Assignment 1

CS329e - Elements of Software Design
Infinite Spiral of Numbers
(100 points)
Due Date on Canvas and Gradescope

1 Description

Consider the natural numbers laid out in a square spiral, with 1 occupying the center of the spiral. The central 11 x 11 subset of that spiral is shown in the table below.

111	112	113	114	115	116	117	118	119	120	121
110	73	74	75	76	77	78	79	80	81	82
109	72	43	44	45	46	47	48	49	50	83
108	71	42	21	22	23	24	25	26	51	84
107	70	41	20	7	8	9	10	27	52	85
106	69	40	19	6	1	2	11	28	53	86
105	68	39	18	5	4	3	12	29	54	87
104	67	38	17	16	15	14	13	30	55	88
103	66	37	36	35	34	33	32	31	56	89
102	65	64	63	62	61	60	59	58	57	90
101	100	99	98	97	96	95	94	93	92	91

Table 1: Spiral of Numbers

This spiral has several interesting features. The southeast diagonal has several prime numbers (3, 13, 31, 57, and 91) along it. The southwest diagonal has a weaker concentration of prime numbers (5, 17, 37) along it.

To construct the spiral we start with 1 at the center, with 2 to the right, and 3 below it, 4 to the left, and so on. A part of the problem for this assignment is to figure out the rule to fill the spiral for an arbitrary size. Once you have that rule you can complete the rest of the assignment.

In this assignment your task is to implement a python program with the name **Spiral.py**.

You program should have the following input and output

Input:

You will read your input data from a file called spiral.in. The format of the file will be as follows:

11
1
42
110
91

The first line will be the dimension of the spiral. It will always be odd and greater than 1 and less than 100. This will be followed by an arbitrary number of lines. There will be a single number on each line. These numbers will be numbers inside the spiral. Some of these numbers will be interior numbers, others will be numbers on the edge, and yet others will be numbers at the corners of the spirals. Assume that the input file that we will be testing your program will be valid.

Output:

For each of the numbers inside the spiral, your output will be the sum of all the numbers adjacent to this number, but not including this number.

For the above input file you will output on the console:

44
382
477
239

We get 44 by adding the numbers adjacent to 1 (2 + 3 + 4 + 5 + 6 + 7 + 8 + 9). Similarly we get 239 by adding the numbers adjacent to 91 (57 + 90 + 92).

You will read your input from stdin like so:

```
 Mac:
python3 Spiral.py < spiral.in

Windows:
python Spiral.py < spiral.in
```

We will use our own input file to test your program. You must read the input in the format described above.

Once you read the first line from the input file you will create a 2-D list with the spiral of numbers. Then from the 2-D list you will obtain the sum of adjacent numbers for a given number in the spiral and print it. The number of lines of input will be arbitrary and greater than 1.

The file that you will be turning in will be called Spiral.py. You will follow the standard coding conventions in Python. Here is the format of your code:

```
#  File: Spiral.py
#  Description:
#  Student Name:
#  Student UT EID:
#  Partner Name:
#  Partner UT EID:
#  Course Name: CS 313E
#  Unique Number:
#  Date Created:
#  Date Last Modified:

# Input: n is an odd integer between 1 and 100
# Output: returns a 2-D list representing a spiral
#         if n is even add one to n
def create_spiral ( n ):

# Input: spiral is a 2-D list and n is an integer
# Output: returns an integer that is the sum of the
#         numbers adjacent to n in the spiral
#         if n is outside the range return 0
def sum_adjacent_numbers (spiral, n):

def main():
  # read the input file
  # create the spiral
  # add the adjacent numbers
  # print the result

if __name__ == "__main__":
  main()
```

You may not change the names of the functions listed. They must have the functionality as given in the specifications. You can always add more functions than those listed.

For this assignment you may work with a partner. Both of you must read the paper on Pair Programming[1] and abide by the ground rules as stated in that paper. If you are working with a partner then only one of you will be submitting the code. But make sure that your partner's name and UT EID is in the header. If you are working alone then remove the partner's name and eid from the header.

1.1 Turnin

Turn in your assignment on time on Gradescope system on Canvas. For the due date of the assignments, please see the Gradescope and Canvas systems.

1.2 Academic Misconduct Regarding Programming

In a programming class like our class, there is sometimes a very fine line between "cheating" and acceptable and beneficial interaction between students (In different assignment groups). Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. We want to be 100% precise, so that there can be no confusion.

[1]Read this paper about Pair Programming `https://collaboration.csc.ncsu.edu/laurie/Papers/Kindergarten.PDF`

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way – visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as **StackOverflow**. As far as going to the web and using Google, we will apply the **"two line rule"**. Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the "two line rule" inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them.

Furthermore, you should cite your sources. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.

We will use the following Code plagiarism Detection Software to automatically detect plagiarism.

- Staford MOSS

  `https://theory.stanford.edu/~aiken/moss/`

- Jplag - Detecting Software Plagiarism

  `https://github.com/jplag/jplag` and `https://jplag.ipd.kit.edu/`