**[CSE 1310](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/) -** [**Practice Problems**](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/) **- Practice Set 3**

**Practice Problem 1, variation a**

Write a function a\_after\_star(String s) that counts and returns the number of times that character 'a' occurs AFTER the first occurrence of character '\*'.

The function should return 0 if the string does not contain character '\*' at all.

For example:

* a\_after\_star("ab\*ab\*ab") should return 2.
* a\_after\_star("a cat\*chases after its tail") should return 3.
* a\_after\_star("a cat\*chases after\*its tail") should return 3.

**Practice Problem 1, variation b**

Write a function a\_between\_stars(String s) that counts and returns the number of times that character 'a' occurs AFTER the first occurrence of character '\*' and BEFORE the second occurrence of character '\*'.

The function should return 0 if the string does not contain character '\*' at all. If the string contains character '\*' only once, the function should return the number of times that character 'a' occurs betweeh character '\*' and the end of the string.

For example:

* a\_between\_stars("ab\*ab\*ab") should return 1.
* a\_between\_stars("a cat\*chases after its tail") should return 3.
* a\_between\_stars("a cat\*chases after\*its tail") should return 2.

**Practice Problem 2**

We have files like [towns.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/practice3/towns.txt), whose contents look like this:

cat1chicago9234

dog3dallas01

sheep2boston4392897

donkey6denver4921109

Each line contains:

* The name of an animal (which consists of lower-case letters).
* Then, a single numerical digit.
* Then, the name of a city (which, again, consists of lower-case letters).
* Then, more numerical digits.

Write a function print\_towns(String filename) that satisfies these specs:

* Its argument is the name of a file containing lines of text as described above.
* The function prints, for each line in the file, the city name that appears in that line.

For example, applied to file [towns.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/practice3/towns.txt), print\_towns("towns.txt") prints this:

chicago

dallas

boston

denver

**Practice Problem 3**

Write a function check\_sorted(int[] array) that:

* Returns boolean value true if the array is empty.
* Returns boolean value true if the values of the array are sorted in ascending order.
* Returns false otherwise.

For example:

* If a1 = {20, 10, 20, 30, 30}, then check\_sorted(a1) should return false.
* If a2 = {5, 5, 8, 10, 10, 12}, then check\_sorted(a2) should return true.

**Practice Problem 4**

Write a function check\_squares(int[] array), that satisfies these specs:

* If the array is empty, it returns false.
* If there exists an integer N such that the array contains both N and N2, the function returns boolean value true.
* The function returns false otherwise.

For example:

* If a1 = {1, 1, 1, 2}, then check\_squares(a1) should return true.
* If a2 = {20, 10, 20, 30, 40}}, then check\_squares(a2) should return false.
* If a3 = {20, 100, 2, 10, 20, 20, 20}, then check\_squares(a3) should return true.

**Practice Problem 5**

Write a function count\_equal(String filename, int col1, int col2), that satisfies these specs:

* Argument filename is the name of a CSV file.
* The function counts and returns the number of rows where the value at column col1 is equal (as a string) to the value at column col2.
* The function should be able to handle cases where each line may have a different number of values separated by commas.
* The function should be able to handle cases where some line in the file does not contain any values at column col1 or column col2.

For example, using file [file1.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/practice3/file1.txt):

* count\_equal("file1.txt", 0,1) should return 2.
* count\_equal("file1.txt", 0,5) should return 2.
* count\_equal("file1.txt", 1,4) should return 1.
* count\_equal("file1.txt", 1,3) should return 0.

**Practice Problem 6**

Write a function largest\_average(String filename), that satisfies these specs:

* Argument filename is the name of a CSV file.
* The function computes, for each column, the average of all numbers appearing in that column, and returns the largest of all averages. To compute the average for a column, the function considers all values in the column that can be converted to double numbers, and computes the average of those values. If a column does not contain any numbers, the average of that column is considered to be 0.
* The function should be able to handle cases where some values cannot be converted to numbers. Those values should just be skipped.
* The function can assume that all rows in the CSV file have the same number of columns, there is no need to be able to handle files that violate this assumption.

For example, using file [file2.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/practice3/file2.txt):

* largest\_average("file2.txt") should return 1032.25, which is the average of all numbers found in column 2 (the second-to-last column in the file). That column contains numbers 15, 2000, 2014, 100, and their average is 1032.25.

**Practice Problem 7**

Write a function max\_length\_position(String filename) that satisfies these specs:

* Argument filename is the name of a CSV file.
* The function should handle the case where different rows may contain different number of values. The function returns an array containing two integers: the row and the column of the value with the largest length in the file.
* The function should follow the convention that rows and columns are numbered starting at 0.

For example, using file [file2.txt](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/practice3/file2.txt):

* max\_length\_position("file2.txt") should return array {4, 2}, because the string at row 4 and column 2 of that file is "Wednesday", which is the longest string stored in that file.

**Practice Problem 8**

Write a function print\_combinations(String[] array1, String[] array2) that prints all pairs combining a value of array1 with a value of array2. For example, if a1 = {"red", "white", "green", "blue"} and a2 = {"shirt", "pants", "shoes"}, then print\_combinations(a1, a2) should print:

red shirt

red pants

red shoes

white shirt

white pants

white shoes

green shirt

green pants

green shoes

blue shirt

blue pants

blue shoes

**Practice Problem 9**

Write a function multiplication\_table(int k) that satisfies these specs:

* If k is less than 0, the function returns null.
* The function returns a 2-dimensional array of size (k+1)x(k+1), that contains the multiplication table of integers from 0 to k. The result, at position [i][j] contains the value i\*j.

For example:

* multiplication\_table(3) should return array {{0, 0, 0, 0}, {0, 1, 2, 3}, {0, 2, 4, 6}, {0, 3, 6, 9}}

**Practice Problem 10**

Write a function subarray\_sums(int[] array) that returns a 2-dimensional array of integers. The result should have a number of rows and a number of columns that are equal to the length of the input array. At position [i][j], the result should contain the sum of values array[i] + array[i+1] + ... + array[j]. If j < i, result[i][j] should be 0.

For example, if a1 = {7, 3, 8, 1, 2}, then subarray\_sums(a1)should return array

{{7, 10, 18, 19, 21},

{0, 3, 11, 12, 14},

{0, 0, 8, 9, 11},

{0, 0, 0, 1, 3},

{0, 0, 0, 0, 2}}