Member ID: _____

Time: _____

Rank: _____

BUSINESS PROFESSIONALS of AMERICA
Giving Purpose to Potential

PYTHON PROGRAMMING

(355)

NATIONAL 2023

PRODUCTION (90 minutes)

Magic Squares _____ (500 points)

INDUSTRY CERTIFICATION (50 minutes)

IT Specialist Python Certification

General Guidelines:

Failure to adhere to any of the following rules will result in disqualification:

1. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
2. Electronic devices will be monitored according to ACT standards.
3. Your name and/or school name should *not* appear on work you submit for scoring.

PART ONE Instructions: Production
Once you have completed listed below, upload all competition materials (including project files, etc.) to the National BPA Pre-submit System at presubmit.bpa.org.

Compress all files in a single archive (zip file) and use the following naming convention for your documents:

PYTHON-MemberID.zip

PART TWO Instructions: Industry Certification
This exam requires IT Specialist Python Certification. Please do not leave your seat or exit the room until you have completed the certification. After the certification has been administered, you are permitted to leave.

Development Standards:

- Your Code must use a consistent variable naming convention.
- All functions (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any).
- If you create a class, then you must document the class and its methods.

Magic Squares

A magic square is a two-dimensional arrangement of positive integers with a special property: All rows, columns and diagonals total the same number, and each number is only used once.

Example A: (3x3 Square) – IS MAGIC

8	1	6
3	5	7
4	9	2

Figure 1

Row 1: 8 + 1 + 6 = 15
Row 2: 3 + 5 + 7 = 15
Row 3: 4 + 9 + 2 = 15
Column 1: 8 + 3 + 4 = 15
Column 2: 3 + 5 + 7 = 15
Column 3: 6 + 7 + 2 = 15
Diagonal (top left to bottom right): 8 + 5 + 2 = 15
Anti-Diagonal (bottom left to top right): 4 + 5 + 6 = 15
Each cell of the square has a unique number 1 through 9 with no repeats or omissions.

NOTE: This square is magic, because of all statements above are true.

Example (3x3 Square) – IS NOT MAGIC

1	8	6
3	5	7
4	9	2

Figure 2

Row 1: 1 + 8 + 6 = 15
Row 2: 3 + 5 + 7 = 15
Row 3: 4 + 9 + 2 = 15
Column 1: 1 + 3 + 4 = 8
Column 2: 8 + 5 + 9 = 22
Column 3: 6 + 7 + 2 = 15
Diagonal (top left to bottom right): 1 + 5 + 2 = 8
Anti-Diagonal (bottom left to top right): 4 + 5 + 6 = 15
Each cell of the square has a unique number 1 through 9 with no repeats or omissions.

NOTE: This square is not magic, because multiple sums are not equal.

Input Format

Create a python source file, **magic_square.py**, that will read from a data file, **magic.txt** (file contents are listed in Figure 3 below). The first line will contain a positive integer 2 < N < 6 which represents the number of columns and rows in the square. The next N lines will each contain N positive integers separated by spaces.

Figure 3: magic.txt (file contents)

```
3
2 7 6
9 5 1
4 3 8
3
1 2 3
4 5 6
7 8 9
4
7 12 1 14
2 13 8 11
16 3 10 5
9 6 15 4
5
17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9
4
16 3 2 13
5 10 11 8
6 9 7 12
4 15 14 1
```

Program Requirements

- Competitor member ID (########) is commented at the top of the source code
- The program will read the input file and loop until the end of the input file to determine whether each square is magic or not.
    - Input is stored in a two-dimensional array called **square.**
- The program contains a function called **is_magic_square** which takes a square as input and returns a tuple **(result, reason)**, where **result** is a boolean indicating whether the square is a magic square, and **reason** is a string giving the reason why the square is not magic (if **result** is **False**).
    - The function will determine that the sum of each individual row, column, and the two diagonals are equal.
    - The function will determine whether each array cell contains the numbers 1 through N^2 (where N=the number of rows and columns) and that each number is used once and only once.

Output Format

Your python program must write to a file called results.txt the following information:

- Output the square (N items per line) that was originally read in from the file (**magic.txt**)
- State whether the square was either **MAGIC** or **NOT MAGIC** on the next line.
    - If **NOT MAGIC**, the program must include the reason the square is not magic.
        - Each reason below must be accounted for in the function:
            - A row sum is incorrect
            - A column sum is incorrect
            - Main diagonal sum incorrect
            - Anti-diagonal sum incorrect
            - Invalid or duplicate number was detected

Figure 4: Below is the expected formatted file (**results.txt**) based on the input file (**magic.txt)**.

2 7 6
9 5 1
4 3 8
MAGIC

1 2 3
4 5 6
7 8 9
NOT MAGIC: A row sum is incorrect

7 12 1 14
2 13 8 11
16 3 10 5
9 6 15 4
MAGIC

17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9
MAGIC

16 3 2 13
5 10 11 8
6 9 7 12
4 15 14 1
NOT MAGIC: A column sum is incorrect

Solution and Project

The source code is present on the flash drive under a folder named with the competitors' member ID, including magic.txt	___	10 points
Main python file is called magic_square.py	___	10 points

Program Execution (Runtime)

The program runs/executes from the USB flash drive using python command line.	___	10 points

If the program does not execute, then the remaining items in this section receive a score of zero.

The program correctly reads the data in the file (**magic.txt)** line by line.	_____	10 points
The program creates the output file (**results.txt)** and follows the provided format in	_____	10 points
The program properly outputs the originally read square line by line.	_____	20 points
The program correctly determines if a square is MAGIC or NOT MAGIC.	_____	50 points
If NOT MAGIC, the program correctly outputs the possible reason.	_____	20 points
The program properly loops until the end of the input file is processed.	_____	10 points

Source Code Review (magic_square.py)

A comment containing the Member ID is present	_____	10 points
A comment containing the program description is present	_____	10 points
Source code is present to read from file (**magic.txt**) and loop until end of file	_____	10 points
is_magic_square function is generated and accepts the square array as input	_____	10 points
is_magic_square function contains appropriate code to check sum of rows	_____	50 points
is_magic_square function contains appropriate code to check sum of columns	_____	50 points
is_magic_square function contains appropriate code to check sum of the diagonal	_____	50 points
is_magic_square function contains appropriate code to check sum of the anti-diagonal	_____	50 points
is_magic_square function contains appropriate code to check that each array cell contains the numbers 1 through N^2 (where N=the number of rows and columns) and that each number is used once and only once.	_____	50 points
is_magic_square function returns a tuple **(result, reason)**, where **result** is a boolean indicating whether the square is a magic square, and **reason** is a string giving the reason why the square is not magic (if **result** is **False**).		
Source code is present to output to file (**results.txt**) the original square line by line	_____	20 points
Source code is present to output to file (**results.txt**) the determination of **MAGIC** or **NOT MAGIC** and the reason for being **NOT MAGIC**	_____	20 points
Code uses a consistent variable naming convention and data types	_____	20 points

Total Points Possible ____ / 500 points