```
// This program demonstrates the return value of the stream
// object error testing member functions.
#include<iostream>
#include<fstream>
using namespace std;

// Function prototype
void showState(fstream &);

int main()
{
      int num = 10;

      // Open the file for output.
      fstream testFile("stuff.dat", ios::out);
      if (testFile.fail())
      {
            cout << "ERROR: cannot open the file.\n";
            return 0;
      }

      // Write a value to the file.
      cout << "Writing the value " << num << " to the file.\n";
      testFile << num;

      // Show the bit states.
      showState(testFile);

      // Close the file.
      testFile.close();

      // Reopen the file for input.
      testFile.open("stuff.dat", ios::in);
      if (testFile.fail())
      {
            cout << "ERROR: cannot open the file.\n";
            return 0;
      }

      // Read the only value from the file.
      cout << "Reading from the file.\n";
      testFile >> num;
      cout << "The value " << num << " was read.\n";

      // Show the bit states.
      showState(testFile);

      // No more data in the file, but force an invalid read operation.
      cout << "Forcing a bad read operation.\n";
      testFile >> num;

      // Show the bit states.
      showState(testFile);

      // Close the file.
      testFile.close();
      return 0;
}
```

```
//******************************************************************
// Definition of function showState. This function uses            *
// an fstream reference as its parameter. The return values of     *
// the eof(), fail(), bad(), and good() member functions are       *
// displayed. The clear() function is called before the function   *
// reutrns.                                                        *
//******************************************************************

void showState(fstream &file)
{
      cout << "File Status:\n";
      cout << " eof bit: " << file.eof() << endl;
      cout << " fail bit: " << file.fail() << endl;
      cout << " bad bit: " << file.bad() << endl;
      cout << " good bit: " << file.good() << endl;
      file.clear();    // Clear any bad bits
}
```