Project 2
Introduction

In this project, you will implement algorithms to solve optimization problems formulated as linear and integer programming problems.

You should implement and run your code on your **local computer** in order to visualize the project, Mimir will only be used as a submission tool for this project.

In the end, only the **optimization.py** file should be submitted to the Mimir. You are only allowed to submit **3 times**.

As in previous programming assignments, this project includes an autograder for you to grade your answers on your local machine. This can be run with the command:

```
python3 autograder.py
```

Before using this, you may need to install Tkinter by using "sudo apt – get install python3 -tk" in Ubuntu

See the autograder tutorial in programming assignment 1 for more information about using the autograder.

The code for this project consists of several Python files, some of which you will need to read and understand in order to complete the assignment, and some of which you can ignore. You can download and unzip all the code and supporting files from Mimir.

Files you will edit

optimization.py	Where all your optimization code will reside.

Files you might want to look at

pacmanPlot.py	File that helps plotting feasible regions as pacman and ghosts.
test_cases/	Directory containing the test cases for each question

Files you will not edit

autograder.py	Project autograder

game.py	Logistics for Pacman world
ghostAgents.py	Agents to control ghosts
graphicsDisplay.py	Graphics for Pacman
graphicsUtils.py	Support for Pacman graphics
keyboardAgents.py	Keyboard interfaces to control Pacman
layout.py	Code for reading layout files and storing their contents
optimizationTestClasses.py	File that helps run the test cases
pacmanAgents.py	Agents to control pacman
pacman.py	The main file that runs pacman game
testClasses.py	General autograding test classes
testParser.py	Parses autograder test and solution files
textDisplay.py	ASCII graphics for Pacman
util.py	Data structures for implementing search algorithms

Files to Edit and Submit: You will fill in portions of optimization.py during the assignment. You should submit optimization.py only. Please *do not* change the other files in this distribution or submit any of our original files other than this file.

Evaluation: Your code will be autograded for technical correctness. Please *do not* change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only; *please* don't let us down. If you do, we will pursue the strongest consequences available to us.

The Pacman graphics in this assignment are based on the Pacman AI projects developed at UC Berkeley, http://ai.berkeley.edu.

Question 1 (7 points): Find Intersections

Implement the function `findIntersections` in `optimization.py` to return a list of all intersection points given a list of linear inequality constraints.

You can test your implementation with the following:

```
python3 autograder.py -t test_cases/q1/test2D_1

python3 autograder.py -q q1
```

You may want to start with the first set of test cases working. In 2-D, you can use your plotting skills to help visualize and debug your work.

You will probably want to take advantage of numpy library for doing linear algebra in Python. The following numpy operations will be particularly useful:

- `np.dot(A,x)`: Matrix-vector multiplication
- `np.linalg.solve(A,b)`: Solve the system of linear equations $Ax=b$ for x
- `np.linalg.matrix_rank(A)`: A matrix has to be square ($N \times N$) and full rank (rank = N) in order for `np.linalg.solve` to work. *Note:* What is the relationship between rank and intersecting hyperplanes (defined by rows of A and b)?
- `A[(1,3,5), :]`: Select the first, third, and fifth rows of A(and all columns)
- *Tip:* Because it is problematic to compare floating point values, you will still want to allow values to be feasible if they are within `1e-12` of their corresponding limit.
- You can test your implementation with the following:

```
- python3 autograder.py -t test_cases/q2/test2D_1
- python3 autograder.py -q q2
```

- Question 3 (2 points): Find Optimal Intersection

- Implement the function `solveLP` in `optimization.py` to return a feasible intersection point that minimizes the objective. Your algorithm can simply step through all feasible intersection points, looking for the one that gives the minimal objective value. You may assume that if a solution exists that it will be bounded, i.e. not infinity.
- You can test your implementation with the following:

```
- python3 autograder.py -t test_cases/q3/test2D_1
- python3 autograder.py -q q3
```

Question 4 (2 points): Problem Formulation (LP)

Represent the following problem as a linear program. Implement the function `wordProblemLP` in `optimization.py` to construct constraints and a cost vector, then call your `solveLP` function to return the optimal solution.

