CECS 229 Programming Assignment #6

Due Date:

Sunday, 4/28 @ 11:59 PM

Submission Instructions:

Complete the programming problems in the file named `pa6.py` . You may test your implementation on your Repl.it workspace by running main.py . When you are satisfied with your implementation,

1. Submit your Repl.it workspace
2. Download the file `pa6.py` and submit it to the appropriate CodePost auto-grader folder.

Objectives:

1. Apply Gaussian Elimination to solve the system $A\overrightarrow{x} = \overrightarrow{b}$.
2. Use Lp -norm to calculate the error in a solution given by applying Gaussian elimination.
3. Use the REF of the augmented matrix for the system $A\overrightarrow{x} = \overrightarrow{b}$ to determine if it has one solution, no solution, or infinitely-many solutions.
4. Determine the number of free variables that the system $A\overrightarrow{x} = \overrightarrow{b}$ has if it has infinitely-many solutions.
5. Use Gram-Schmidt Process to find an orthonormal basis for a given set of vectors.

Notes:

Unless otherwise stated in the FIXME comment, you may not change the outline of the algorithm provided by introducing new loops or conditionals, or by calling any built-in functions that perform the entire algorithm or replaces a part of the algorithm.

Problem 1

Complete the function `norm(p, v)` that returns the L_p-norm of `Vec` object `v`. Recall that the L_p-norm of an n-dimensional vector $\overrightarrow{v}$ is given by, $||v||_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p}$. Input `p` should be of the type `int`. The output norm should be of the type `float`.

In [ ]:
```
def norm(v, p):
    """
    returns the p-norm of Vec v
    INPUT:
        p - an integer determining the norm to be calculated
        v - the Vec object for which the norm will be applied
    OUTPUT:
        the norm as a float
    """
```

```
    # TODO: implement this function
    pass
```

Problem 2

Complete the helper function `_ref(A)` that applies Gaussian Elimination to create and return the Row Echelon Form of the given `Matrix` object `A`. The output must be of the type `Matrix`. The method should ***NOT*** modify the contents of the original `Matrix` object `A`. It should create and return a new `Matrix` object.

In [ ]:
```
import copy

def _ref(A):
    """
    returns the Row Echelon Form of the Matrix A
    INPUT: Matrix A
    OUTPUT: distinct Matrix object that is the
            Row-Echelon Form of A
    """
    matrix = Matrix(copy.deepcopy(A.rowsp))
    m, n = matrix.dim()
    # TODO: Finish implementation for this function
    return matrix
```

Problem 3

Create a function `rank(A)` that returns the rank of `Matrix` object `A` as an integer.

HINT: Look at Claim 2.4.1 of the "Gaussian Elimination Lecture Notes"

In [ ]:
```
def rank(A):
    """
    returns the rank of the given Matrix object
    as an integer
    """
    # TODO: implement this function
    pass
```

Problem 4

Implement the function `gauss_solve(A, b)` that solves the system $A\overrightarrow{x} = \overrightarrow{b}$. The input `A` is of the type `Matrix` and `b` is of the type `Vec`.

- If the system has a unique solution, it returns the solution as a `Vec` object.
- If the system has no solution, it returns `None`.
- If the system has infinitely many solutions, it returns the number of free variables (`int`) in the solution.

In [ ]:
```
def gauss_solve(A, b):
    """
    returns the solution to the system Ax = b
    if the system has a solution.  If the system
    does not have a solution, None is returned.
```

```
    If the system has infinitely-many solutions,
    the number of free variables as an 'int' is returned
    INPUT:
        A - a Matrix object
        b - a Vec object

    OUTPUT:
        Vec object if the system has a unique solution
        None if the system has no solution
        int if the system has infinitely-many solutions
    """
    # TODO: Implement this function
    pass
```

Problem 5

Implement the function `gram_schmidt(S)` that applies the Gram-Schmidt process to create an orthonormal set of vectors from the vectors in set `S`. The function assumes that the set `S` is linearly independent.

INPUT:

- `S` a linearly independent set of `Vec` objects

OUTPUT:

- a set of `Vec` objects representing orthonormal vectors.

HINT:

If $S = \{\overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_n}\}$ is a set of linearly independent vectors, then Gram-Schmidt process returns the set $\{\overrightarrow{u_1}, \overrightarrow{u_2}, \ldots, \overrightarrow{u_n}\}$ where,

- $\overrightarrow{u_i} = \frac{1}{||\overrightarrow{w_i}||_2}\overrightarrow{w_i}$ for $i = 1, 2, \ldots n$,

and

- $\overrightarrow{w_1} = \overrightarrow{x_1}$
- $\overrightarrow{w_i} = \overrightarrow{x_i} - \sum_{j=1}^{i-1} proj_{\overrightarrow{w_j}}(\overrightarrow{x_i})$ for $i = 2, 3, \ldots n$

In [ ]:
```
def gram_schmidt(S):
    """
    returns the orthonormal basis of given set S
    INPUT: S - a set of linearly independent 'Vec' objects
    OUTPUT: An orthonormal set of 'Vec' objects
    """
    # TODO: Implement this function
    pass
```