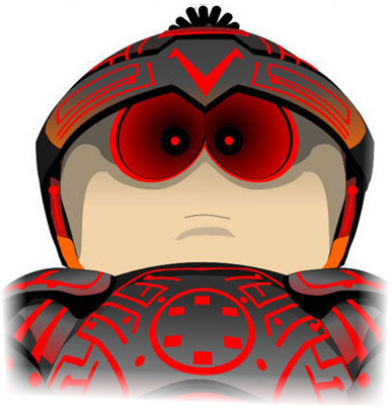

B4- Synthesis Pool

B-ADM-343

AutoCompletion

Autocompletion Engine for GPS Devices

v1.5

AutoCompletion

Autocompletion Engine for GPS Devices

binary name:	autoCompletion
repository name:	autoCompletion_$YEAR
repository rights:	ramassage-tek
language:	C, C++, perl 5, python 3 (≥ 3.5), ruby 2 (≥ 2.2), php 5.6, bash 4
group size:	1
compilation:	via Makefile, including re, clean and fclean rules

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Ever since uncle Edmond's tragic car accident while he was adjusting his GPS device, the FindYourWay.com company has decided to review its address entry process.
Ever-alert to IT firms' news, you decide to propose a POC of a French address autocompletion algorithm.

Here is the algorithm:

1. the address is formatted the following way: *city, number streetType streetName*.
   streetType can take one of the following values:
   - *allée*,
   - *avenue*,
   - *boulevard*,
   - *chemin*,
   - *impasse*,
   - *place*,
   - *quai*,
   - *rue*,
   - *square*.
2. the autocompletion engine first looks for the city, then the street name,
3. the engine works on any part of the city's or street's name (for instance *La Rochelle* contains 2 parts),
4. for each letter, the engine proposes the most probable choices (5 maximum), in descending order,
5. if there is only one possibility for a letter, this one is automatically validated and the engine looks for the next one,
6. once a word is completed,
   (a) if the city is still unknwon:
      i. if only one city contains the completed word, this city is validated and the engine starts looking for the street name,
      ii. otherwise, the list of all the cities containing this word is proposed in alphabetic order. All of the cities are associated with a number, and the user must enter the number of the correct city ; if a letter is entered instead, the propositions are reset.
   (b) if the city is already known:
      i. if only one street name contains the completed word, and only one address matches this city and

street name, this address is validated and the program stops,

ii. otherwise, all theses addresses matching the city and a street names, and containing the completed word, are proposed in alphanumeric order, each associated with a number. The user must enter the correct adress' number to validate it and stop the program ; if a letter is entered instead, the propositions are reset.

In case of equiprobable letters, they will be proposed in alphanumeric order.

Your program must read the address letter-by-letter on the standard input and then write the propositions made by the autocompletion engine on the standard output. It takes a dictionnary as unique argument.
If ever the keyword *ABORT* is read, the program stops.

Terminal

```
~/B-ADM-343> ./autoCompletion -h
USAGE
     ./autoCompletion dictionnary

DESCRIPTION
     dictionnary   file, containing one address per line, serving as knowledge base
```

In the dictionnary, upper and lower cases must be ignored, in addition to dashes and apostrophes. Badly-formatted addresses must be ignored as well, and printed on the error output.

The only two error messages to be printed are *Invalid argument* and *Unknown address*.

Once this first part is functionnal, implement an error-correction algorithm that rectifies badly-formattted addresses (the examples below do not show this error correction).

Obviously, only non-ambiguous addresses containing all of the needed information can be reformatted.

A good parser seems to be necessary... especially as the associated grammar is dead simple!

Your program output has to be strictly identical to the one below.

The validated letters are printed in uppercase.
The one proposed in lowercase.