Pat is packing for his trip to Hawaii. He wants to bring sunscreen and Tantrum energy drink, the two most important things to pack for a vacation. Help Pat determine how many fluid ounces of suncreen and how many fluid ounces of Tantrum he wants to pack while making sure he doesn't pay extra for a heavy bag.

Assume that Pat is checking a suitcase filled entirely with suncreen and Tantrum. Pat needs at least 20 fluid ounces of suncreen and at least 15.5 fluid ounces of Tantrum. Pat's suitcase can fit 100 units of stuff. A fluid ounce of sunscreen takes 2.5 units of space and a fluid ounce of Tantrum takes 2.5 units of space. Furthermore, to avoid baggage fees, the suitcase can weight at most 50 pounds. A fluid ounce of sunscreen weighs 0.5 pounds and a fluid ounce of Tantrum weighs 0.25 pounds. We want to make sure Pat has a good time in Hawaii, so we want to pack things that maximizes Pat's happiness in Hawaii. Each fluid ounce of sunscreen gives Pat a utility of 7 while each fluid ounce of Tantrum gives Pat a utility of 4. How can we pack so that Pat has the best vacation ever?

You can test your implementation with the following:

```
python3 autograder -q q4
```

Question 5 (7 points): Branch and Bound

Now that you're a linear programming wizard, let's take things a step farther. Solve *integer programming* problems by implementing the branch and bound algorithm in the `solveIP` function in `optimization.py`.

Tip: You can check to see if floating point values in your program are integers, by seeing if they are within `1e-12` of the nearest integer value.

You can test your implementation with the following:

```
python3 autograder.py -t test_cases/q5/test2D_1

python3 autograder.py -q q5
```

Food Distribution

No Poverty and Zero Hunger are listed as the top two Sustainable Development Goals by the United Nations. Food rescue service provides a promising way to reduce food waste, overcome food insecurity and improve environmental sustainability. Organizations providing food rescue service rescue the surplus food from different food providers and redistributing to local communities that are in need of food.

For questions 6 and 7, you will help a food rescue organization FS to decide how to redistribute the food in an efficient way. The problem is abstracted in the following way: There are M food providers (referred to as providers) and N local communities in need of food (referred to as communities). Community j needs at least C_j integer units of food. The transportation cost per unit of food from provider i to community j is $T_{i,j}$.
When transporting food from a provider to a community, trucks cannot be overweight. All trucks have the same weight limit per truck, and only one truck moves between provider i and community j. The food coming from provider i has weight Wiper unit.
We need to determine the integer number of food units to transport from each provider to each community, while meeting the above constraints and minimizing transportation costs.

Question 6 (2 points): Problem Formulation (IP)

Represent the following food distribution problem as an integer program. Implement the function `wordProblemIP` in `optimization.py` to construct constraints and a cost vector, then call your `solveIP` function to return the optimal solution.

Before solving world hunger, we will practice with campus hunger following the food distribution setup above. In this specific case, we have three providers (Dunkin Donuts, Eatunique, and the Underground) and two communities (Gates and Sorrells).

The communities require at least the following number of units of food:

- Gates: 15
- Sorrells: 30

The truck weight limit is 30, and the weight per unit and transportation cost per unit are as follows:

	Weight	Gates	Sorrells
Dunkin Donuts	1.2	$12	$20
Eatunique	1.3	$4	$5
Underground	1.1	$2	$1

You can test your implementation with the following:

```
python3 autograder.py -q q6
```

Note: Some people find it more convenient to implement the more general food distribution in question 7 before implementing this specific problem.

Question 7 (3 points): Food Distribution IP

Implement the function `foodDistribution` in `optimization.py` to handle the general food distribution integer programming.

You can test your implementation with the following:

```
python3 autograder.py -q q7
```

Submission

Complete questions 1 through 7 as specified in the project instructions. Then upload `optimization.py` to Mimir.

Prior to submitting, be sure you run the autograder on your own machine. Running the autograder locally will help you to debug and expediate your development process. The autograder can be invoked on your own machine using the command:

```
python3 autograder.py
```

To run the autograder on a single question, such as question 3, invoke it by

```
python3 autograder.py -q q3
```

Note that running the autograder locally will **not** register your grades with us. Remember to submit your code below when you want to register your grades for this assignment.

The autograder on Mimir might take a while but don't worry, **as long as you submit before the due date, it's not late**.