# Assignment 5: Chapter 7, “Arrays” & Chapter 17, “Recursion” (8%)

This programming project should be completed and submitted by the end of Week 13, and is worth 8% of your final grade. Please refer to the “Assignments Instructions” for details on the marking rubric and submission instructions.

1. Design and implement a class called RandomArray, which has an integer array. The constructor receives the size of the array to be allocated, then populates the array with random numbers from the range 0 through the size of the array. Methods are required that return the minimum value, maximum value, average value, and a String representation of the array values. Document your design with a UML Class diagram. Create a separate driver class that instantiates a RandomArray object and outputs its contents and the minimum, maximum, and average values.

**Testing:** Include the output for several different test runs that shows the array contents with its minimum, maximum, and average values.

1. In Assignment 4, you created a Card class that represents a standard playing card. Use this to design and implement a class called DeckOfCards that stores 52 objects of the Card class. Include methods to shuffle the deck, deal a card, and report the number of cards left in the deck, and a toString to show the contents of the deck. The shuffle methods should assume a full deck. Document your design with a UML Class diagram. Create a separate driver class that first outputs the populated deck to prove it is complete, shuffles the deck, and then deals each card from a shuffled deck, displaying each card as it is dealt.

**Hint:** The constructor for DeckOfCards should have nested for loops for the face values (1 to 13) within the suit values (1 to 4). The shuffle method does not have to simulate how a deck is physically shuffled; you can achieve the same effect by repeatedly swapping pairs of cards chosen at random.

**Testing:** Include two complete runs to demonstrate the random effect of shuffling.

1. Design and implement a recursive program to determine and print up to the Nth line of Pascal’s Triangle, as shown below. Each interior value is the sum of the two values above it.

1  
 1 1  
 1 2 1  
 1 3 3 1  
 1 4 6 4 1  
 1 5 10 10 5 1  
 1 6 15 20 15 6 1  
 1 7 21 35 35 21 7 1  
 1 8 28 56 70 56 28 8 1

**Hint:** You should use an array to hold the values for a given line.

One recursive approach is:

T(n, 0) = T(n, n) = 1

T(n, d) = T(n - 1, d - 1) + T(n - 1, d)

|  |  |
| --- | --- |
| **Assignment Marking Criteria** | **Weighting** |
| **Correctness of solution:** Algorithm is implemented and produces correct results for the stated problem | /4 |
| **Testing:** Submission of test exhibits to indicate the solution works for a range of cases (e.g. minimum and maximum inputs) and handles unexpected exceptions | /2 |
| **Comments and documentation:** Source code contains comments that explain in plain English what the code is intended to do  **Note:** Javadoc style is **not** required. | /2 |