**ArrayUtility**

1. CreateEmpty
2. CreateRandom
3. CreateMatrix
4. FindMax
5. FindMaxPos
6. FindMin
7. FindMinPos
8. Swap
9. ShiftRight
10. ShiftLeft
11. Copy
12. Find

**Maximal Length of Ascents in Arrays**

Consider an array A[1..n] of integers. The subarray A is an ascent if A[j] ≤ A[j + 1] for all j where k ≤ j < l. In other words, an ascent is a nondecreasing segment of A.

A = [3, 1, 4, 2, 4, 4, 5, 3], the maximal length of an ascent would be 4, because the subarray A[4..7] = [2, 4, 4, 5] is the longest ascent in that array.

**Array of Averages**

Given an array A[1..n] of floating point numbers, it returns a two-dimensional array, say M, of size n × n in which the entry M[i][j] for i ≤ j contains the average of the array entries A[i] through A[j].

That is: if i ≤ j, then

M[i][j] =( A[i] + · · · + A[j])/( j − i + 1) ,

whereas for i > j we have that M[i][j] = 0

**Clump**

Say that a "clump" in an array is a series of 2 or more adjacent elements of the same value. Return the number of clumps in the given array.   
  
countClumps({1, 2, 2, 3, 4, 4}) → 2  
countClumps({1, 1, 2, 1, 1}) → 2  
countClumps({1, 1, 1, 1, 1}) → 1

**Multiple of 10**

For each multiple of 10 in the given array, change all the values following it to be that multiple of 10, until encountering another multiple of 10. So {2, 10, 3, 4, 20, 5} yields {2, 10, 10, 10, 20, 20}.   
  
tenRun({2, 10, 3, 4, 20, 5}) → {2, 10, 10, 10, 20, 20}  
tenRun({10, 1, 20, 2}) → {10, 10, 20, 20}  
tenRun({10, 1, 9, 20}) → {10, 10, 10, 20}

Given a non-empty array, return true if there is a place to split the array so that the sum of the numbers on one side is equal to the sum of the numbers on the other side.   
  
canBalance({1, 1, 1, 2, 1}) → true  
canBalance({2, 1, 1, 2, 1}) → false  
canBalance({10, 10}) → true