Corrections script

This script takes an input file and a comma separated list of corrections which it applies to the file, producing a separate output file for safety.

Usage

```
correct -i <input file> -c <corrections file>
```

```
correct -h
```

Example with the supplied test files:

```
correct -i corr_test.tex -c corrections.txt
```

Input file

The input file is any text file (typically your .tex source) A sample input file is given in `corr_test.tex`.

Corrections file

The corrections file contains pairs of words or regular expressions separated by a comma. The first element is the search regex and the second is the replace regex.

A sample corrections file is given in `corrections.txt`.

Since global regex replacement can be quite tricky, it's best to test carefully!

Some comments on sed regular expression escaping

The regular expression matching engine of `sed` on the Mac is a subset of the regex expressions you may be used to using TeXShop or most other editors. The most notable difference (and in fact deficiency) is the lack of a non-greedy `*` operator (usually `*?`). For the cases that contain a delimiter, which is often the case if e.g. you're looking for something that is enclosed in `{...}` or `[...]` there is a simple workaround (see below). First, here's a table of the kinds of things that need to be escaped in the search string along with whether they need to be escaped in the replacement string.

Remember that an expression enclosed in `[...]` matches those characters or sets of characters and characters inside the `[...]` typically don't themselves need to be escaped.

Table 1: Characters that need to be escaped for searching

Character	Inside search string	Inside replacement string
`{` and `}`	`\{` and `\}`	`{` and `}`
`[` and `]`	`\[` and `\]`	`[` and `]`
`(` and `)`	`\(` and `\)`	`(` and `)`
`\`	`\\`	`\\`
`^`	`\^`	`^`
`.`	`\.`	`.`

Table 2: Characters that require `\` to acquire a special meaning

Character	inside search string	inside replacement string
`\1 \2` ..	NA	search group 1, 2 etc.
`\+`	match 1 or more	NA

Non-greedy `*`

As mentioned above, `sed` doesn't support non-greedy `*` but at least for delimited expressions, it can be easily emulated. For example, if we want to replace `foo*` inside the argument of a TeX macro, we can use the following regex:

```
\{foo.*[^}]\}
```

This says "find all strings prefixed with `foo` (= `foo.*`) delimited by everything that isn't `}` (= `[^}]`)"