

Übungen zu Informatik I

In den folgenden Aufgaben leistet Ihnen die Funktion *print* (vom Typ `string` \rightarrow `unit`), die einen Text auf dem Bildschirm ausgibt, gute Dienste.

Aufgabe 10-1 Türme von Hanoi (keine Abgabe)

Die „Türme von Hanoi“ sind ein berühmtes Geduldspiel: Man verwendet n Holzscheiben unterschiedlichen Durchmessers, welche durch ein Loch in der Mitte auf Holzstäbe gesteckt werden können. Drei solche Stäbe sind vorhanden. In der Ausgangssituation befinden sich alle n Scheiben auf Stab 1 und zwar der Größe nach geordnet, also die kleinste Scheibe oben und die größte unten. Die kleinste und oberste bezeichnen wir als Scheibe 1, die unterste ist Scheibe n . Ziel des Spiels ist es, den Scheibenturm von Stab 1 nach Stab 3 zu transportieren, so dass wieder alle Scheiben der Größe nach geordnet (kleinste oben) liegen. Stab 2 steht zur Zwischenablage von Scheiben zur Verfügung. In jedem Zug darf man nur die oberste Scheibe eines Turms entfernen und auf einen der beiden anderen legen, allerdings darf dabei niemals eine größere Scheibe auf eine kleinere gelegt werden.

Gegeben sei der Variantentyp `datatype Stab = Eins | Zwei | Drei`. Geben Sie eine SML-Funktion `hanoi : int * Stab * Stab \rightarrow unit` an, so dass `hanoi(n, Eins, Drei)` eine Lösung für das Türme von Hanoi-Problem mit n Scheiben ausdrückt, d.h. auf dem Bildschirm sollen zielführende Anweisungen im exakten Wortlaut „Bringe Scheibe i von Stab j nach Stab k .“ *untereinander* ausgegeben werden. Dabei müssen alle Züge den obigen Regeln gehorchen.

Hinweis: Schreiben Sie zunächst eine Funktion `hanoistr : int * Stab * Stab \rightarrow string`, die den auszugebenden Text erzeugt. Die Funktion `Int.toString` liefert die textuelle Darstellung ganzer Zahlen.

Aufgabe 10-2 Kryptographie (keine Abgabe)

Sie wollen mit den anderen Teilnehmern Ihrer Übungsgruppe verschlüsselte Nachrichten austauschen. Die Ver- und Entschlüsselung soll von einem SML-Programm durchgeführt werden.

Als Grundlage der Verschlüsselung verwenden Sie eine so genannte *monoalphabetische Substitution*. Dabei wird jeder Buchstabe durch einen anderen Buchstaben ersetzt, z.B.

$$a \rightsquigarrow v, \quad b \rightsquigarrow c, \quad \dots \quad x \rightsquigarrow u, \quad y \rightsquigarrow f, \quad z \rightsquigarrow h.$$

Um die Aufgabe zu vereinfachen, werden nur die Kleinbuchstaben a – z verschlüsselt, alle anderen Zeichen bleiben im Klartext erhalten. Dann lässt sich die Substitution durch eine Liste von 26 Buchstaben beschreiben: Der erste Buchstabe beschreibt die Verschlüsselung von a , der zweite die Verschlüsselung von b usw. Die als Beispiel angegebene Substitution ergibt also die Liste

$$["v", "c", \dots, "u", "f", "h"].$$

Wir nennen eine derartige Liste einen *Schlüssel*. Zur Abkürzung schreiben wir diesen Schlüssel auch in der Form $[v, c, \dots, u, f, h]$.

- a) Schreiben Sie eine SML-Funktion `encode_char` vom Typ `char list * char \rightarrow char`, so dass ein Aufruf `encode_char(k, c)` das Zeichen c mit dem Schlüssel k verschlüsselt.

- b) Schreiben Sie eine SML-Funktion *encode* vom Typ **char list * string → string**, so dass ein Aufruf *encode(k, s)* den Text *s* mit dem Schlüssel *k* verschlüsselt. *Hinweis:* Verwenden Sie die Funktion *explode* : **string → char list**, um den Text in die Liste seiner Zeichen umzuwandeln, führen Sie eine Rekursion über dieser Liste aus, und verwenden Sie dann die Funktion *implode* : **char list → string** um die Liste wieder in einen Text umzuwandeln.
- c) Schreiben Sie eine SML-Funktion *decode* vom Typ **char list * string → string**, die die von *encode* durchgeführte Verschlüsselung entschlüsselt, d.h. für die gilt: Wenn *s'* das Ergebnis von *encode(k, s)* ist, so gibt ein Aufruf *decode(k, s')* den Text *s* zurück.
- d) Entschlüsseln Sie den mit dem Schlüssel

$[d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, a, b, c]$

verschlüsselten Text "Furkh Whlkqdfkwhq".

