Allgemeine Informatik I für Mathematiker/Wirtschaftsmathematiker (WS 2005/2006)

Abgabetermin: 11. Januar 2006

Auf diesem Übungsblatt beschäftigen wir uns erneut mit dem EVA-Prinzip. In der ersten Aufgabe stellen wir nochmals eine Verbindung zu Grammatiken bzw. Automaten her, die wie vor einigen Wochen bereits kennengelernt haben. Die zweite Aufgabe wartet als Besonderheit dann mit Vorgaben nicht nur für die Ein- sondern sogar auch für die (erwartete) Ausgabe auf. Man könnte bereits von einer kleinen "Spezifikation" sprechen, gegen die Ihr Programm geschrieben werden soll. Dies entspricht im Kern dem Vorgehen, das man auch oftmals bei "realen" Softwareprojekten vorfindet.

Aufgabe 21: Der Ulmer AI1-Automat reloaded (5 Punkte)

Erinnern Sie sich noch an den "Ulmer AI 1-Automaten" aus Aufgabe 12? Damals haben Sie sich Tage und Nächte den Kopf über ihn zerbrochen – und jetzt? Jetzt wärmen wir ihn wieder auf, indem wir ihn in Java simulieren! Denn Ihre Aufgabe lautet schlicht und einfach:

> "Schreiben Sie ein Java-Programm, das den Automaten aus Aufgabe 12 von Übungsblatt 6 simuliert, d.h. eine (beliebig lange) Zeichenfolge ("Satz") von der Standardeingabe liest und ausgibt, ob dieser Satz von dem Automaten erkannt wird oder nicht. Selbstverständlich soll er auch überprüfen, daß als Eingabe nur die erlaubten Zeichen (Terminal-Symbole) kommen."

Beispiel:

```
theseus$ java Automata
Bitte geben Sie den zu bewertenden Satz ein:
100
Der eingegebene Satz gehoert zur Grammatik!

theseus$ java Automata
Bitte geben Sie den zu bewertenden Satz ein:
10000100
Der eingegebene Satz gehoert zur Grammatik!

theseus$ java Automata
Bitte geben Sie den zu bewertenden Satz ein:
10101010
Der eingegebene Satz gehoert nicht zur Grammatik!

theseus$ java Automata
Bitte geben Sie den zu bewertenden Satz ein:
01210
Der eingegebene Satz enthaelt das unerlaubte Zeichen: 2!
```

Aufgabe 22: Software Engineering light (5 Punkte)

Nach 10 Wochen Informatik fühlen Sie sich mittlerweile dermaßen fit, daß Sie bei einem großen Software-Konzern anheuern um sich ein wenig Ihr Taschengeld aufzubessern.

Mit der *Human Resources*-Abteilung (hieß früher einfach Personalabteilung) werden Sie sich schnell einig über das Salär, der *Facility Manager* (hieß früher Hausmeister) überreicht Ihnen die *Personal Access Card* (früher: Firmen-Ausweis) und schon finden Sie sich an Ihrem *desktop* (Schreibtisch) als *Associate Junior Programmer* ("der Neue") wieder.

Nach dem *Lunch* mit dem gesamten *Team* bekommen Sie ihre Aufgabe, pardon – Ihren *Task*:

> "Schreiben Sie ein Java-Programm, das drei Integer-Zahlen von der Standard-Eingabe liest. Als Ausgabe liefert Ihr Programm dann, ob es sich um ein Dreieck handelt und wenn ja, um welches. Wie Sie das machen, ist mir egal – Hauptsache, unser Testroboter liefert für unsere Testeingaben die in der Spezifikation stehende Ausgabe."

Ihr Vorgesetzter muß sofort weg zu einem wichtigen *Meeting* und hat keine Zeit mehr für weitere Erklärungen.

Testroboter? Spezifikation? Drei Zahlen? Dreieck? Hä? Da stehen Sie nun und ärgern sich, daß Sie zum gleichen Lohn (plus ein freies Essen täglich inklusive) nicht doch beim Fast-Food-Laden um die Ecke angefangen haben. Aber zu Ihrer Rettung kommt die hübsche Praktikantin (die hier übrigens *student apprentice* heißt) und erklärt Ihnen alles nochmal in Ruhe:

- Dein Programm soll einfach nur drei Integer-Zahlen einlesen, welche die drei Seitenlängen eines Dreiecks bedeuten.
- Nach der Eingabe überprüft Dein Programm dann, ob diese drei Seiten überhaupt ein Dreieck bilden[1] und wenn ja, ob es vielleicht sogar ein gleichseitiges oder gleichschenkliges ist.

[1] Dazu könnte man zum Beispiel die Dreiecksungleichung prima verwenden...

- Nehmen wir ein Beispiel: Angenommen, Du gibt die Seitenlängen 3, 3 und 5 ein. Dann sollte Dein Programm die Ausgabe `Wow, ein gleichschenkliges Dreieck!` liefern.
- Falls die Eingabe fehlgeschlagen ist oder ungültige (z.B. negative) Zahlen eingegeben wurden, soll das Programm entsprechende Fehlermeldungen ausgeben.
- Auf unserem FTP-Server liegt die Datei *testdaten*, die einige Eingabedaten für verschiedene Fälle beinhaltet.
- Mache Dir zunächst Gedanken darüber, welche Ausgabe Dein Programm für diese Eingaben produzieren sollte.
- Außerdem findest Du auf unserem FTP-Server die Datei *ExpectedResults* sowie unseren *Testroboter* (für Linux und Solaris).
- Der Testroboter hat den Sinn, Dein Programm mit Testdaten aus einer als Parameter übergebenen Datei[2]zu "füttern" und die Ausgaben Deines Programms in die Datei `results` aufzuzeichnen.
- Damit Dein Testroboter funktioniert, muß Dein Programm den Namen `Triangle.java` bzw. `Triangle.class` heißen.
- Was es genau mit dem Testroboter auf sich hat, erklärt man Dir in den Übungen.
- Du hast Deine Aufgabe genau dann zu unserer vollsten Zufriedenheit erfüllt, wenn die erzeugte Datei `results` mit der Datei *ExpectedResults* übereinstimmt.

Na bitte, wenigstens ein normaler Mensch in dem Laden! Jetzt, wo Ihnen die Aufgabe klar ist, können Sie sich ja sogleich an die Arbeit machen!

Noch ein paar Hinweise

- Damit der Testroboter funktioniert, ist es notwendig, daß das Programm `Triangle.java` (bzw. `Triangle.class`) heißt.
- Ebenso ist es (für den Testroboter) notwendig, daß nur das "eigentliche" Ergebnis ausgegeben wird. Auf Eingabeaufforderungen ("Bitte geben Sie ein...") muß hier verzichtet werden.

---

[2]Das ist hier natürlich sinnvollerweise die Datei `testdaten`!

Das SAI-Team wünscht Ihnen
ein frohes Weihnachtsfest
und alles Gute für das Jahr 2006!