**[CSE 1310](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/) -** [**Practice Problems**](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/practice/) **- Practice Set 2**

**Practice Problem 1**

Write a function foo(int A, int B) that satisfies these specs:

* If A is greater than B, foo(A, B) returns -1.
* If A is less than or equal to B, foo(A, B) returns the sum of all integers N such that A <= N <= B.

For example:

* foo(5, 9) should return 35, since 5+6+7+8+9 = 35.
* foo(5, 3) should return -1.
* foo(5, 5) should return 5.

**Practice Problem 2**

Write a function print\_special(String S) that satisfies these specs:

* It prints, on the first line, the letter at position 0 of S.
* It prints, on the second line, the letter at position 1 of S, repeated twice.
* It prints, on the third line, the letter at position 2 of S, repeated three times.
* And so on, printing on the N-th line the letter at position N-1 of S, repeated N times, until all letters of S have been printed.

For example, print\_special("Texas") prints this:

T

ee

xxx

aaaa

sssss

**Practice Problem 3**

Write a function print\_special\_to\_file(String S, String output\_file) that creates a new file, whose name is specified by argument output\_file, such that the contents of the file are the same as the output shown on the screen in the previous problem.

More specifically, the contents of the output file should satisfy these specs:

* The first line contains the letter at position 0 of S.
* The second line contains the letter at position 1 of S, repeated twice.
* The third line contains the letter at position 2 of S, repeated three times.
* And so on, the N-th line contains the letter at position N-1 of S, repeated N times, until all letters of S have been printed.

For example, print\_special\_to\_file("Texas", "target.txt") creates a file called target.txt, whose contents are these:

T

ee

xxx

aaaa

sssss

**Practice Problem 4**

Write a function check\_letters(String A, String B), that satisfies these specs:

* If every character of B appears at least once in A, the function should return Boolean value true.
* If there exists a letter in B that does not appear at all in A, the function should return Boolean value false.
* Letter comparisons should be case sensitive.

For example:

* check\_letters("this is a test", "asteh") returns true.
* check\_letters("THIS IS a test", "asteh") returns false (because letter 'h', in lower case, appears in the second argument but does NOT appear in the first argument).

**Practice Problem 5**

Write a function check\_ab(String S), that satisfies these specs:

* If S contains an equal number of A's and B's the function returns true.
* Otherwise it returns false.

In other words, the function counts how many times letter 'a' appears in S, (counting both upper and lower cases), and how many times letter 'b' appears in S (counting again both upper and lower cases), and checks whether 'a' appears as many times as 'b'. For example:

* check\_ab("hello") returns true, because there are 0 A's and 0 B's.
* check\_ab("cat") returns false, because there is 1 A and 0 B's.
* check\_ab("Barbados") returns true, because there are 2 A's and 2 B's.
* check\_ab("ALL BREAD and butter") returns false, because there are 3 A's and 2 B's.

**Practice Problem 6**

Write a function check\_ab\_in\_file(String filename), that satisfies these specs:

* Argument filename specifies the name of an existing text file.
* If that file contains an equal number of A's and B's the function returns true.
* Otherwise it returns false.

In other words, the function counts how many times letter 'a' appears in the text file, (counting both upper and lower cases), and how many times letter 'b' appears in the text file (counting again both upper and lower cases), and checks whether 'a' appears as many times as 'b'.

**Practice Problem 7**

Write a function find\_N(int number) that satisfies these specs:

* If there exists an integer N >= 0 such that number = 2N + 3N, the function should return that integer N.
* Otherwise, the function should return -1.

For example:

* find\_N(2) returns 0, because 2 = 20 + 30.
* find\_N(4) returns -1.
* find\_N(5) returns 1, because 5 = 21 + 31.
* find\_N(10) returns -1.
* find\_N(13) returns 2, because 13 = 22 + 32.

**Practice Problem 8**

Write a function subarray\_sums(int[][] items) that satisfies these specs:

* items is an array of arrays of ints. Thus, each element of items is an array of ints.
* The function returns an array of ints called result, of the same size as items. In other words, the number of elements of result is the same as the number of arrays of ints contained in items.
* The value stored at result[i] should be the sum of all values of the array stored in items[i].

For example:

* subarray\_sums({{10, 5, 3}, {3, 1}} should return array {18, 4}, because the sum of all values in array {10, 5, 3} is 18 and the sum of all values in array {3, 1} is 4.

**Practice Problem 9**

Write a function pick\_elements(int[] items, int[] positions) that satisfies these specs:

* The function returns an array of ints called result, of the same size as positions.
* The value stored at result[i] should be the value stored in array items, at the position specified by positions[i]

For example:

* pick\_elements({10, 5, 3, 20, 30}, {3, 1}) should return array {20, 5}. The result contains the values at positions 3 and 1 of array items.
* pick\_elements({10, 5, 3, 20, 30}, {0, 4, 1}) should return array {10, 30, 5}. The result contains the values at positions 0, 4, and 1 of array items.

**Practice Problem 10**

Write a function insert(int[] items, int position, int value) that satisfies these specs:

* The function returns an array of ints called result, whose size is 1 + the size of items.
* For any i such that 0 <= i < position, result[i] = items[i].
* result[position] = value.
* For any i such that position < i < result.size(), result[i] = items[i-1].

For example:

* insert({10, 5, 3, 20, 30}, 0, 11) should return array {11, 10, 5, 3, 20, 30}. The result is the same as the items argument, except that we have inserted value 11 at position 0.
* insert({10, 5, 3, 20, 30}, 4, 11) should return array {10, 5, 3, 20, 11, 30}. The result is the same as the items argument, except that we have inserted value 11 at position 4.