Math 236 Problem Set 5 Spring, 2001

Due date: Monday, March 12, *except* # 8 due Friday, March 16

1. Do page 47, Exercise 10.
2. (a) Do page 70, Exercise 4.

   (b) Answer same question as in (a), but for $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 5 \end{bmatrix}$.
3. Do page 71, Exercise 28. It's OK to use MATHEMATICA here.
4. Do page 72, Exercise 41 (a)–(c) only. Justify your answers!
5. Do page 85, Exercise 20. If the formula is true, prove it; if it is not true, give a specific example where it fails.
6. Do page 86, Exercise 40.
7. (a) Find at least two 2×2 matrices A other than I_2 for which $A^2 = I_2$.

   (b) Do page 89, Exercise 71.
8. (*Counts as two problems.*) Define a MATHEMATICA function `invert` that takes as argument a single *square* $n \times n$ matrix `A` for arbitrary n (in the usual form of a list of lists) and that returns as result: the inverse of A (in the form of a list of lists) in case `A` is invertible but `Null` in case `A` is not invertible. For example:

```
   invert[{{1, 3}, {2, 5}}]
{{-5, 3}, {2, -1}}

   invert[{{1, 3}, {2, 6}}]
(* no result is produced, since the result is Null *)
```

   The method to use is to join the identity matrix to the argument and use the reduced row-echelon form of the result. Some relevant MATHEMATICA functions are:

   - The built-in function `IdentityMatrix`.
   - For joining one matrix alongside another, the function `AppendRows` in the Standard AddOn package `LinearAlgebra‘MatrixManipulation‘`. (Don't let the name of this function confuse you: you are appending rows of one matrix to the rows of another matrix.) To load the package, use `Needs` or `Get` or the abbreviation `<<`.
   - For obtaining the reduced row-echelon form, the built-in function `RowReduce` (which is more reliable than `GJ`). Thus you do *not* need `scale`, `swap`, `addrow`, or `roundoff`.
   - For extracting blocks of columns of a matrix, index using `All` as the first argument.

- For comparing things, the built-in function `===` (that's *three* equal signs in a row), which is an abbreviation for `SameQ`; this is *instead of* the built-in function `==` (that's two equal signs in a row), which is an abbreviation for `EqualQ`.

While designing and debugging your `invert`, you may wish to compare its results with those of the built-in function `Inverse`.

Test your function `invert` in the manner that is prescribed in the notebook `About invert.nb`, available from the course web site.

For **extra credit**: In the case that `A` is not invertible, `invert` should also issue an appropriate warning message via the `Message` mechanism.