To verify that the terms are input in row-major order, we can compute the row-major location index $(r-1)*m+c$ for the (r,c) term of a matrix that has m columns. In row-major order, terms are input in increasing order of this index. It is actually easier to work with m + *index*, as this is just $r*m+c$. In the input code below, `OldIndex` is the $r*m+c$ value of the last term input and `NewIndex` the value for the current term. Row-major input can be verified by verifying that `OldIndex` is less than `NewIndex`.

```
template<class T>
istream& operator>>(istream& in, SparseMatrix<T>& x)
{// Input a sparse matrix.

   // input matrix characteristics
   cout << "Enter number of rows, columns, and terms"
        << endl;
   in >> x.rows >> x.cols >> x.terms;
   if (x.terms > x.MaxTerms) throw NoMem();

   // OldIndex = (row of last term) * x.cols +
   //         column of last term
   int OldIndex = 0;

   // input terms
   for (int i = 0; i < x.terms; i++) {
      cout << "Enter row, column, and value of term "
           << (i + 1) << endl;
      in >> x.a[i].row >> x.a[i].col >> x.a[i].value;
      // verify input
      int NewIndex = x.a[i].row * x.cols + x.a[i].col;
      if ((NewIndex <= OldIndex) || (x.a[i].value == 0))
         throw BadInput();
      OldIndex = NewIndex;
      }

   return in;
}
```

The codes are in the files `smatrix1.*`.