JAVA PROGRAMMING LANGUAGE
Assignment 12

1. There are n people in a room, where n is an integer greater than or equal to 1. Each person shakes hands once with every other person. What is the total number, h(n), of handshakes?
   Write a recursive function to solve this problem.

   To get you started, if there are only one or two people in the room, then
   handshake(1) = 0
   handshake(2) = 1

   If a third person enters the room, he or she must shake hands with each of the two people already there. This is two handshakes in addition to the number of handshakes that would be made in a room of two people, or a total of three handshakes. If a fourth person enters the room, he or she must shake hands with each of the three people already present. This is three handshakes in addition to the number of handshakes that would be made in a room of three people, or six handshakes. If you can generalize this to *n* handshakes, then it should help you write the recursive solution.

2. The game of "Jump It" consists of a board with n positive integers in a row except for the first column, which always contains zero. These numbers represent the cost to enter each column.

   Here is a sample game board where n is 6:

0	3	80	6	59	10

   The object of the game is to move from the first column to the last column in the lowest total cost. The number in each column represents the cost to enter that column. Always start the game in the first column and have two types of moves. You can either move to the adjacent column or jump over the adjacent column to land two columns over. The cost of a game is the sum of the costs of the visited columns.

   In the board shown above, there are several ways to get to the end. Starting in the first column, our cost so far is 0. We could jump to 80, then jump to 57, then move to 10 for a total cost of 80 + 57 + 10 = 147. However, a cheaper path would be to move to 3, jump to 6, then jump to 10, for a total cost of 3 + 6 + 10 = 19.

   Write a recursive solution to this problem that computes the cheapest cost of the game and outputs this value for an arbitrarily large game board represented as an array. Your program does not have to output the actual sequence of jumps, only the cheapest cost of this sequence. After making sure that your solution works on small arrays, test your solution on boards of larger and larger values of n to get a feel for how efficient and scalable your solution is.

3. Write a recursive method named contains with the following header:

   public static boolean contains(String haystack, String needle)

   The method should return true if needle is contained within haystack and false if needle is not in haystack.

For example, *contains("Java programming", "ogr")* should return true contains*("Java programming", "grammy")* should return false You are not allowed to use the substring method to find a match