- e) Um die Sicherheit der Verschlüsselung zu verbessern, implementieren Sie folgendes Verfahren: Sie geben sich zwei verschiedene Schlüssel vor und verschlüsseln alle Zeichen eines Textes, die an einer Position mit geradem Index vorkommen, mit dem ersten Schlüssel, alle anderen Zeichen mit dem zweiten Schlüssel. (Das erste Zeichen des Textes hat den Index 0.) Z.B. wird bei einem Aufruf *encode2(k1, k2, "abcde")* der Buchstabe **a** mit *k1* verschlüsselt, **b** mit *k2*, **c** mit *k1*, **d** mit *k2* und **e** mit *k1*. Implementieren Sie eine Funktion *encode2* vom Typ **char list * char list * string → string**, durchführt.
- f) Schreiben Sie eine SML-Funktion *decode2*, die die von *encode2* durchgeführte Verschlüsselung entschlüsselt.
- g) Entschlüsseln Sie den mit der Funktion *encode2* unter Verwendung der Schlüssel

$k_1 = [d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, a, b, c]$

$k_2 = [h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, a, b, c, d, e, f, g]$

verschlüsselten Text "xug hpq jbwlv qlxlv Jhky".

Aufgabe 10-3

Weihnachtsbaum

(keine Abgabe)

Schreiben Sie eine SML-Funktion *printX* : **int → unit**, die für ein Argument *n* einen Weihnachtsbaum mit *n* „Stockwerken“ der folgenden Form auf den Bildschirm zeichnet.

Für n=0:

```
*
^[_]^
```

Für n=1:

```
*
/.\
/,.,\
^[_]^
```

Für n=2:

```
*
/.\
/,.,\
/,.,.\
/,.,.,\
^^[_]^^
```

für n=3:

```
*
/.\
/,.,\
/,.,.\
/,.,.,\
/,.,.,.\
/,.,.,.,\
^^^[_]^^^
```

und entsprechend für größere *n*. Dabei darf das Innere des Baumes beliebig gestaltet werden, nur der Umriss muss wie angegeben aussehen.

Hinweis: Konstruieren Sie zunächst einen Text, und drucken Sie diesen mit *print* auf dem Bildschirm aus. Das Zeichen \ muss wegen seiner Sonderfunktion als \\ geschrieben werden.

Aufgabe 10-4**Listenfunktionen**

(1 + 2 + 1 + 2 Punkte)

- a) Schreiben Sie eine SML-Funktion *ordered* vom Typ **int list** → **bool**, so dass ein Aufruf *ordered l* genau dann *true* zurückgibt, wenn *l* bezüglich < geordnet ist, d.h. wenn für alle unmittelbar aufeinander folgenden Elemente *m, n* in *l* gilt: $m < n$.
- b) Schreiben Sie eine SML-Funktion *swap2*, die jeweils zwei aufeinander folgende Elemente einer Liste vertauscht. Falls die Liste eine ungerade Anzahl von Elementen hat, bleibt das letzte Element unverändert. Es gilt z.B.: *swap2*[1, 2, 3, 4, 5] liefert [2, 1, 4, 3, 5].
- c) Schreiben Sie eine rekursive SML-Funktion *listeq* vom Typ *"a list * "a list* → **bool**, die genau dann *true* zurückgibt, wenn die beiden als Argument übergebenen Listen elementweise gleich sind. (Zwei Listen vom Typ *"a list* lassen sich auch mit der Basisfunktion = vergleichen. In dieser Aufgabe soll aber aus Übungsgründen eine auf Listen spezialisierte Vergleichsfunktion implementiert werden.)
- d) Zwei Datenelemente *x, y* heißen *gleichartig bezüglich p*, wenn gilt $p(x, y) = true$. Schreiben Sie eine SML-Funktion *listeq1* vom Typ *('a * 'b → bool) * 'a list * 'b list* → **bool**, so dass ein Aufruf *listeq1(p, l₁, l₂)* genau dann *true* zurückgibt, wenn *l₁* und *l₂* elementweise gleichartig bezüglich *p* sind.

Aufgabe 10-5**„Bounty“ (Josephus-Problem)**

(6 Punkte)

In der antiken Seefahrt stand auf Meuterei die Todesstrafe. Die Angeklagten mussten sich in einem Kreis an Deck aufstellen, und jeder Zehnte wurde zum Tode verurteilt (und musste über Bord). Wir wollen herausfinden, welche Positionen bei solch einem Auszählen zur Verurteilung führen, wobei wir die Aufgabe dahin verallgemeinern, dass wir die Lösung nicht nur für den Fall berechnen, dass jeder zehnte Meuterer verurteilt wird, sondern allgemein für den Fall, dass jeder *d*-te Meuterer (für $d > 0$) verurteilt wird. Unsere Meuterer sind aufsteigend nummeriert, die Person, bei der zu zählen begonnen wird, trägt die Nummer 0.

Geben Sie eine SML-Funktion *bounty* an, so dass *bounty(n, k, d)* eine Liste der ganzen Zahlen (d.h. Personen) liefert, die nach der Verurteilung von *k* Personen übrig bleiben, wenn die Auszählung mit dem Intervall *d* durchgeführt und mit den *n* Personen 0, 1, . . . *n* - 1 begonnen wurde.

Bitte beachten Sie, dass wir Ihre Lösung nur dann korrigieren und bewerten können, wenn sie als *Textdatei* abgegeben wird. Als Lösung ist ein *lauffähiges* SML-Programm abzugeben. Nicht zur Lösung gehörende Bemerkungen sind zwischen die Kommentarzeichen (* und *) einzuschließen. Alle hiervon abweichenden Hausaufgaben werden mit 0 Punkten bewertet.

Abgabe: Dienstag, 7.1.2003, 12:00 Uhr.