```
------------------------------------------------------------------
iostream1.cpp: C++ style formatted file I/O
------------------------------------------------------------------

#include <fstream>
#include <iomanip>
#include <iostream>
#include <...>
using namespace std;

void readwrite(istream &fin, ostream &fout) {
  string s;
  float f;

  fout.setf(ios::fixed);
  fout.precision(2);

  while (fin >> s >> f) {
    fout << setw(12) << left << s << " "
         << "0x" << setw(8) << right
         << setfill('0')
         << hex << *(int *)&f << " "
         << setfill(' ')
         << setw(10) << f << "\n";
  }
}

int main(int argc, char *argv[]) {
  if (argc != 3) {
    cerr << "usage: " << argv[0] << " input.txt output.txt\n";
    return 0;
  }

  ifstream fin;
  ofstream fout;

  fin.open(argv[1]);
  fout.open(argv[2]);

  readwrite(fin, fout);

  fin.close();
  fout.close();
}

------------------------------------------------------------------
unix> cat input.txt      |     unix> iostream input.txt
Knoxville 1.234568       |     Knoxville    0x3f9e0653       1.23
Nashville 0.014142       |     Nashville    0x3c67b3d9       0.01
```

```
------------------------------------------------------------------
cstdio1.cpp: C style formatted file I/O
------------------------------------------------------------------

#include <cstdio>
using namespace std;

void readwrite(FILE *fin, FILE *fout) {
  char s[80];
  float f;

  while (fscanf(fin, "%s %f", s, &f) == 2) {
    fprintf(fout, "%-10s 0x%08x %12.2f\n", s, *(int *)&f, f);
  }
}

int main(int argc, char *argv[]) {
  if (argc != 3) {
    fprintf(stderr, "usage: %s input.txt output.txt\n", argv[0]);
    return 0;
  }

  FILE *fin = fopen(argv[1], "r");
  FILE *fout = fopen(argv[2], "w");

  readwrite(fin, fout);

  fclose(fin);
  fclose(fout);
}

------------------------------------------------------------------
Hint:
Use cplusplus.com learn about IOSTREAM data formatting.

Use man pages to learn about CSTDIO data formatting:
unix> man 3 scanf
unix> man 3 printf

cin >> is similar to scanf      fin >> is similar to fscanf
cout << is similar to printf    fout << is similar to fprintf

scanf(...)  == fscanf(stdin, ...)
printf(...) == fprintf(stdout, ...)

------------------------------------------------------------------
unix> cstdio input.txt output.txt; cat output.txt
Knoxville    0x3f9e0653       1.23
Nashville    0x3c67b3d9       0.01
```

```
-------------------------------------------------------------
iostream2.cpp: Advanced C++ style formatted file I/O on a string
-------------------------------------------------------------

#include <sstream>
#include <...>
using namespace std;

void readwrite(istream &fin, ostream &fout) {
  ...
  string line;
  while (getline(fin, line)) {
    istringstream iss(line);
    iss >> s >> f;

    fout << same as before << "\n";
  }
}

int main(int argc, char *argv[]) {
  ifstream fin;
  ofstream fout;

  if (argc == 2 || argc == 3) fin.open(argv[1]);
  else                        fin.istream::rdbuf(cin.rdbuf());

  if (argc == 3) fout.open(argv[2]);
  else           fout.ostream::rdbuf(cout.rdbuf());

  readwrite(fin, fout);

  if (fin.is_open())  fin.close();
  if (fout.is_open()) fout.close();
}

-------------------------------------------------------------
Hint: The istringstream class allows formatted >> reading of
string data. The ostringstream class allows formatted << writing.

ADVANCED: Data can be redirected from one iostream to another
by switching the underlying stream buffers (rdbuf()).

UNUSUAL CODE: The fstream class has an rdbuf() member but we
need the ios::rdbuf() version to pass an argument, thus the
fin.istream::rdbuf() and fout.ostream::rdbuf() syntax above.

Use cplusplus.com to work out behavior you don't understand.