Terminal

```
~/B-ADM-343> cat exampleDict
Paris, 458 boulevard Saint-Germain
Paris, 343 boulevard Saint-Germain
Marseille, 343 boulevard Camille Flammarion
Marseille, 29 rue Camille Desmoulins
Marseille, 1 chemin des Aubagnens
Paris, 12 rue des singes
Paris, 34 quai VoLtAiRe
Paris, 34 rue Voltaire
Lille, 120 boulevard Victor Hugo
Marseille, 50 rue Voltaire
Toulouse, 90 rue Voltaire
Strasbourg 84 rue du Bouclier
Marseille, 78 boulevard de la libération
Lille, 30 rue Victor Danel
Mont Saint Martin, 42 rue de Lyon
Mont de Marsan, 100 avenue Pierre de Coubertin
Strasbourg, 391 boulevard de Nancy
Lyon, 56 rue du Docteur Albéric Pont
Lyon, rue du Docteur Albéric Pont
56 rue du Docteur Albéric Pont, Lyon
Lyon 56 grande rue
Lille, 90 rue d'Arras
Lille, 76 impasse Georges Pompidou
Lyon, 2 allée des fleurs
```

Terminal

```
~/B-ADM-343> echo x | ./autoCompletion exampleDict 1>/dev/null
Strasbourg 84 rue du Bouclier
Lyon, rue du Docteur Albéric Pont
56 rue du Docteur Albéric Pont, Lyon
Lyon 56 grande rue
Unknown address
```

Terminal

```
~/B-ADM-343> cat test1
L
ABORT
~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test1
{m} {l} {p} {s} {d}
{Li} {Ly}
```

```
Terminal
~/B-ADM-343> cat test2
l
i
v
2

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test2
{m} {l} {p} {s} {d}
{Li} {Ly}
{LILLE, d} {LILLE, v} {LILLE, g} {LILLE, h} {LILLE, p}
{1 :  LILLE, 30 rue VICTOR danel} {2 :  LILLE, 120 boulevard VICTOR hugo}
=> Lille, 120 boulevard Victor Hugo
```

```
Terminal
~/B-ADM-343> cat test3
p
S
a
2

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test3
{m} {l} {p} {s} {d}
{PARIS, s} {PARIS, v} {PARIS, d}
{PARIS, Sa} {PARIS, Si}
{1 :  PARIS, 343 BOULEVARD SAINT-GERMAIN} {2 :  PARIS, 458 BOULEVARD SAINT-GERMAIN}
=> Paris, 458 boulevard Saint-Germain
```

```
Terminal
~/B-ADM-343> cat test4
D

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test4
{m} {l} {p} {s} {d}
=> Mont de Marsan, 100 avenue Pierre de Coubertin
```

```
Terminal
~/B-ADM-343> cat test5
p
v
1

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test5
{m} {l} {p} {s} {d}
{PARIS, s} {PARIS, v} {PARIS, d}
{1 :  PARIS, 34 quai VOLTAIRE} {2 :  PARIS, 34 rue VOLTAIRE}
=> Paris, 34 quai VoLtAiRe
```

```
Terminal
~/B-ADM-343> cat test6
l
i
P

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test6
{m} {l} {p} {s} {d}
{Li} {Ly}
{LILLE, d} {LILLE, v} {LILLE, g} {LILLE, h} {LILLE, p}
=> Lille, 76 impasse Georges Pompidou
```

```
Terminal
~/B-ADM-343> cat test7
l
i
d
R

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test7
{m} {l} {p} {s} {d}
{Li} {Ly}
{LILLE, d} {LILLE, v} {LILLE, g} {LILLE, h} {LILLE, p}
{LILLE, DAn} {LILLE, DAr}
=> Lille, 90 rue d'Arras
```

```
Terminal
~/B-ADM-343> cat test8
m
A
s
e
d
S
M

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test8
{m} {l} {p} {s} {d}
{Ma} {Mo}
{MARs} {MARt}
{MARSe} {MARSa}
{MARSEILLE, d} {MARSEILLE, c} {MARSEILLE, l} {MARSEILLE, a} {MARSEILLE, f}
{1 :  MARSEILLE, 78 boulevard DE la libération}
{1 :  MARSEILLE, 1 chemin DES aubagnens}
=> Marseille, 29 rue Camille Desmoulins
```

```
Terminal
~/B-ADM-343> cat test9
m
0
1

~/B-ADM-343> ./autoCompletion exampleDict 2>/dev/null < test9
{m} {l} {p} {s} {d}
{Ma} {Mo}
{1 :  MONT de marsan} {2 :  MONT saint martin}
=> Mont de Marsan, 100 avenue Pierre de Coubertin
```