CSE 465/565
Spring 2019
Homework #5

Notes:

- *In a PDF file named yourUniqueIDProbs.PDF, include:*
  - *The answers to the textbook questions*
- *In a second PDF file named yourUniqueIDSlides.PDF, include:*
  - *Your presentation slides for the Named Constants issue*
- `access.py` *should contain all of your source code.*
- *Submit a zip file that contains access.py and your two PDFs*

1. **(12 pts; 4 pts each)** Do the following Review Questions from the textbook's Chapter 5:
   a. #9 - What is the l-value of a variable? What is the r-value?
   b. #11 - Define static, stack-dynamic, explicit heap-dynamic, and implicit heap-dynamic variables. What are their advantages and disadvantages?
   c. #21 - What are the advantages and disadvantages of dynamic scoping?

2. **(8 pts; 4 pts each)** Do the following Review Questions from the textbook's Chapter 6:
   a. #41 - Define strongly typed.
   b. #42 - Why is Java not strongly typed?

3. **(36 pts; 9 pts each)** Do the following Problem Set questions from the textbook's Chapter 6:
   a. #9 - Multidimensional arrays can be stored in row major order, as in C++, or in column major order, as in Fortran. Develop the access functions for both of these arrangements for three-dimensional arrays.
   b. #14 (50 words or less) - What are the arguments for and against Java's implicit heap storage recovery, when compared with the explicit heap storage recovery required in C++? Consider real-time systems.
   c. #16 (50 words or less) - What would you expect to be the level of use of pointers in C#? How often will they be used when it is not absolutely necessary?
   d. #17 (50 words or less) - Make two lists of applications of matrices, one for those that require jagged matrices and one for those that require rectangular matrices. Now, argue whether just jagged, just rectangular, or both should be included in a programming language.

4. **(24 pts)** Create 4-6 effective presentation slides that describe how programmers can define "constants" in Java, C++, C#, and Python. Your slides should be targeted to a fellow 465/565 student and should include all major aspects relevant to this topic. As a minimum:
   a. Identify the keywords used to create constants
   b. Give examples of how to create constants and state exactly they prevent the programmer from doing.
   c. Examine the difference contexts in which the keywords can be applied.
   d. Any information that would be useful for a 465/565 student to know.

5. **(20/10 pts)** An `Triangular` matrix is a square matrix whose elements below the diagonal are defined to be 0.

   For example, the matrix element $M_{r,c} = 0$ if $r > c$. The following is an example matrix of size 4.

	0	**1**	**2**	**3**
0	100	200	300	400
1	0	500	600	700
2	0	0	800	900
3	0	0	0	1000

While it is possible to use a regular 2D array to represent an Triangular matrix, doing so is wasteful with memory. Instead, we plan on creating an ADT that stores only the non-zero information as shown below:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
100	200	300	400	500	600	700	800	900	1000
$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	$M_{0,3}$	$M_{1,1}$	$M_{1,2}$	$M_{1,3}$	$M_{2,2}$	$M_{2,3}$	$M_{3,3}$

The basis of an effective ADT for an `Triangular` matrix requires two important computations:

Determine how many non-zero elements are required
Access function.

You are to write the following Python functions in a file named `access.py` that computes the access function for an `Triangular`. All of your functions should be O(1).

```
# Modify these functions. Add helper functions if you wish.
# These functions must be O(1).
def numNonZeros2DTriangular(N):
     return 0
def access2DTriangular(N, row, col):
     # return -1 if any parameters are nonsensical
     return 0
```

Here are some examples of how your code should behave:

```
numNonZeros2DTriangular(4) → 10
numNonZeros2DTriangular(1) → 1
numNonZeros2DTriangular(5) → 15
access2DTriangular(4, 0, 0) → 0
access2DTriangular(4, 3, 3) → 9
access2DTriangular(4, 3, 0) → -1
```

YOU DO NOT HAVE TO IMPLEMENT THE ADT FOR TRIANGULAR MATRICES.

6. **(0/10 pts) 565 students only.** Repeat the previous problem but extend to 3D triangular matrices. The non-zero entries are those $M_{i,j,k}$ $i \leq j \leq k$. The 1D layout of a 3x3x3 `Triangular` matrix is as follows:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	...	[19]
$M_{0,0,0}$	$M_{0,0,1}$	$M_{0,0,2}$	$M_{0,0,3}$	$M_{0,1,1}$	$M_{0,1,2}$	$M_{0,1,3}$	$M_{0,2,2}$	$M_{0,2,3}$	$M_{0,3,3}$	$M_{1,1,1}$	$M_{1,1,2}$	$M_{1,1,3}$	$M_{1,2,2}$	...	$M_{3,3,3}$

```
# Modify these functions in access.py. These functions must be
# O(1).
def numNonZeros3DTriangular(N):
     return 0
def access3DTriangular(N, row, col, depth):
     # return -1 if any parameters are nonsensical
     return 0
```

Here are some examples of how your code should behave:

```
numNonZeros3DTriangular(4) → 20
numNonZeros3DTriangular(1) → 1
numNonZeros3DTriangular(5) → 35
access3DTriangular(4, 0, 0, 0) → 1
access3DTriangular(4, 3, 3, 3) → 19
access3DTriangular(4, 1, 2, 3) → 14
```