input file one at a time. // It keeps
track of the length of the current line so far; if the next
// word would cause that line to exceed the maximum length,
the program // terminates that line, writes the word on the
next line, and resets the // line length. The program writes
a blank after each word, except perhaps // the last word on
a line. A word is a string of contiguous non-blank //
characters, and we assume that no input word is longer than
the input line // length set for the run. #include
<stdlib.h> #include <iostream.h> #include <iomanip.h>
#include <fstream.h> #include <string.h> const int MIN = 30;
// Minimum line length const int MAX = 80; // Maximum line
length typedef char string[MAX+1]; void open_input_file (
ifstream& in_f ); // Opens for input a file named from the
terminal. // Postcondition: A file stream has been opened
for input. void open_output_file ( ofstream& out_f ); //
Opens for output a file named from the terminal. //
Postcondition: A file stream has been opened for output. int
read_int ( int small, int large ); // Reads an input value
within specified bounds. // Precondition: small and large
are positive integers, with small <= large. //
Postcondition: The function returns a value in [small,large]
entered from // the terminal. int main() { ifstream in_file;
// The input file stream ofstream out_file; // The output
file stream int max_length; // Maximum line length string s;
// Each string read in and printed out int s_len; // The
length of the string s int line_len; // The length of the
current output line so far open_input_file(in_file); // Open
the input file. open_output_file(out_file); // Open the
output file. max_length = read_int(MIN,MAX); // Read the
maximum line length. line_len = 0; // Initially, the line
length is zero. in_file >> s; // Read from the input file.
while ( ! in_file.eof() ) // Are we done yet? { s_len =
strlen(s); // Identify the string's length. if ( line_len +
s_len <= max_length ) // If there is room on the line ... {
out_file << s; // Write to the output file. line_len =
line_len + s_len; // Increment the line length. } else //
Start a new line. { out_file << endl << s; // Write to the
output file. line_len = s_len; // Reset the line length. }
if ( line_len < max_length ) // If there is room for a blank
... { out_file << ' '; // Write to the output file.
++line_len; } in_file >> s; // Read from the input file. }
out_file << endl; // Write to the output file.
in_file.close(); // Close the input file. out_file.close();
// Close the output file. return EXIT_SUCCESS; } void
open_input_file ( ifstream &in_f ) { char
input_file_name[80]; do { cout << "Enter input file name: ";
cin >> input_file_name; in_f.open(input_file_name); } while
( in_f.fail() ); } void open_output_file ( ofstream &out_f )
{ char output_file_name[80]; do { cout << "Enter output file
name: "; cin >> output_file_name;
out_f.open(output_file_name); } while ( out_f.fail() ); }
int read_int ( int small, int large ) { int value; do { cout
<< "Enter an integer value between " << setw(1) << small <<
" and " << setw(1) << large << ": "; cin >> value; } while (
value < small || value > large ); return value; }
```