Aufgabe 1 Festival-Algebra

25 Punkte

In dieser Aufgabe sollen Sie eine Algebra entwerfen, mit der sich Personen, die sich auf einem Festival befinden, verwalten lassen.

Das Festivalgelände selbst besteht aus verschiedenen Camps, die sich durch einen Namen identifizieren lassen und jeweils eine Menge von Personen beherbergen. Das Gelände vor der Bühne wird ebenfalls als Camp mit dem Namen „stage“ modelliert. Ein weiteres spezielles „Camp“ trägt den Namen „home“ und bezeichnet das Gebiet außerhalb des Geländes. Die dazugehörige Menge an Personen ist stets leer, da dieses Camp sich außerhalb der Festivalverwaltung befindet.

Jede Person besitzt einen eindeutigen Namen, ein Heimatcamp sowie einen gewissen Vorrat an Raviolidosen, Bier und Geld. Unnütze Dinge wie Zelt, Hygieneartikel etc. werden zur Vereinfachung in dieser Algebra nicht modelliert.

Die Funktionen der Algebra sind die folgenden:

createPerson Hier wird eine Person mit Namen, dem Vorrat an Ravioli, Bier und Geld erzeugt. Das Heimatcamp ist das „home“-Camp.

createFestival Diese Funktion dient dem Aufbau des Festivalgeländes. Nach Aufruf dieser Funktion enthält das Gelände nur das Camp „stage“, in dem sich (noch) keine Personen befinden.

insertPerson Diese Operation modelliert die Anreise einer Person auf das Festival. Dieser Person wird über ein Argument, das den Namen des Heimatcamps angibt, ein Heimatcamp zugewiesen. Die Person selbst muss vorher als Heimatcamp „home“ haben, sonst ist diese Funktion wirkungslos. Eine Anreise direkt vor die Bühne ist nicht erlaubt. Existiert bislang kein Camp mit diesem Namen, wird ein neues Camp mit genau dieser einen Person eröffnet, ansonsten wird diese Person in das bereits existierende Camp eingeordnet. Dieses Camp wird nun zum Heimatcamp der Person.

name Liefert den zu einer Person oder einem Camp gehörenden Namen.

names Liefert die Namen aller Personen, die sich derzeit in einem Camp oder auf dem gesamten Festival aufhalten.

cnames Liefert die Namen aller Camps eines Festivals.

who Ermittelt zu einem Namen die komplette Person auf dem Festival oder in einem Camp. Sollte die Person nicht auf dem Festival anwesend sein, ist das Ergebnis Person mit Heimatcamp „home“, ohne Vorräte und mit Namen „nobody“.

position Hiermit wird der derzeitige Aufenthaltsort (Camp) einer Person (gegeben durch ihren Namen) auf dem Festival ermittelt. Rückgabe bildet das komplette Camp. Sollte die Person sich nicht auf dem Festivalgelände aufhalten, ist das Ergebnis das Camp mit Namen „home“ ohne Personen.

visit Diese Funktion beschreibt den Besuch einer Person in einem anderen Camp oder den Gang zur Bühne. Als Eingabe dienen das Festival, der Name der Person sowie der Name des Zielcamps. Die Rückgabe ist ein Festival. Sollte das Zielcamp dem aktuellen Aufenthaltsort der Person entsprechen, die Person oder das Zielcamp nicht gefunden werden, bleibt das Festival unverändert. Ist der Aufenthaltsort der Person ihr Heimatcamp, verringert sich ihr Biervorrat

um eins, sofern dieser noch nicht erschöpft ist. Der Fachmann spricht hier vom „Wegbier“.

eat Hier dienen das Festival und der Name einer Person als Eingabe, die Rückgabe ist ein Festival. Diese Funktion zeigt nur dann eine Wirkung, wenn sich die betreffende Person in ihrem Heimatcamp oder vor der Bühne befindet. Im Heimatcamp wird der Ravioli-Vorrat um eins verringert, falls noch welcher vorhanden ist. Vor der Bühne hingegen wird der Geldvorrat um 5 € verringert, falls die Person über genügend finanzielle Mittel verfügt.

drink Eingabe und Rückgabe dieser Funktion entsprechen denen der *eat* Funktion. Auch hier ist das Ergebnis abhängig vom aktuellen Aufenthaltsort der Person. Im Heimatcamp wird der Biervorrat der Person um eins vermindert, sofern dieser vorhanden ist. Vor der Bühne werden vom vorhandenen Geld der Person 3€ abgezogen, solange sich die Person das Bier noch leisten kann. In einem fremden Camp wird das Bier von der Person ausgegeben, deren Heimatcamp das besuchte Camp ist, die anwesend ist und die unter allen Personen, die diese Eigenschaften ebenfalls erfüllen, das meiste Bier besitzt. Sollte dies für mehrere Personen zutreffen, gibt die mit dem alphabetisch kleinsten Namen das Bier aus. Ist keine solche Person vorhanden, z.B. bei akuter Biernot, bleibt diese Operation wirkungslos.

