Project 1: Beeline

Submission Due: March 8 by 11:59PM

Program Description:

For this project you will be building a game called Beeline. In this game, you will be playing as the queen bee, whose job it is to make sure that the hive has enough honey, which is generated from the pollen in flowers, for the coming winter. As the queen cannot actually leave the hive to collect pollen, she has both scout bees and worker bees to help in that task. Scout bees fly out to specified locations in the field and report back where flowers are located. Worker bees fly out to specified locations in the field, harvest pollen, and also report back where flowers are located. However, a bee can only be given a single task before it retires to the hive to make honey, and a flower can only be harvested for pollen once. Additionally, scattered across the field are pitcher plants that are irresistible to bees and will trap a bee, thus preventing it from returning to the hive. The field itself will be a rectangle and can contain the symbols H (for the hive), various letters for different flowers, U (for a flower that has been used), P (for a pitcher plant), and space for an empty spot in the field. The field could look similar to the following as the game is played:

```
  0 1 2 3 4 5 6 7 8 9
0| | | | | | | |L|G|L|
1| | | | | | | |G|C|G|
2| |B| | | | | | |G| |
3| | | | | | | | | | |
4| | | | | | | | | | |
5| | | | |H| | | | | |
6| | | | | | | | | | |
7| | | | | | | | | | |
8| |U|U| | | | | | | |
9| |U|U| | | | | | | |
```

Thus, it is up to you, the queen, to manage your bees accordingly and ensure the prosperity of the hive!

Program Requirements

- The code for the program should be divided into two files:
    - A **BeeFunctions.py** file that contains all of the requested functions except the `main()`
    - A **Beeline.py** file that contains a `main()` and imports the **BeeFunctions** module
- Inside the **BeeFunctions.py** file you should have:
    - A function to load the flower list file, that has no parameters, returns a `Dictionary` of `Tuples` containing the information regarding the flowers and the amount of pollen each is worth, and provides the following functionality:
        - Creates a new `Dictionary`
        - Prompts the user for the name of the file containing the flower to pollen mapping
            - This file will be in CSV format

        - Check if the file exists using the appropriate functions and if it does not, repeatedly prompt the user for a new filename until the provide the name of a file that does exist
        - Open the file for reading
        - Read one line at a time from the file and for each line:
            - Remove any leading and trailing whitespace from the line
            - Split the line on commas, and store the resulting `List`
            - Using the flower's letter as a key, store a new `Tuple` containing the flower's letter, name, and pollen count in the `Dictionary`
        - Close the file
        - Return the `Dictionary`
    - A function to create the field, that takes in a `Dictionary` of `Tuples` containing flower information and returns a two-dimensional `List`, and provides the following functionality:
        - Creates a variable to store a new two-dimensional `List`
        - Prompt the user for the name of the file containing the field, this file will be in CSV format
        - Check if the file exists using the appropriate functions and if it does not, repeatedly prompt the user for a new filename until the provide the name of a file that does exist
        - Open the file for reading
        - Using a loop, read and parse each line to create your two-dimensional `List`
            - For each line, after splitting the line, you will need to examine each character and if a character is not a hive, a pitcher plant, an empty space or one of the flowers in the flower mapping (use the `Dictionary` here), `raise` a `TypeError` exception containing a message informing the user that that particular flower is not a known flower type
        - Return the newly constructed two-dimensional `List`
    - A function to make a copy of a field, that takes in a two-dimensional `List` as a parameter and returns a two-dimensional `List`, and provides the following functionality:
        - Create a variable to store the new two-dimensional `List`
        - Generate a copy of the passed in two-dimensional `List`, such that
            - The copy has the same dimensions as the passed in `List`
            - All flowers and pitcher plants are converted to spaces
            - All spaces remain as spaces
            - The H is in the same position in the copy as it is in the original
        - Return the newly constructed two-dimensional `List`
    - A function to print the field, that takes in one, two-dimensional `List`, returns no values, and provides the following functionality:
        - Prints out the passed in two-dimensional `List` in a pleasing grid format with index numbers along the top and left-hand side of the grid to help the player choose a location to send their bees (see the example outputs). Make sure the index numbers along the top of the grid are directly above the column they index