-------------------------------------------------------------
```

```
-------------------------------------------------------------
cstdio2.cpp: Advanced C style formatted file I/O on a string
-------------------------------------------------------------

#include <cstdio>
using namespace std;

void readwrite(FILE *fin, FILE *fout) {
  ...
  char *line = NULL;
  size_t linesize = 0;
  ssize_t N = 0;

  while ((N = getline(&line, &linesize, fin)) > 0) {
    sscanf(line, "%s %f", s, &f);
    fprintf(fout, same as before);
  }
}

int main(int argc, char *argv[]) {
  FILE *fin;
  FILE *fout;

  if (argc == 2 || argc == 3)
    fin = fopen(argv[1], "r");
  else
    fin = stdin;

  if (argc == 3)
    fout = fopen(argv[2], "w");
  else
    fout = stdout;

  readwrite(fin, fout);

  fclose(fin);
  fclose(fout);
}


-------------------------------------------------------------
Hint:
sscanf(s, ...) allows parsing of char * string s
sprintf(s, ...) allows formatted printing to s
-------------------------------------------------------------
Hint:
stdin, stdout, and stderr are preopened FILE streams
-------------------------------------------------------------
```

```
-----------------------------------------------------------------
iostream-copy.cpp: C++ style binary file I/O
-----------------------------------------------------------------

#include <fstream>
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {
  int n=1024, nread;
  char buf[n];

  while (1) {
    cin.read(buf, n);
    nread = cin.gcount();
    if (nread == 0 && cin.eof())
      break;
    cout.write(buf, nread);
  }
}

-----------------------------------------------------------------
cstdio-copy.cpp: C style binary file I/O
-----------------------------------------------------------------

#include <cstdio>
using namespace std;

int main(int argc, char *argv[]) {
  int n=1024, nread;
  char buf[n];

  while (1) {
    nread = fread(buf, 1, n, stdin);
    if (nread == 0 && feof(stdin))
      break;
    fwrite(buf, 1, nread, stdout);
  }
}
```

```
-----------------------------------------------------------------
PPM FILES: Portable Pixel Map (cf. wikipedia.org)
-----------------------------------------------------------------

Most image files consist of a header (size, metadata) followed
by binary data (jpg and other formats store compressed data).

PPM files have an ASCII header that should be read using formatted
file I/O. The pixel data is binary and should be read using binary
file I/O. Each pixel is an RGB triplet that designates how much
Red, Green, and Blue to mix. That is,

P6
ncols nrows
maxvalue

R1 G1 B1  R2 G2 B2  R3 G3 B3  ...  RN GN BN

where ncols refer to the horizontal (left-right) dimension, nrows
refer to the vertical (up-down) dimension, and N = nrows*ncols.

We will assume the RGB triplets are stored as unsigned chars in
which case maxvalue = 255, but other formats are possible. See
wikipedia.org for more details.

-----------------------------------------------------------------
ppmdump.cpp: program for converting PPM to human readable output
-----------------------------------------------------------------

#include <...>
using namespace std;

void ppmdump(istream &fin) {
  string magicid;
  int ncols, nrows;
  int maxvalue;

  fin >> magicid >> ncols >> nrows >> maxvalue;

  cout << magicid << "\n";
  cout << ncols << " " << nrows << "\n";
  cout << maxvalue << "\n";

  while (fin.get() != '\n') { /* skip past newline */ }

  int nrgb = 5;         // pixels per line
  int nrgb_read;        // pixels read per line

  int npixels_read = 0; // pixels read in total

  unsigned char *rgb_ptr;       // data pointer
  unsigned char buf[3*nrgb];    // data buffer

  char text[80];        // text buffer

  while (1) {
    fin.read((char *)buf, 3*nrgb);
    nrgb_read = fin.gcount()/3;
    if (nrgb_read == 0 && fin.eof())
      break;

    sprintf(text, "%07d ", npixels_read);
    cout << text;

    rgb_ptr = buf;
    for (int i=0; i<nrgb_read; i++) {
      cout << " ";
      for (int j=0; j<3; j++) {
        sprintf(text, " %03u", *rgb_ptr++);
        cout << text;
      }
    }
    cout << "\n";

    npixels_read += nrgb_read;
  }
}

int main(int argc, char *argv[]) {
  if (argc != 2) {
    cerr << "usage: " << argv[0] << " input.ppm\n";
    return 0;
  }

  ifstream fin;
  fin.open(argv[1]);

  ppmdump(fin);

  fin.close();
}

-----------------------------------------------------------------
Hint: Note the mixed use of C and C++ style file I/O. This is
done both to showcase functions and for convenience.
-----------------------------------------------------------------
```