10. Aufgabenblatt zu Funktionale Programmierung vom 15.01.2004, fällig: 21.01.2004

Themen: *Ein- und Ausgabe anhand eines einfachen Interpretierers*

Für dieses Aufgabenblatt sollen Sie einen Datentyp und Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe10.hs` auf oberstem Niveau in Ihrem Homeverzeichnis ablegen. Kommentieren Sie Ihre Programme wieder aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

- Die Funktion `inOut::(String->String)->IO Integer` soll wiederholt Zeilen von `stdin` lesen, auf die Zeichenketten (ohne New-Lines „`\n`") die übergebene Funktion anwenden und die Ergebnisse als getrennte Zeilen in `stdout` schreiben. Bei der Eingabezeile `"end"` wird abgebrochen; `"end"` ist nicht mehr zu bearbeiten. Das Ergebnis von `inOut` ist die Anzahl ausgegebener Zeilen.
- Führen Sie den Datentyp `Token` mit den Datenkonstruktoren `Invalid`, `Mult`, `Add`, `Fak` und `Pot` (jeweils Parameter) sowie `Value` mit einem Parameter vom Typ `Integer` ein. `Token` soll von der Typklasse `Eq` sein. Schreiben Sie eine Funktion `tokenize::String->[Token]`, die einen String in eine Liste von Token umwandelt. Jedes Zeichen `'*'` soll in `Mult`, jedes `'+'` in `Add`, jedes `'!'` in `Fak` und jedes `'^'` in `Pot` umgewandelt werden. Jede ununterbrochene Folge von Ziffern (möglicherweise mit direkt vorangestelltem `'-'`) soll in `Value x` umgewandelt werden, wobei `x` die Zahl ist, die durch die Zeichenfolge dargestellt wird. Leerzeichen werden in kein Token umgewandelt. Alle anderen Zeichen werden in `Invalid` umgewandelt.
- Die Funktion `filter::[Token]->[Token]` soll in der Argumentliste alle Vorkommen des Tokens `Invalid` streichen, ansonsten die Reihenfolge der Token in der Argumentliste unverändert lassen.
- Die Funktion `toString::[Integer]->String` soll die Argumentliste umdrehen und in einen String mit durch Leerzeichen getrennten Zahlen umwandeln. Z. B. soll `[1,-2,35]` in `"35 -2 1"` umgewandelt werden. Die leere Liste soll in `"Error"` umgewandelt werden.
- Schreiben Sie eine Funktion `interpret::[Token]->[Integer]`, die Tokenlisten mit Hilfe eines Stacks auswertet und den Stack (als Liste, „top of stack" vorne) zurück gibt. Anfangs ist der Stack leer. Für `Value`-Token wird die Zahl auf den Stack gelegt. Für `Mult` und `Add`

werden die beiden obersten Zahlen vom Stack genommen, multipliziert bzw. addiert und das Ergebnis auf den Stack gelegt. Ähnlich für `Pot`. Auch hier werden die beiden obersten Zahlen vom Stack genommen und durch das Resultat von $a^{|b|}$ ersetzt, wobei a das oberste und $|b|$ den Absolutbetrag des zweitobersten Stackelements bezeichnet. `Fak` schließlich ersetzt das oberste Stackelement a durch den Wert von $a!$ ("a Fakultät"), falls a größer oder gleich 0 ist, und durch -1 sonst. Wenn der Stack zu wenige Elemente enthält, gibt `interpret` die bis dahin aufgebaute Liste zurück.

- Schreiben Sie die Funktion `main::IO()`, die `inOut` so aufruft, dass jede gelesene Zeile mit Hilfe von `tokenize`, `filter`, `interpret` und `toString` in eine Ausgabezeile umgewandelt wird. Zum Schluß soll "n `Zeilen bearbeitet`" ausgegeben werden, wobei n das Ergebnis von `inOut` ist.