

Praktikumsblatt zur Wiederholung

Folgendes ist zu beachten:

- Das Praktikumsblatt gilt sowohl für HASKELL als auch für C/C++. Die jeweilig erreichbare Punktzahl ist für beide Teile explizit angegeben.
- Es sind keinerlei Vorgaben für Datenstrukturen und die Form von Ein- und Ausgaben gemacht. Für jede der Aufgaben ist dies eigenständig zu konzipieren und zu implementieren. Das betrifft auch die Wahl der Testfälle.
- Die Abgabe erfolgt direkt beim Betreuer, der legt auch die Form der Abgabe fest.

Aufgabe 1 (HASKELL: 10 Punkte | C/C++: 10 Punkte)

Gegeben sei eine Menge M von natürlichen Zahlen und eine natürliche Zahl z . Gesucht ist eine Darstellung von z als Produkt von gewissen Elementen aus M , wobei

1. in der Version 1 jedes Element aus M höchstens einmal benutzt werden darf,
2. in der Version 2 jedes Element aus M beliebig oft benutzt werden darf.

Aufgabe 2 (HASKELL: 10 Punkte | C/C++: 15 Punkte)

Betrachtet werden binäre Bäume, deren Knoten mit ganzen Zahlen markiert sind. Implementieren Sie folgende Funktionen:

1. Es ist festzustellen, ob ein Baum ub mit dem Unterbaum von einem Baum b an einer gegebenen Adresse α übereinstimmt.
2. Es sind alle Adressen zu bestimmen, an denen ein Baum ub mit einem Unterbaum von einem Baum b übereinstimmt.

Aufgabe 3 (HASKELL: 20 Punkte | C/C++: 25 Punkte)

Wir betrachten Texte, d.h. endliche Folgen von Zeichen, die nur aus Buchstaben (ohne Umlaute) und Dezimalziffern bestehen.

Ein Muster ist ein solcher Text, in dem zusätzlich auch die "Wildcards" $?$ und $*$ auftreten können. Wir sagen: ein *Match* eines Musters m mit einem Text t ist eine Teilzeichenkette von t , die aus m entsteht, indem die in m auftretenden Wildcards $?$ durch jeweils einen beliebigen Buchstaben oder eine beliebige Dezimalziffer und die in m auftretenden Wildcards $*$ durch einen beliebigen Text (auch den leeren Text) ersetzt werden. Implementieren Sie Funktionen, die

1. einen Match eines Musters mit einer Zeichenkette bestimmt.
2. alle Matches eines Musters mit einer Zeichenkette bestimmt.

Aufgabe 4 (HASKELL: 20 Punkte | C/C++: 20 Punkte)

Ausdrücke der Booleschen Algebra (der Aussagenlogik), geschrieben in der Prefix-Notation, bestehen aus:

- Aussagensymbolen,
- dem einstelligen Operator für die Negation (\neg),
- dem zweistelligen Operator für die Konjunktion (\wedge),
- dem zweistelligen Operator für die Disjunktion (\vee),
- dem zweistelligen Operator für die Implikation (\rightarrow) und
- dem zweistelligen Operator für die Äquivalenz (\leftrightarrow).

Für eine gegebene Zuordnung von Aussagensymbolen mit den Wahrheitswerten *wahr* und *falsch* wird ein Ausdruck entsprechend der bekannten Wirkung der Operatoren *wahr* oder *falsch*.

Ein *Literal* ist ein Aussagensymbol oder eine Negation gefolgt von einem Aussagensymbol.

Eine *Klausel* ist eine Menge von Literalen. Für eine gegebene Zuordnung von Aussagensymbolen mit den Wahrheitswerten *wahr* und *falsch* wird eine Klausel genau dann wahr, wenn eines seiner Literale wahr wird. (Die leere Klausel kann deshalb nicht *wahr* werden!)

Jeder Ausdruck a der Booleschen Algebra kann in eine Menge K von *Klauseln* so transformiert werden, dass für jede Belegung der Aussagensymbole mit Wahrheitswerten a genau dann wahr wird, wenn jede Klausel aus K wahr wird.

Folgende Transformationsregeln können genutzt werden, um Konjunktionen von Disjunktionen von Literalen zu erzeugen:

- $a b \Rightarrow \wedge \rightarrow a b \rightarrow b a$
- $\rightarrow a b \Rightarrow \vee \neg a b$
- $\neg \wedge a b \Rightarrow \vee \neg a \neg b$
- $\neg \vee a b \Rightarrow \wedge \neg a \neg b$
- $\vee \wedge a b c \Rightarrow \wedge \vee a c \vee b c$
- $\vee a \wedge b c \Rightarrow \wedge \vee a b \vee a c$

Es verbleibt, jede Disjunktion in eine Klausel und jede Konjunktion in eine Menge von Klauseln zu transformieren.

Implementieren Sie eine Funktion, die einen Ausdruck der Booleschen Algebra in eine solche Menge von Klauseln transformiert.

Aufgabe 5 (HASKELL: 25 Punkte | C/C++: 20 Punkte)

Ein arithmetischer Ausdruck sei eine Zeichenkette, die aus dem Metasymbol AA der folgenden Grammatik ableitbar ist.

$$\begin{aligned} AA & ::= MD \mid MD+AA \mid MD-AA \\ MD & ::= OP \mid OP*MD \mid OP-MD \\ OP & ::= ZA \mid (AA) \\ ZA & ::= DZ \mid DZ ZA \\ DZ & ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

Ein arithmetischer Formelbaum ist ein Baum, dessen Blätter mit einer Zahl markiert sind, die anderen Knoten sind mit einem der Zeichen $+, -, *, /$ markiert und besitzen 2 Söhne. Ein Blatt repräsentiert seine Markierung, eine Zahl, ein Knoten mit einem der Zeichen $+, -, *, /$ als Marke die entsprechende Operation mit den Unterbäumen als Operanden.

Implementieren Sie

- Eine Funktion, die prüft, ob ein Text ein korrekt gebildeter arithmetischer Ausdruck ist und daraus einen entsprechenden Formelbaum erzeugt.
- Eine Funktion, die den Wert eines arithmetischen Ausdrucks bestimmt.

Aufgabe 6 (HASKELL: 30 Punkte | C/C++: 30 Punkte)

Wir betrachten ein Schachbrett der Seitenlänge $2 * n$. Auf jedem der schwarzen Feldern kann ein Spielstein platziert sein, insgesamt maximal $2 * n * n$ viele.

Ein Spielstein kann einen Zug machen, wenn eine der diagonalen Positionen links oben, rechts oben, links unten oder rechts unten von einem Spielstein besetzt und die entsprechende diagonal darüber bzw. darunter liegende Position frei ist. Der übersprungene Stein gilt als geschlagen und wird vom Brett entfernt. Das entspricht einem Sprung beim Damespiel.

Implementieren Sie folgende Funktionen:

- Kann von einer gegebenen Position aus eine Zugfolge von n -Zügen erfolgen?
- Wie lang ist die maximale Zugfolge von einer gegebenen Position aus?
- Kann ein Stein von einer gegebenen Position aus alle anderen Steine schlagen?
- Wie lang ist die maximale Zugfolge, die einer der Steine auf dem Brett ausführen kann?