## Exercise 1. Stacking Boxes (adapted)

Boxes are placed at integer spaces along number line from 0 to 1000, possibly stacked atop each other.

A prime spacing of boxes is such that for every box $A$, every other box $B$ is either in the same stack as $A$ or a prime distance from $A$. What is the minimum cost of moving the boxes from their initial positions to a prime spacing? Moving a box from position $x$ to position $y$ costs $abs(x - y)$ and there up to 1,000,000 boxes.

| Sample input | Sample output |
|---|---|
| `2 1 0 2 1 3` | `2` |

## Exercise 2. Cyberline - TopCoder SRM 187: Division I, Level One

As managing editor of BadCyberPoetry.net you find to your dismay that you must follow through on your threat to replace your "Editor of Conventionally Structured Poetry" with a very small shell script (or something to that effect).

The first part of this project is to build a method that returns the last cyberword of a cyberline of cybertext, in a form that is convenient for rhyme testing. Since this is bad cyber poetry, words may contain symbols other than letters.

A "cyberword" may contain letters, numbers, the hyphen (minus) character `-` and the `@` character. Cyberwords must contain at least one character that is a letter, number or `@`. Isolated hyphens or strings of hyphens alone are not cyberwords. Any other character is considered punctuation or white space and causes a cyberword break.

Cyberlines may contain cyberwords, punctuation, and spaces in any order, so the last cyberword may not be at the end of the cyberline. The cyberword you return should have all hyphens removed to simplify rhyme testing.

| Sample input | Sample output |
|---|---|
| `Zowie:  This is a line of##cyber-poetry## !` | `cyberpoetry` |
| `2-b\|!2-b -- ?` | `2b` |
| `if then; elseif schmelshif();` | `schmelshif` |
| `-y?` | `y` |
| `@@@@@@@@@@@@@@@@@@@@@@@@@` | `@@@@@@@@@@@@@@@@@@@@@@@@@` |

## Exercise 3. UnLinker - TopCoder SRM 203: Division II, Level Three

You are implementing the portion of an online dating site where members display their profiles. Most of the profile content is automatically generated from member data. Part of a profile, however, is furnished directly by the member as free-form text. Weblinks frequently crop up in this text, despite a site policy that forbids advertisement and linking of any kind. Your job is to seek and destroy all weblinks in a given piece of text.

For the purposes of this problem, a weblink is a string consisting of three parts. From left to right, these are the prefix, domain, and suffix.

The prefix consists of either `http://`, `http://www.`, or `www.`. The suffix is one of `.com`, `.org`, `.edu`, `.info`, or `.tv`.

The domain is a sequence of one or more characters, each of which is a letter (a character from `a` to `z` or from `A` to `Z`), a numeral (`0` to `9`), or a period (the character `.`).

There must be no space character within the weblink. The weblink may have any kind of character to the left and right of it. It may also occur at the beginning of the text, at the end of the text, or it may itself be the entire text. Furthermore, a weblink extends as far as possible to the left and right.

Each weblink in the text must be replaced with a string consisting of the word "OMIT" followed by a number. The first weblink is to be replaced with `OMIT1`, the second with `OMIT2`, and so forth. All portions of text that do not consist of a weblink must remain intact.

| Sample input | Sample output |
|---|---|
| `espihttp://www.tv.org.superwww.cali.comaladocious` | `espiOMIT1aladocious` |
| `check www.foo.com 4 www.foo.com www.scoopz.com` | `check OMIT1 4 OMIT2 OMIT3` |
| `check www.foo.com 4 www.foo.comwww.scoopz.com` | `check OMIT1 4 OMIT2` |
| `check www.foo.com 4 www.foo.comhttp://scoopz.com` | `check OMIT1 4 OMIT2OMIT3` |
| `http://411.com goodz 4 www.733t.com, 2http://..com` | `OMIT1 goodz 4 OMIT2, 2OMIT3` |
| `http.//say.org,www.jeeves.x.info,www.comhttp://.tv` | `http.//say.org,OMIT1,www.comhttp://` |
| `http://www.www.com/www` | `OMIT1/www` |

### Exercise 4. SWERC 2009 - Stammering Aliens (adapted)

Given an integer $m$ and a string $s$, find the longest *substring* $t$ of $s$ that appears at least $m$ times. Output the length of $t$ and its rightmost start index in $s$.

For example, in the string `baaaababababbababbab`, the length-5 word `babab` is contained 3 times, namely at positions 5, 7 and 12 (where indices start at zero). No substring appearing 3 or more times is longer (see the first example from the sample input). On the other hand, no substring appears 11 times or more (see example 2).

| Sample input | Sample output |
|---|---|
| `3 baaaababababbababbab` | `5 12` |
| `11 baaaababababbababbab` | `none` |
| `3 cccccc` | `4 2` |