`kate.lit.m` — KaTify text

J. Cupitt

July 21st, 1989

There is a group on USENET called `rec.music.gaffa`, dedicated to the singer Kate Bush. A running joke in this group is to pretend fanatical devotion to Kate And Her Works — this reaches an extreme form in some posters, who capitalise all the Ks and Ts in their articles in reference both to KaTe herself and to the Knights Templar. This Miranda[1] script can be used as a UNIX[2] filter to prepare your articles for posting to `gaffa`. The main function is called `kate` and is at the end.

Do some simple maps on text. We do:

c,C,k	→	K
t	→	T
qu,Qu	→	Kw
ck	→	K
ch,Ch	→	Khe

We also look for Kommon words that can be easily swapped for something with more ks and ts.

The dictionary we use to look for common words. This is very small at the moment! I should perhaps find a thesaurus and fix this up properly.

```
> kateMap
>     = [(["interpose", "insert"],
>               "interject"),
>         (["frequent", "general", "usual", "normal"],
>               "common"),
>         (["program", "file"],
>               "script"),
>         (["name"],
>               "appelation"),
>         (["however"],
```

[1]Miranda is a trademark of Research Software Ltd.
[2]UNIX is a trademark of AT&T in the USA and other countries.

```
>                 "though"),
>          (["serve"],
>                 "officiate"),
>          (["intersperse"],
>                 "punctuate")
>         ]
```

First map. Very easy!

```
> swapCase :: [char] -> [char]
> swapCase ('c':'k':x) = 'K':swapCase x
> swapCase ('c':'h':x) = 'K':'h':'e':swapCase x
> swapCase ('C':'h':x) = 'K':'h':'e':swapCase x
> swapCase ('c':x) = 'K':swapCase x
> swapCase ('C':x) = 'K':swapCase x
> swapCase ('k':x) = 'K':swapCase x
> swapCase ('t':x) = 'T':swapCase x
> swapCase ('q':'u':x) = 'K':'w':swapCase x
> swapCase ('Q':'u':x) = 'K':'w':swapCase x
> swapCase (a:x) = a: swapCase x
> swapCase [] = []
```

Second map. We loop down the input again, chopping out words. Each one gets put through tryMap.

```
> swapWords :: [char] -> [char]
> swapWords [] = []
> swapWords inp
>       = punk ++ tryMap word ++ swapWords tail
>         where
>         punk = takewhile ((~) . letter) inp
>         start = dropwhile ((~) . letter) inp
>         word = takewhile letter start
>         tail = dropwhile letter start
```

Try to map a word through the KaTe thesaurus we defined earlier. We try to be clever about what we swap. For example, we want `insert` to be mapped to `interject`, and `inserting` to be mapped to `interjecting`. This isn't always the most sensible way to do it . . .

```
> tryMap :: [char] -> [char]
> tryMap word
>      = word, if maps = []
>      = hd maps, otherwise
>        where
>        maps = [ to ++ drop (#x) word |
>                         (from, to) <- kateMap; x <- from;
>                         x $isprefix word ]
```

Test for first argument a prefix of the second argument.

```
> isprefix :: [*] -> [*] -> bool
> isprefix a b = take (#a) b = a
```

And our entry point. We just pipe stuff first through swapWords, then through swapCase.

```
> kate :: [char] -> [char]
> kate = swapCase . swapWords
```