Verwenden Sie in der Algebra unter Anderem die Sorten *person*, *camp*, *festival* mit den folgenden Trägermengen:

$$person = \{(name, camp, beer, ravioli, money) | name, camp \in string, beer, ravioli, money \in int^+\}$$

$$camp = \{(n, p) | n \in string, p \subseteq person, \forall p_1, p_2 \in p : p_1 = p_2 \vee name(p_1) \neq name(p_2)\}$$

$$festival = \{c | c \subseteq camp, \forall c_1, c_2 \in c : c_1 = c_2 \vee names(c_1) \cap names(c_2) = \emptyset\}$$

(a) Geben Sie die Signatur der Algebra an. Sie können dabei von der Existenz der Typen *int*, int^+, *bool* und *string* mit den üblichen Operatoren ausgehen. Hierbei gilt: $int^+ = \{i \in int | i \geq 0\}$. Weiterhin können Sie annehmen, dass der Test auf Gleichheit (=) für alle Typen (auch für neu definierte) existiert. Eine Signatur für die Funktion *eat* finden Sie weiter unten. Diese muss nicht mit angegeben werden. *6 Punkte*

(b) Geben Sie Trägermengen für die Sorten der Algebra an, die nicht vorausgesetzt sind und deren Trägermengen noch nicht definiert wurden. *3 Punkte*

(c) Geben Sie die Funktionsdefinitionen für die Operatoren der Algebra außer für *eat* an. Sie können dabei von der Existenz der folgenden Funktionen ausgehen: *16 Punkte*

getHome ermittelt das Heimatcamp einer Person.
$getHome : person \rightarrow string$

searchCamp ermittelt das Camp eines Festivals zu einem gegebenen Namen.
$searchCamp : festival \times string \rightarrow camp$

contains prüft, ob in einem Camp eine Person mit dem angegebenen Namen vorhanden ist.
$contains : camp \times string \rightarrow bool$

decBeer verringert den Biervorrat einer Person um eins, solange dieser nicht erschöpft ist.
$decBeer : person \to person$

decRavioli verringert den Raviolivorrat einer Person um eins, solange dieser nicht erschöpft ist.
$decRavioli : person \to person$

decMoney verringert den Geldvorrat einer Person um einen gegebenen Euro-Betrag, falls dieser höher als der angegebene Betrag ist.
$decMoney : person \times int^+ \to person$

add fügt einem Camp eine Person hinzu, solange dort noch keine Person mit gleichem Namen existiert. Das Camp wird zum Heimatcamp der Person.
$add : camp \times person \to camp$

getDonor ermittelt die zu einem Camp gehörende Person, die dort das nächste Bier ausgeben muss (Regel hierfür siehe *drink*-Operator). Existiert diese Person nicht, so ist das Ergebnis eine Person namens „nobody“ mit Heimatcamp „home“ und ohne Vorräte
$getDonor : camp \times string \to person$

Hinweis: Wenige der genannten Operatoren können etwas komplexer werden. Um Ihnen zu zeigen, wie die Funktionen solcher Operatoren lesbar aufgeschrieben werden können, zeigen wir hier die Signatur sowie die Funktionsdefinition für *eat*. Dieser Operator erhält ein Festival sowie den Namen einer Person als Eingabe und liefert das Festival mit evtl. geänderten Personendaten als Ergebnis.

$$eat: \quad festival \times string \quad \to festival$$

$$eat(f,n) = \begin{cases} f & \text{falls } (k \neq getHome(p) \wedge k \neq \text{„stage“}) \vee n \notin names(f) \\ f_1 & \text{falls } k = getHome(p) \wedge n \in names(f) \\ f_2 & \text{sonst} \end{cases}$$

mit:
$p = (n, c, r, b, m) = who(f, n)$, die Person zum Namen
$k = name(position(f, n))$, Name des Heimatcamps der Person
$f_1 = (f \setminus \{c_1\}) \cup \{c_2\}$
$f_2 = (f \setminus \{c_1\}) \cup \{c_3\}$
$c_1 = (n_1, p_1) = position(f, n)$, Heimatcamp der Person
$c_2 = (n_1, (p_1 \setminus \{p\}) \cup \{decRavioli(p)\})$
$c_3 = (n_1, (p_1 \setminus \{p\}) \cup \{decMoney(p, 5)\})$

