Dr. Andreas Borchert
Michael Wiedemann

Allgemeine Informatik II (SS 2004)

Abgabetermin: 17. Juni 2004

5 Lass uns darüber reden ... – 10 Punkte

Um dem Tutor mal zu zeigen, was Ihr inzwischen alles gelernt habt, solltet Ihr ihm (mündlich) folgende Fragen beantworten können:

- Was generell ist überhaupt Backtracking?
- Wie funktioniert Backtracking?
- Wie sieht der typische Ablauf eines Oberon-Programms aus (es kann zum Beispiel das EVA Programm benutzt werden)?
- Was ist Parsieren?
- Wie funktioniert eine Rekursion?
- Ist der Aussage ,,Eine rekursive Lösung ist immer besser als eine iterative! " generell zuzustimmen? Begründet Eure Aussage.

Ziel des ganzen ist es, einen Dialog mit Eurem Tutor herzustellen und somit Stärken (und vielleicht auch Schwächen) festzustellen und gegebenenfalls auszuräumen.

6 Wie zahle ich die Zeche? – 10 Punkte

Man stelle sich vor, Ihr geht mit Eurem Freund / Eurer Freundin gepflegt essen. Der Abend verläuft äußerst harmonisch, bis es ans Bezahlen geht. Damit Ihr mit Eurer Begleitung möglichst schnell nach Hause kommt und der Ober so wenig Arbeit wie möglich mit Euch hat, wollt Ihr mit so wenig Münzen wie möglich auskommen. Ihr habt, wie so oft, n Münzen im Geldbeutel mit Werten $w_1, w_2, \ldots, w_n$ und sollt also den Betrag Z mit möglichst wenig Münzen bezahlen (der Einfachheit halber nehmen wir an, höherer Geldeinheiten liegen auch in Münzen vor, wir machen also keinen Unterschied zwischen Münzen und Scheinen).

Hierbei handelt es sich um ein klassisches Problem, welches sich mit Hilfe von *Branch and Bound* lösen lässt.

Korrekte Eingaben spezifizieren wir durch folgende EBNF:

CoinProblem = *Amount* { *Coin* } .
Coin = *Quantity Value* .
Amount = *Integer* .
Quantity = *Integer* .
Value = *Integer* .

Zum Betrag Z:
es kann angenommen werden, dass entweder
1. mindestens eine Lösung existiert, so dass der Betrag genau durch die Münzen abgedeckt wird oder
2. keine solche Lösung existiert (was dann entsprechend von Eurem Programm ausgegeben werden soll.
Das bedeutet, der Betrag Z soll genau erreicht werden.

Tipps:
Als Einstieg solltet Ihr Euch also überlegen, welche Sätze zum Beispiel zu der definierten Sprache gehören (vergleiche zum Beispiel Blatt 3 – Parsieren) und dementsprechend Eure Einleseprozedur anpassen. Das bedeutet, Sätze, die nicht zu der definierten Sprache gehören, sollten abgelehnt werden. Dementsprechend sollte die Prozedur bei solchen Sätzen reagieren. Selbstverständlich erhält Euer Programm immer nur einen Satz, Ihr habt ja schliesslich im Regelfall nur einen Geldbeutel dabei.

Gehören zum Beispiel folgende Sätze zur Sprache:
123 2 3 4
123 2 3 4 5
123
123 1

Überlegt Euch eine passende Datenstruktur.

Zunächst werden für die ,, Unterforderten " unter Euch keine weiteren Tipps zur Verfügung gestellt. Ab Anfang / Mitte nächster Woche werden dann allerdings weitere Hilfestellungen veröffentlicht (Rumpf, etwas zum Einlesen, Hinweise zur Datenstruktur, Beispiel-Prozeduren usw.), um allen die Möglichkeit zu geben, die Aufgabe einigermaßen zu lösen.

Viel Erfolg!