- A function to output the game's introductory information, that takes in a `Dictionary` of `Tuples` containing flower information, the number of scout bees, the number of worker bees, and the amount of pollen needed to win the game, returns no values and informs the user of the rules of the game. Your intro must include the following information in some form:
  - You play as the queen bee trying to produce honey from the pollen of flowers
  - You have two kinds of bees, scouts and workers
  - Scouts fly to a location and reveal the flowers in a 3x3 grid centered on that location
  - Workers fly to a location, reveal the flowers in a 3x3 grid centered on that location, and harvest polled from any unharvested flowers
  - You only have some number scout bees and some number worker bees to harvest some amount of pollen
    - Use the parameter variables for these values
  - A bee can only be sent out once, and a flower can only be harvested once
  - A warning about pitcher plants and their effect
  - A listing of the various flowers of the area containing their letter, their name, and how much pollen each one is worth (Use your `Dictionary` here)
  - See the example output for formatting
- A function to check the area a bee is sent to, that takes in one, two-dimensional `List`, representing the hidden `List`, one, two-dimensional `List`, representing the visible `List`, x and y integer coordinates, a String containing a single character representing the type of bee, and the `Dictionary` of `Tuples` containing flower information, and returns the integer count of the amount of pollen harvested, and provides the following functionality:
  - Create a variable to store the amount of polled harvested
  - Check if the x and y coordinates are outside the bounds of the two-dimensional `List`, and if they are outside the bounds, inform the user their bee has left the field and has been lost
  - Otherwise:
    - Examine a 3x3 grid centered on the passed in x and y coordinates and if any of the flowers in the area are pitcher plants inform the user that the bee must have fallen into the pitcher plan because it did not return, and immediately return the amount of pollen harvested (which should be 0)
    - Examine a 3x3 grid centered on the passed in x and y coordinates and
      - If the bee is a scout bee, copy all flower letters and U's in the 3x3 grid from the hidden `List` to the visible `List`
      - If the bee is a worker bee, copy all U's in the 3x3 grid from the hidden `List` to the visible `List`, and for each flower letter in the hidden `List`, convert it to a U and increment the amount of harvest pollen by the amount for each flower (Use your `Dictionary` here)
  - Return the amount of pollen harvested

- Inside the **Beeline.py** file you should have:
    - A `main` function which takes in no parameters, returns no values, and provides the following functionality:
        - Call the function to load the flower list file and store the resulting `Dictionary` of `Tuples`
        - Call the function to create the field and store the resulting two-dimensional `List` representing the hidden version of the field
            - Note this function could have a `TypeError` exception generated inside of it so it will require a `try`/`except` statement around it. If the exception is generated, catch it and store it in a variable, print the message inside the exception, and immediately exit the program with a status code of -1
        - Call the function to make a copy of a field and store the resulting two-dimensional `List` representing the visible version of the field
            - Be sure to pass it the hidden version of the field!
        - Create appropriate variables to store the number of scout bees (5), worker bees (5), current amount of harvested pollen(0), and the amount of pollen needed to win, which should be a constant variable (20)
        - Call the function to output the game's introductory information and pass it all of the necessary information
        - Create a loop that allows the user to keep playing the game as long as they have worker bees left or have not harvested enough pollen. Inside the loop you should:
            - Inform the user how many scout and worker bees remain and how much pollen has currently been harvested, as well as what the H and U symbols on the field mean
            - Output the visible version of the field
            - Prompt the user for which type of bee they would like to send out (S for scout, W for worker)
            - If the user chooses scout, check if enough scout bees remain
                - If not inform the user no more scout bees remain
                - If there are remaining scout bees:
                    - Prompt the user for x and y coordinates
                    - Decrement the number of scouts by 1
                    - Inform the user you are sending out a scout
                    - Call the function to check the area a bee is sent to and pass in all of the appropriate information
            - If the user chooses worker, check if enough worker bees remain
                - If not inform the user no more worker bees remain
                - If there are remaining worker bees:
                    - Prompt the user for x and y coordinates
                    - Decrement the number of workers by 1
                    - Inform the user you are sending out a worker

