Math 236 **Problem Set 3** **Spring, 2001**

Due: Tuesday, Feb. 27 *but* Problem 1 due Friday, Feb. 23

1. In a MATHEMATICA notebook define MATHEMATICA functions `swap` and `scale` that effect the corresponding elementary row operations on a matrix. Test each of the two function you defined with examples of your own to show it works. Turn in printouts of the definitions and convincing testing.

   The function `swap` is used in the form `swap[M, i, k]` where `M` is a matrix and `i` and `k` are integers between 1 and the number of rows of `M`. (You may use any names you wish for the actual arguments.) It returns as result the matrix obtained by interchanging the `i`th and `j`th rows of `M`. For example:

```
   M = {{3, 5, 8, 1}, {2, 6, 3, 5}, {0, 2, 5, 1}}; M // MatrixForm
3 5 8 1
2 6 3 5
0 2 5 1
   swap[M, 1, 3] // MatrixForm
0 2 5 1
2 6 3 5
3 5 8 1
```

   The function `scale` is used in the form `scale[M, c, i]` where `M` is a matrix, `c` is any nonzero number, and `i` is an integer between 1 and the number of rows of `M`. It returns as result the matrix obtained by multiplying row `i` of `M` by the scalar `c`. For example, with the same `M` as before:

```
   M = {{3, 5, 8, 1}, {2, 6, 3, 5}, {0, 2, 5, 1}};
   scale[M, 7, 2]
 3  5  8  1
14 42 21 35
 0  2  5  1
```

2. (Counts as two problems.) In this problem you will carry out the Gauss-Jordan algorithm step-by-step in MATHEMATICA so as to find the reduced row-echelon forms of four particular matrices. The elementary row operations are to be effected by your functions `swap` and `scale` from Problem 1 together with my function `addrow`. You will also need my function `roundoff`.

   In case you are uncertain that your own definitions of `swap` and `scale` are correct, you can obtain versions of mine—at least if your are working at a PC rather than a Mac—that you can use without being able to see their definitions. See notebook `GJStepByStep.nb`.

   The function `addrow` is defined as follows (and also in `GJStepByStep.nb`):

```
   addrow[mat_List, c_?NumericQ, k_Integer, i_Integer] :=
      Module[{A = mat}, A[[i]] = A[[i]] + c A[[k]]; A]
```

   It is used in the form `addrow[M, c, k, i]` where `M` is a matrix, `c` is a number, and `k` and `i` are integers between 1 and the number of rows of `M`. It gives as result the matrix obtained by replacing the `i`th row of `M` with the sum of that row and `c` times the `k`th row. For example, with the same `M` as before:

```
   M = {{3, 5, 8, 1}, {2, 6, 3, 5}, {0, 2, 5, 1}}; M // MatrixForm
3 5 8 1
2 6 3 5
0 2 5 1
   addrow[M, -5, 1, 2]
  3   5   8 1
-13 -19 -37 0
  0   2   5 1
```

By following the instructions in notebook GJStepByStep, you will get your own individual set of four matrices M1, M2, M3, M4 to reduce. Some of them will be determined by random choice, with the random numbers determined, in part, from your student ID number.

The names M1, M2, M3, M4 are *protected*: you will not be able to assign new values to them. So the first thing you should do before row-reducing each matrix is assign it to a new name, for example:

```
M = M1 ; M // MatrixForm
```

Then do all the elementary row operations on M—

```
M = swap[M, 1, 3]; M // MatrixForm
```

—etc.

Here are two **precautions you should take** to guard against spurious results due to roundoff errors made by the computer (with its limited precision).

- Because MATHEMATICA ordinarily displays far fewer digits than it actually stores, a matrix entry you see displayed may not be the actual value. For example, suppose at some stage of the reduction your matrix displays as:

  ```
     M // MatrixForm
  1 4  0  0 0
  0 0 5.  3 4
  0 0  7 -4 6
  ```

  You are ready to scale row 2 by 1/5 to make the leading entry 1. But you don't know that the entry is exactly 5; it might really be 5.000000238. Use indexing ([[...]]) to get the actual value of that entry, and then you can let MATHEMATICA form its reciprocal. For example,

  ```
  M = scale[M, 1/M[[2, 3]], 2]; M // MatrixForm
  ```

  Use indexing similarly when applying addrow.

- Despite the preceding precaution, after you do a step of the reduction, your matrix may contain "small" entries that really "ought" to be 0 but are not. We shall consider a number "small" when its absolute value is strictly less than 10^{-12}, which in MATHEMATICA notation may be also be entered as 10^-12 or 1.*^-12). For example, you might get:

  ```
     M // MatrixForm
  1 0 0 0
  0 1 0 1.34*^-14
  0 0 1 0
  ```

  Probably that 1.34*^-14 (meaning 1.34×10^{-14}) is really supposed to be zero exactly. To make such small entries exactly zero, you should use the function roundoff that is automatically defined when you open the notebook GJStepByStep.nb. For example, starting with the last M displayed above:

  ```
     M = roundoff[M]; M // MatrixForm
  1 0 0 0
  0 1 0 0
  0 0 1 0
  ```

3. Do page 47, Exercise 6.

4. Do page 47, Exercises 24, 26, 28, and 30; instructions precede Exercise 24.