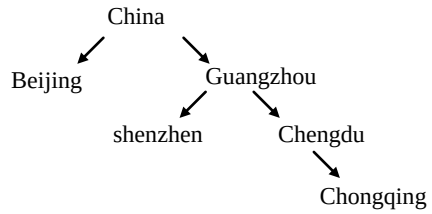
Level 4: The Compiler Generator Coco/R

This task requires you to use the compiler generator Coco/R for building programs that process structured input. It consists of two subtasks of which you have to implement at **least one**.

Task 1: Reading and Building a Binary Tree

Binary trees are dynamic data structures consisting of nodes, where every node has at most 2 sons, which are again binary trees. Assume that we want to build the following binary tree:



Data type for nodes

```
class Node {
    String name;
    Node left;
    Node right;
}
```

We want to read the tree from an input file, which represents the tree structure with parentheses, i.e.:

```
(China
    (Beijing)
    (Guangzhou
        (shenzhen)
        (Chengdu
            ()
            (Chongqing)
        )
    )
)
```

Describe the **input** of such trees by a **recursive EBNF grammar. Write a Coco/R compiler description using this grammar.** Terminal symbols are identifiers as well as '**(**' and '**)**'**.** Add **attributes** and **semantic actions** to your compiler description in order to build the corresponding **binary tree**. Write also a dump method that prints the tree after it was built.

In order to use Coco/R and download the files *Co-co.jar*, *Scanner.frame* and *Parser.frame* into a new directory ***Tree***. If your compiler description is in a file ***Tree.atg*** in the directory *Tree* go to this directory and type

```
java -jar Coco.jar Tree.atg
```

This will generate the files *Scanner.java* and *Parser.java* in the directory *Tree*. Write a main program ***TreeBuilder.java*** that creates a scanner and a parser and calls the parser.

Task 2: Building a Phone Book

Assume that we have a text file with phone book entries. Every entry consists of a person's name and one or more phone numbers for that person. A sample phone book might look like this:

```
Boulder, John M.
    home 020 7815 1234
    office 020 3465 234
Brown, Cynthia 1234567
Douglas, Ann Louise
    office +86 (0)20 234 567
    mobile +86 (0)664 7865 234
...
```

- Names consist of letters and may be abbreviated with a dot.
- Phone numbers consist of an optional country code (e.g. +86), an optional city code (e.g. 020) and a phone number consisting of one or several digit sequences. Country codes start with a '+' and must be followed by a city code (with a '0' in brackets). City codes without country codes start with a '0'. If there is no country code the default is +86. If there is no city code the default is 020.
- Phone numbers may be preceded by the words "home", "office" or "mobile". If such a word is missing the default is "home".

Describe the syntax of such a phone book file by a grammar. Write a Coco/R compiler description that processes such input files by reading them and building a phone book data structure in memory, where every entry of this data structure holds the family name, the first name(s), and the phone number(s) including the country code, the city code and the kind of phone number as separate fields. **Write also a dump method that prints the whole phone book.**