The Basis other problems

Coding the Matrix, 2015

For auto-graded problems, edit the file The_Basis_other_problems.py to include your solution.

Problem 1: Write and test a procedure `is_superfluous(S, v)` with the following spec:

- *input:* a set S of vectors, and a Vec $\boldsymbol{v}$ in S
- *output:* `True` if the span of the vectors in S equals the span of the set of vectors in S that are not equal to $\boldsymbol{v}$

Your procedure should not use loops or comprehensions but can use procedures defined in the module `matutil` and can use the procedure `solve(A,b)` defined in `solver` module. Your procedure will most likely need a special case for the case where len(S) is 1.

Note that the `solve(A,b)` always returns a vector $\boldsymbol{u}$. It is up to you to check that $\boldsymbol{u}$ is in fact a solution to the equation $A\boldsymbol{x} = \boldsymbol{b}$. Moreover, over $\mathbb{R}$, even if a solution exists, the solution returned by `solve` is approximate due to roundoff error. To check whether the vector $\boldsymbol{u}$ returned is a solution, you should compute the residual $\boldsymbol{b} - A * \boldsymbol{u}$, and test if it is close to the zero vector:

```
>>> (b - A*u).is_almost_zero()
True
```

For a vector over the real numbers, the `is_almost_zero()` method checks whether the dot-product of the vector with itself is less than 10^{-20}. This isn't a very principled test but it will work for this course. A vector over $GF(2)$ is of course considered "almost zero" only if it is truly a zero vector.

Problem 2: Write and test a procedure `is_independent(S)` with the following spec:

- *input:* a set S of vectors
- *output:* `True` if the set is linearly independent

Your algorithm for this procedure should be based on the Span Lemma. You can use as a subroutine any one of the following:

- the procedure `is_superfluous(S, v)` from the previous problem or
- the `solve(A,b)` procedure from the `solver` module (but see the provisos in the previous problem).

You will need a loop or comprehension for this procedure.

Note: This problem illustrates one way of thinking about linear dependence but don't make the mistake of making it your only or even your main way of thinking about it. For most purposes, it is best to go back to the *definition* of linear dependence.

Also, the intended algorithm for this problem is *not* a particularly efficient algorithm for testing linear dependence. We'll see some better methods later.

Problem 3: Write and test a procedure `exchange(S, A, z)` with the following spec:

- *input:* A set S of vectors, a set A of vectors that are all in S (such that len$(A) <$ len(S)), and a vector $\boldsymbol{z}$ such that $A \cup \{z\}$ is linearly independent
- *output:* a vector $\boldsymbol{w}$ in S but not in A such that

$$\text{Span } S = \text{Span } (\{\boldsymbol{z}\} \cup S - \{\boldsymbol{w}\})$$

Your procedure should follow the proof of the Exchange Lemma. You should use the `solver` module or the procedure `vec2rep(veclist, u)` from a previous problem. You can test whether a vector is in a list or set `C` using the expression `v in C`. Note that S need not be linearly independent.