                - Call the function to check the area a bee is sent to and pass in all of the appropriate information
            - Be sure to increment the amount of harvested pollen by the value returned from the function
        - If the user chooses a bee type that is not scout or worker, inform them of their mistake
    - Once the user has run out of workers and/or harvested enough flowers inform them they have won or lost depending on if they have enough harvested pollen, using an appropriate message
  - Call your `main()` function
- Your program should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, date, and brief description), comments for each variable, docstrings for each function, and commented blocks of code.
- You are not allowed to use any globally scoped variables in this project. All variables must be locally scoped.
- See sample input and output files to test and calibrate your program

Example Output

See the example output files

Submission

- Your program will be graded largely upon whether it works correctly
- Your program will also be graded based upon your program style. This means that you should use comments (as directed), meaningful variable names, and a consistent indentation style
- You must submit all of your .py files to Canvas by the due date and time
- Projects will be accepted up to two days late, at a 10% penalty per 24-hour period after the due date
- Projects are meant to be individual coding assignments, so no collaboration is allowed. This includes downloading code off of the internet. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- You are not allowed to use generative AI on this project. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- Be sure that you have tested the version of the program you wish to submit to make sure it works correctly. You will not be allowed to resubmit work after the deadline

Rubric

The entire assignment is worth 100 points and partial credit is possible. No credit will be given for portions of the program that cannot be tested due to the program crashing.

- **Program Executes Successfully**
    - If your program crashes due to syntax errors, you will lose 10 points, but we will attempt to fix minor issues (incorrect indentations, stray character, missing import) so that we can execute and test the program. We will not fix major issues that would require functionality to be further implemented, or a reorganization of logic in your code.
- **Loading the Flower List File (10 points)**
    - Program prompts the user for the filename and uses a loop to repeatedly reprompt for a new filename if the file does not exist (5 points)
    - Program constructs and returns the `Dictionary` of `Tuples` based on the information in the user specified file (5 points)
- **Creating the Hidden List (15 points)**
    - Program prompts the user for the filename and uses a loop to repeatedly reprompt for a new filename if the file does not exist (5 points)
    - Program raises a TypeError exception if a flower not in the `Dictionary` is detected in the input file (5 points)
    - Program constructs and returns the two-dimensional `List` based on the information in the user specified file (5 points)
- **Creating the Visible List (5 points)**
    - Program correctly creates and returns a new copy of the passed in two-dimensional `List` according to specification (5 points)
- **Checking the Area a Bee is Sent to (20 points)**
    - Program correctly determines if the user specified x,y coordinates are outside the bounds of the field, and informs the user if the coordinates are (5 points)
    - Program correctly determines if the bee has been trapped by a pitcher plant, informs the user if it has been, and returns 0 amount of pollen harvested (5 points)
    - Program correctly reveals flowers from the hidden field to the visible field if the player uses a scout bee (5 points)
    - Program correctly reveals flowers from the hidden field to the visible field, harvests any unharvested flowers by updating them from their respective letters to U in both Lists and returns the amount of pollen harvested if the player uses a worker bee (5 points)
- **Outputting the Field (5 points)**
    - Program outputs the field in a pleasing grid format (2 points)
    - Program outputs the index values along the top and left of the grid (3 points)
- **Game Introduction (5 points)**
    - Program outputs the instructions for the game along with all of the requisite information (5 points)
- **Program is divided into two files (5 points)**
- **Main Functionality (30 points)**
    - Program contains appropriate variables which are initialized correctly (5 points)
    - Program handles exception correctly (5 point)
    - Program repeatedly loops until the user has no worker bees left or harvests enough flowers (5 points)
    - Program displays appropriate information to the user regarding the remaining number of bees and harvested flowers (5 points)

    - Program correctly handles the user's choice of bees (5 points)
    - Program correctly informs the user whether they have won or lost based on the number of worker bees and amount of pollen harvested (5 points)
- **Program contains sufficient comments (5 points)**

Bonus

For 10 bonus points, write and submit a program named **fieldGenerator.py** that will create and save a randomly generated field to a file. You will need to prompt the user for the name of the file to save to, the dimensions of the field, the number of flowers, the number of pitcher plants, the flower to pollen mapping file, and then generate a field with the user specified number and types of flowers randomly dispersed throughout the map, as well as the number of pitcher plants and a randomly placed beehive, and save the field to a file using the same format as the example files. Be sure not to overwrite any flowers that have already been placed when placing new flowers and the beehive.