CISC 610 Assignment 1

Late Fall 2016 Session
Due Date: November 22, 2016 by 11:55 P.M. EST

1 Introduction

This assignment has two programming problems. The first one involving recursion and the second one use of sorting. A description of these two problems is as follows:

1. Use recursion to implement Pascal triangle using the following function call **pascalTriangle(depth)**
   In this method the input parameter is an int called depth and you are required to return **Pascal's Triangle** with depth number of rows.

   Explanation of Pascal's Triangle: In mathematics, a combination of two numbers, represented as $C(n, i)$ is defined as a ratio of factorials as follows:

   $$C(n, i) = \frac{n!}{i! \times (n-i)!}$$

   These factorial values of $C(n, i)$ can be arranged in the form of a triangle called **Pascal's Triangle** which is shown as below up to a depth of 4 (it can have more levels than just 4):

   C(0,0)
   C(1,0) C(1,1)
   C(2,0) C(2,1) C(2,2)
   C(3,0) C(3,1) C(3,2) C(3,3)

   This triangle has a property that each number is a sum of the two numbers above it except the left and the right edges which are always 1. For e.g., **Pascal's Triangle** of depth 4 for the above combinations can be written as:

   $$\begin{matrix} 1 \\ 1\ 1 \\ 1\ 2\ 1 \\ 1\ 3\ \textcircled{3}\ 1 \end{matrix}$$

   The circled number above is a combination $C(3, 2)$ and equal to the sum of 2 and 1 to the left and right of it in the line above it. The above result is returned by a call, **pascalTriangle(4)**.

   For this assignment use the summation property and not the combination property to write a recursive implementation of Pascal's Triangle.

2. The second problem involves a file that contains all of the 100,000 integers between 1 and 100,000 (inclusive) in some order, with no integer repeated. Your task is to write code to compute the number of inversions in the file given where the i^{th} row of the file indicates the i^{th} entry of an array.

   Because of the large size of this array, you should implement or a divide-and-conquer algorithm provided in the class.

1.1 Grading Criteria

This assignment carries 10 points of your assignment grade. You will be graded on logic, comments, whether the code runs or not and the output. You are expected to do this work on your own. You may use the code provided in the class and modify it to apply on this assignment.

1.2 Submission

You have the option to use 2 extra days to finish this assignment unless you have already used two extensions.

Upload all your source code files (.py, or .java, or .c or .c++) to Moodle. Please add a README file to include any instructions on how to successfully run your code if there are any specific steps.

Note: Please do not upload all the project files, just the source code. Submitting wrong files or in the wrong format will not be accepted nor will any re-submission be allowed for any such mistake.