CSC 120 **ICA-3**

Work with your neighbor. (This will be graded for participation only.)

1. This solution uses `split()` to create a list from each line of input and then iterates over each element of the list to strip the punctuation.

   ANS:
```
def print_some_words(filename,n):
    textfile = open(filename)
    for line in textfile:
        line = line.split()
        for word in line:
            clean_word = word.strip(".,?;")
            if len(clean_word) >= n:
                print(clean_word)
```

2. There are two things that have to be worked out for this problem: (1) the coordinates for the beginning of the diagonal for a given offset; and (2) the change in x- and y-coordinate values between successive elements of any diagonal.

   We can figure out the first quantity by just examining some different offset values; the key is to notice that negative offsets have to be treated differently than positive offsets. For any given diagonal, we can figure out the change in x- and y-coordinates by stepping through successive elements of a diagonal a few times. Once these two things are done by hand (on paper), programming up the solution is easier.

   As a first step, write the loop for the positive offsets, and then write the loop for the negative offset:
```
sum = 0
if offset >= 0:
    # write the code for the positive case
else:
    # write the code for the negative case
```

   ANS:

```
def sum_diag_UL_LR(grid, offset):
    sum = 0
    if offset >= 0:
        i = 0
        # ensure both indexes stay within bounds
        while i < len(grid) and offset < len(grid):
            sum += grid[i][offset]
            i = i + 1
            offset = offset + 1
    else:
        i = -offset
        j = 0

        # ensure both indexes stay within bounds
        while i < len(grid) and j < len(grid):
            sum += grid[i][j]
            i = i+1
            j = j+1

    return sum
```

Once you have written both loops, you may notice that you can use only one loop if you initialize the indices for the row and columns correctly. Here is a combined solution:

```
def sum_diag_UL_LR(grid, offset):
    sum = 0
    if offset > 0:
        i,j = 0,offset
    else:
        i,j = -offset,0
    # ensure both indexes stay within bounds
    while i < len(grid) and j < len(grid):
        sum += grid[i][j]
        i, j = i+1, j+1

    return sum
```

3. Write a function `print_keys(d)` that prints the keys in the dictionary `d`. For example, if the dictionary passed in is

   {"I": 1, "V": 5, "X": 10, "L": 50}

   then the function prints the following:

```
I
V
X
L
```

ANS:

```
def print_keys(d):

    for key in d:
        print(key)
```