Im ersten Fall befindet sich die Person, deren Name angegeben wurde, weder vor der Bühne noch im eigenen Camp. Da woanders nicht gegessen werden kann, wird das unveränderte Festival zurückgegeben. Der zweite Fall (f_1) beschreibt die Veränderung des Festivals, wenn sich die Person gerade in ihrem Heimatcamp aufhält, im letzten Fall befindet sich die Person vor der Bühne. In den letztgenannten Fällen wird das Camp, in dem sich die Person gerade befindet (c_1), aus der Menge der Camps entfernt und das geänderte Camp (c_2 bzw. c_3) in die Menge eingefügt. In beiden Fällen wird die Person aus dem Camp entfernt und die geänderte Person (weniger Ravioli oder weniger Geld) zur Personenmenge des Festivals hinzugefügt.

Aufgabe 2 Hashing

20 Punkte

Fügen Sie die unten angegebene Folge von Strings in eine Hashtabelle der Größe 10 ein. Verwenden Sie als Hashfunktion die Mittel-Quadrat-Methode und Quadratisches Sondieren als Kollisionsstrategie. Um den initialen Zahlenwert für die Mittel-Quadrat-Methode zu erhalten, summieren Sie die Buchstabencodes ('a' = 'A' = 1, 'b'= 'B'=2 usw.) der ersten drei Buchstaben der Zeichenkette auf. Bei einer geraden Anzahl von Ziffern wählen Sie bitte die weiter rechts stehende Zahl als mittlere Zahl aus. Geben Sie für jeden Eintrag den initialen Hashwert sowie ggf. die Folge der Positionen an, bei denen eine Kollision auftritt. Geben Sie schließlich die gefüllte Hashtabelle an.

Berlin, Bochum, Frankfurt, Hamburg, Karlsruhe, Dortmund, Bregenz, Wien

Aufgabe 3 Sortieren

31 Punkte

(a) Gegeben sei die folgende Zahlenfolge *14 Punkte*

4, 6, 42, 10, 5, 15, 11, 24, 1

Sortieren Sie die obige Zahlenfolge *aufsteigend* mit Hilfe von Quicksort. Geben Sie dazu den Baum der rekursiven Aufrufe an. Jeder Knoten des Baum entspricht dabei einem Aufruf von Quicksort und enthält die Eingabefolge des Aufrufs. Die Kinder eines Knotens entsprechen den beiden rekursiven Aufrufen. Der Merge-Schritt wird nicht dargestellt. Markieren Sie in jedem Knoden das Pivotelement, das mit der Funktion *findx* des Kurstextes zu bestimmen ist.

(b) Begründen Sie anhand eines eigenen Beispiels, dass Quicksort nicht stabil ist. *3 Punkte*

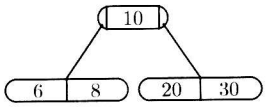
(c) Sortieren Sie die folgenden Wörter alphabetisch mittels Radixsort. Geben Sie hierzu die Inhalte der nicht-leeren Behälter nach jeder Phase an. *14 Punkte*

Hund, Teil, Ast, Mund, Hai, Wut, Kahn, Fahne, Kai, Wolf, Backe, Wetter

Aufgabe 4 B-Bäume

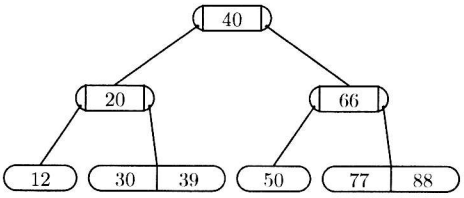
23 Punkte

(a) Fügen Sie in den folgenden B-Baum der Ordnung 1 die Schlüsselwerte der Folge F in der gegebenen Reihenfolge ein. Zeichnen Sie den Baum vor und nach jeder Strukturänderung. *14 Punkte*



$F =< 9, 25, 27, 29, 23 >$

(b) Löschen Sie aus dem unten angegebenen B-Baum der Ordnung 1 nacheinander die Elemente 12, 39, 66 und 30. Geben Sie den Baum jeweils vor und nach einer Strukturänderung an und benennen Sie die Art der Unterlaufbehandlung. *9 Punkte*



Aufgabe 5 Deckblatt

1 Punkt

Lesen Sie sich die „Hinweise zur Bearbeitung“ sorgfältig durch. Füllen Sie den dort aufgeführten Anweisungen entsprechend beide Deckblätter vollständig und korrekt aus.