First Factory Coding Challenge

Instructions

Please read these instructions carefully, your solution will be invalid if it does not follow them.

1. This is a timed challenge. You have three hours from now to email back your solution. Please take all of that time, there is no bonus for completing early, so spend any extra time you may have checking your work, documenting it and clarifying it. If you cannot finish in time, submit the work that you have at the two hour mark and send a follow up email explaining what remains.
2. Answer as much as you can, but work in the order that the problems are presented.
3. We will accept solutions in .NET, Java, Scala, Python, and JavaScript using only standard libraries.
4. If you have any questions about the problem set, please use your best judgement and document your assumptions appropriately.
5. It is important to imagine that your solution will need to be employed in a production environment in the company of other developers. It should be correct but also efficient, well structured, well designed and clearly written such that others who are unfamiliar with your work will be able to easily read and extend it in the future.
6. If you have time, please document all the methods and classes using a standard documenting tool.
7. If you have even more time, please write an ad hoc test harness and accompanying unit tests for your code.
8. To prove your solutions, please provide screenshots or video of it running on your computer.
9. When finished, please reply to this email with your solution attached in a ZIP file. Only include source code, screenshots and videos of the working solution (no binaries or executables please) in your archive.

Thank you and good luck!

Problem 1

Description

"A quick brown fox jumps over the lazy dog."

This is a *pangram*: it contains every letter in the English alphabet.

Please write a method **`listMissingLetters`** that takes a String s as input and returns all the letters missing from it (i.e. those that prevent it from being a pangram) concatenated in one String.

Your method should be case-insensitive, ignore anything but alphanumeric US-ASCII characters, and provide its output as lower-case letters in alphabetical order.

Your method signature should conform to your chosen language's variant of:

```
public String listMissingLetters(String s)
```

Examples

1. "A quick brown fox jumps over the lazy dog"
Returns: ""

2. "Four score and seven years ago."
Returns: "bhijklmpqtwxz"

3. "To be or not to be, that is the question!"
Returns: "cdfgjklmpvwxyz"

4. ""
Returns "abcdefghijklmnopqrstuvwxyz"

Problem 2

Description

Some number of bombs sit in a linear chamber, at time 0. At time 1, the bombs explode. Each sends one piece of shrapnel to the left and one piece of shrapnel to the right. At each successive time iteration, the pieces move at a constant speed on their due courses, passing through each other unimpeded, until all have left the chamber.

You are to create a NetHack-esque animation of this process. Given a String describing the initial locations of the bombs, and an Integer describing the bombs' concussive power, you are to compute the locations of the pieces of shrapnel at each time iteration, terminating once all have left the chamber.

Your method will take as input a String **`bombs`** and an Integer **`force`**.`bombs` will have a "B" at each position containing a bomb, and a "." at each empty position. At time 1, the B's disappear, sending one "<" piece of shrapnel to the left, and one ">" piece of shrapnel to the right, each moving at the constant speed **`force`**. If a "<" and ">" ever occupy the same position at the same time, they will be collectively represented by a single "X".

Your method will return an array of Strings in which each successive element shows the occupied locations at each time unit. The first element should show the initial locations of the bombs, using "B" and "." characters. The last element should show the empty chamber at the first time that it becomes empty.

(Please see the examples on the following page if this is unclear — it helps to see it in motion.)

Your method signature should conform to your chosen language's variant of:

```
public String[] explode(String bombs, int force)
```

`explode` should accept its input under the following constraints:

- **`bombs`** should contain between 1 and 50 characters, inclusive.
- Each character in **`bombs`** should be either a "." or a "B".
- **`force`** should be between 1 and 10, inclusive.

Examples

1. `explode("..B....", 2)`
Returns:
```
["..B....",
 "<...>..",
 ".....>",
 "......."]
```

2. `explode("..B.B..B", 10)`
Returns:
```
["..B.B..B",
 "........"]
```

3. `explode("B.B.B.BB.", 2)`
Returns:
```
["B.B.B.BB.",
 "<.X.X<>.>",
 "<.<<>.>.>",
 "<<....>.>",
 "........>",
 "........."]
```

4. `explode("..B.B..B", 1)`
Returns:
```
["..B.B..B",
 ".<.X.><.",
 "<.<.><>.",
 ".<..<>.>",
 "<..<..>.",
 "..<....>",
 ".<......",
 "<.......",
 "........"]
```

5. `explode("..B.BB..B.B..B...", 1)`
Returns:
```
["..B.BB..B.B..B...",
 ".<.X<>><.X.><.>..",
 "<.<<>.X><.><>..>.",
 ".<<..X.X>.<>.>..>",
 "<<..<.X.>X..>.>..",
 "<..<.<.><>>..>.>.",
 "..<.<..<>.>>..>.>",
 ".<.<..<..>.>>..>.",
 "<.<..<....>.>>..>",
 ".<..<......>.>>..",
 "<..<........>.>>.",
 "..<.........>.>>",
 ".<..........>.>",
 "<............>.",
 "..............>",
 "..............."]
```