FH-OÖ Hagenberg/HSD
Software Entwurf 2, SS 2013
Übung 3

Name: Abgabetermin: KW 15

Mat.Nr: Punkte:

Übungsgruppe: korrigiert:

Aufwand in h:

Beispiel 1 (10 Punkte) Testdaten für mehrere Dateien: Schreiben Sie ein C++ Programm, mit dem Sie Dateien mit zufälligen Testdaten erzeugen können. Erstellen Sie die Dateien im ASCII-Format nach folgender Grammatik:

```
TestData = Header { Value "," } .
Header = "Values" "/" Counter ":".
Counter = Digit {Digit} .
Value = Digit {Digit} .
Digit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0".
```

Der `Counter` gibt die Anzahl der enthaltenen Einträge von `Values` an. Verwenden Sie zur Erzeugung der zufälligen Daten den Zufallszahlengenerator auf der Kommunikationsplattform. Implementieren Sie ihr Programm so, dass gleichzeitig mehrere Dateien mit Testdaten erzeugt werden können. Die Namen der Dateien und die Anzahl der zu erzeugenden Elemente werden als Kommandozeilenparameter übergeben. Die Anzahl der zu erzeugenden Elemente pro Datei ist auf 1000 beschränkt.

Beispiel für eine mögliche Kommandozeile:

```
Testdata.exe TestData1.txt 500 TestData2.txt 1000 TestData3.txt 750
             TestData4.txt 7500 ...
```

Beispiel 2 (14 Punkte) Duplikate entfernen und Schnittmenge bilden: Erweitern Sie das Programm aus Beispiel 1 und lesen Sie mit Hilfe des Scanners alle generierten Dateien entsprechend der Grammatik ein und legen Sie die Werte in einer entsprechenden Datenstruktur ab!

Dateien, die der Grammatik nicht entsprechen, sollen mit einer Hinweismeldung auf der Konsole abgelehnt werden (z.B. `"unknown file format"`).

Implementieren Sie nun einen Algorithmus, der alle mehrfach vorkommenden Elemente (Duplikate) aus einer Datei entfernt und anschließend jene Elemente die in jeder Datei vorkommen (Schnittmenge) in eine Ergebnisdatei (`result.txt`) entsprechend der Grammatik aus Beispiel1 schreibt.

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Verwenden Sie immer **Module**, um den Testtreiber und die eigentliche Implementierung zu trennen! Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit **Kommentaren! Testen** Sie ihre Implementierungen ausführlich! Geben Sie **Lösungsideen** an!