Exercise 7: Algorithms

1. Search I (20 points)

We are given an array of N positive integers and a single integer x. We want to determine if the array contains two numbers whose sum equals x. For example, if the array is 2, 5, 1, 17, 8, 6, 4 and x = 11, the answer is yes since 5 + 6 = 11. Note that the same number in the array can be used twice. For example, if x = 16 the answer is also yes since 8 + 8 = 16.

a. We seek an algorithm that solves this problem. The algorithm's run-time should be order of N^2. Design and write (in pseudo-code) one such algorithm.

b. Implement your algorithm in Java. Write a `main` method with test data illustrating that your program works correctly. Call this class `Search1b`.

c. Same problem as before, but now the array is sorted. We seek an algorithm that solves this version of the problem. The algorithm's run-time should be order of $N \log_2 N$. Design and write (in pseudo-code) one such algorithm.

d. Implement your algorithm in Java. Write a `main` method with test data illustrating that your program works correctly. Call this class `Search1d`.

2. Search II (10 points)

The binary search algorithm was discussed in lecture 8.1. Write a static method that implements this algorithm on a sorted array of Strings. The input is a sorted array of strings and a single string. If the string is in the array, the method returns its position (index). Otherwise, the program returns -1. Call this method `search` and put it in a class called `StringArrayUtils`. Write a `main` method with test data illustrating that your program works correctly.

3. Merge Sort (10 points)

The merge sort algorithm was discussed in lecture 8.2. Write a static method that implements the merge sort algorithm on an array of Strings. The input is an array of strings. The output is the same array, sorted in increasing alphabetical order. Call this method `sort` and put it in the class `StringArrayUtils` defined before. Write a main method with sample data illustrating that your program works correctly.

4. Word Puzzles (60 points)

Consider the following word puzzle (תשבץ) problem. The input consists of a two-dimensional array of letters and a list of words. Each word in the list appears in the table either in a horizontal orientation, or in a vertical orientation, or in a diagonal orientation going from top-left to bottom-right. The goal is to find where each one of these words appears in the puzzle.

For each word in the list, the program prints the table coordinates of the word and the word itself. For example, suppose that the input is as follows:

	A	B	C	D	E	F	G
1	b	g	o	m	l	e	t
2	a	d	y	v	o	u	a
3	j	a	v	a	e	r	l
4	o	v	e	i	f	r	e
5	k	e	y	y	n	u	b
6	e	s	t	u	d	e	a

Word list: java, vine, oyvey, omlet, tale, joke, key, more, over, if, go, let, verb, in

Given this input, the algorithm prints the following output:

3A - 3D : java
3C – 6F : vine
1C – 5C: oyvey
Etc. (each word is listed once)

Formally speaking, the table is of size N rows by M columns, the number of words in the list is W (N, M and W are positive), and each word is a sequence of 2 or more letters that does not exceed the table's boundaries. For example, "xfguz" may well be a valid word, although it means nothing in any language (to the best of Google's knowledge).

a. Design an efficient algorithm that solves a given word puzzle.

b. Determine and justify an upper-bound on your algorithm's worst-case run-time. Your answer should be something like "the run-time of my algorithm is at most an order of … (some function of N, M, W)"

c. Write a Java program (`WordPuzzle`) that implements your algorithm. For simplicity of testing, assume that the input is exactly like that described in the above example. In other words, although your algorithm must work for any input, the implementation that we ask you to write and submit should demonstrate this ability by working on the input shown in the example. Thus, instead of reading the puzzle input from a file, as we will normally do, define the input using the following statements:

```
char[][] puzzle = {
      {b, g, o, m, l, e, t},
      {a, d, y, v, o, u, a},
      etc.
};

String[] words = {"java", "vine", "oyvey", "omlet", etc.}
```

Your implementation can use the services of the `StringArrayUtils` class.

What to submit

A zip file containing the 5 Java source files (`.java` files only) and a word / text document with answers to questions 1a, 1c, 4a, 4b. Follow all the regular submission instructions.