Letzter Abgabetermin: Mittwoch, 21.01.2015, 18^{00} Uhr

Coco/R (24 Punkte)

1. Telefonbuch (12 Punkte)

Verwenden Sie Coco/R, um ein textbasiertes Telefonbuch in eine Datenstruktur einzulesen und auf der Konsole auszugeben. Jeder Eintrag beginnt mit dem Nachnamen gefolgt von einem ',' und beliebig vielen Vornamen, welche auch mit dem Anfangsbuchstaben und einem '.' abgekürzt werden können. Danach folgen mindestens eine oder aber auch beliebig viele Telefonnummern. Jede Telefonnummer besteht aus einer optionalen Länderkennung (zB. „+43“), einer optionalen Vorwahl (zB. „0732“) und einer beliebigen Anzahl von Ziffernsequenzen (aber mindestens einer). Jede Länderkennung beginnt mit '+' oder „00“. Ist eine Länderkennung vorhanden, muss eine Vorwahl folgen. Eine Vorwahl beginnt mit „0“, jedoch muss dann '0' geklammert werden. Jede Telefonnummer kann optional mit „home“, „office“, oder „mobile“ eingeleitet werden.

```
Boulder, John M.
     home 020 7815 1234
     office 020 3465 234
  Brown, Cynthia 1234567
  Douglas, Ann Louise
     office +44 (0)20 234 567
     mobile +43 (0)664 7865 234
```

Erstellen Sie eine Grammatik für dieses Format (die Zeilenumbrüche und Tabulatoren sind NICHT Teil der Syntax) und attributieren Sie diese Grammatik mittels Coco/R, um das Telefonbuch aufzubauen. Implementieren sie ebenfalls eine dump()-Methode, die den Inhalt des gelesenen Telefonbuchs ausgibt. Erstellen Sie ein kleines Telefonbuch und ein Testprogramm, das die Datei mit dem generierten Parser liest und mittels der dump()-Methode ausgibt.

2. Komplexitätsanalyse von MicroJava-Code (12 Punkte)

Implementieren Sie mit Coco/R ein Werkzeug zur Analyse der Komplexität von MicroJava-Code. Verwenden Sie dafür von unserer Website die EBNF-Form der MicroJava-Grammatik, die sie einfach in den Produktionen-Teil der Coco/R-Eingabedatei übernehmen können. Definieren Sie Zeichen, Terminalsymbole und Kommentarregeln nach den Informationen aus den Unterlagen (Abschnitt 4.2 Syntax). Attributieren Sie die schließlich die Grammatik, um nach folgenden Regeln für jede Methode ein Komplexitätsmaß zu berechnen:

- Jede Anweisung zählt einen Punkt
  - *while*-Anweisungen zählen zwei Punkte zusätzlich (also insgesamt 3 Punkte)
  - ein *else*-Zweig bei einem *if*-Konstrukt zählt einen zusätzlichen Punkt
  - jeder einzelne *case* und *default* eines *switch* zählen jeweils einen zusätzlichen Punkt
- Die Schachtelung von Anweisungen wird durch Multiplikation mit einem Faktor $f = 1.5$ berücksichtigt, was zu einer exponentiell steigenden Gewichtung führt (1, 1.5, 2.25, 3.38, 5.06, 7.09, …). Als Schachtelung gilt:
  - Zweige einer *if*-Anweisung (mit oder ohne *{ }*)
  - Schleifenkörper
  - Code in *case* und *default* von *switch*
  - Blöcke, die keinen *if*-Zweig oder Schleifenkörper darstellen

Rechenbeispiel:

```
void primes (int n)
  int[] numbers;
  int i, j;
{
  numbers = new int[n];                1
  i = 0;                               1
  while(i < n) {                       3
    if(1 < i && numbers[i] == 0) {     1
      print(i);                        1
      j = i;                           1
      while(j < n) {                   3
        numbers[j]++;                  1   (1 + 1) * 1.5 = 3
        j += i;                        1
      }                                                          8 * 1.5 = 12
    } else {                           1
      print('.');                      1   1 * 1.5 = 1.5
    }
    i++;                               1
  }                                                                          16.5 * 1.5 = 24.75
}                                                                                              29.75
```

Geben Sie in Ihrem Werkzeug die Komplexität jeder Methode und des Programms (d.h. die Summe der Komplexitäten) aus. Beispielausgabe:

```
Method "fib" has complexity 5.00
Method "primes" has complexity 29.75
Method "main" has complexity 2.00
Program "Numbers" has complexity 36.75
```

Geben Sie ein Testprogramm ab, das eine MicroJava-Datei lesen kann und die Komplexität ausgibt.

Abgabe und Hinweise

Die Abgabe der Übungen muss elektronisch erfolgen. Geben Sie folgende Dateien ab:

- Elektronisch in das Repository: ATG-Datei und Java-Dateien mit der Lösung.
- svn://ssw.jku.at/2014W/UB/k<MatrNr>/branches/UE8

Coco/R kann steht auf http://ssw.jku.at/Coco/#Java zur Verfügung. Sie benötigen die Dateien Scanner.frame, Parser.frame und Coco.jar. Um den Scanner und Parser zu erzeugen, führen Sie in der Konsole aus: `java -jar Coco.jar mygrammer.atg`