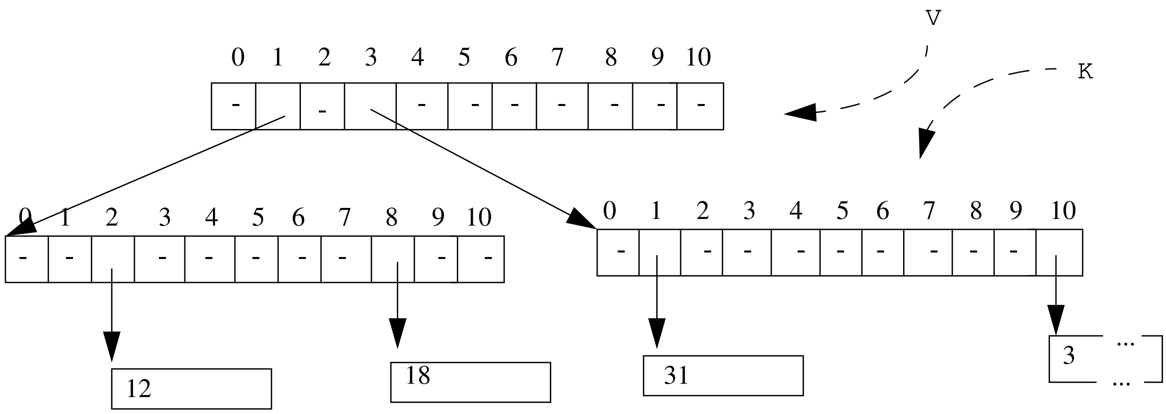
WS 95/96

Klausur zu
Algorithmen und Programmierung 3
Di. 13.2.96, 10.15 - 14.00

Mindestens 50% der Gesamtpunktzahl müssen erreicht werden.
Ergebnisse hängen ab 22.2.96 aus.
Wiederholungsklausur: Do. 11.4.96, 10 -14h

Aufgabe 1 (90 min, 20 P)
Eine Menge von Schlüsseln (Worten über einem endlichen Alphabet) kann mit Hilfe eines *Trie* genannten Vielwegbaums so implementiert werden, daß gleiche Anfangsstücke von Schlüsseln nur je einmal gespeichert werden. Hier geht es um die Implementierung in einer imperativen Programmiersprache (Modula2).
Ein *Trie* über dem Alphabet $\{a_1, a_2, ..., a_n\}$ ist ein Mehrwegbaum `T` der Ordnung `n` (jeder Knoten hat $0 \le r \le n$ Nachfolger). Die inneren Knoten sind Verweisknoten, die Blätter enthalten einen Schlüssel (und evtl. einen zugehörigen Datensatz).
Ein Verweisknoten $K=(P_1, P_2, ..., P_{n+1})$ enthält `n+1` Zeiger. $P_1, ..., P_n$ entsprechen jeweils einem Zeichen des Alphabets. P_{n+1} ist entweder NIL oder verweist auf ein Blatt mit dem Schlüssel, dessen Wurzelpfad im Vater `V` von `K` endet oder P_{n+1} = `NIL`. Wurzelpfaden entsprechen also Schlüsseln bzw. Schlüsselpräfixen (s. Beispiel).

Beispiel: Schlüsselmenge `S = {3,12,18,31}` über dem Alphabet `Z = {0,1,2,3,...,9}`
Beachten Sie, daß P_{n+1} von `K` auf ein Blatt mit dem Schlüssel `3` verweist. Der Wurzelpfad des Präfix `3` endet in `V`.

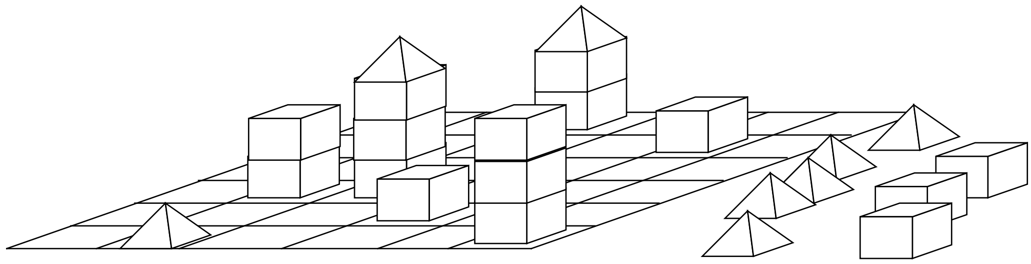


Gegeben ist das Definitionsmodul

```
DEFINITION MODULE Menge;
CONST maxLength = 21;

TYPE alphabet = ['A'..'Z'];

      KeyT = ARRAY[0..maxLength] OF alphabet;
     (* Schlüssel sind aus Großbuchstaben bestehende Zeichenketten *)
     (* Zeichenketten werden durch 0H abgeschlossen                *)

     KeysetT;
     (* ADT als opaker Typ *)

     PROCEDURE create(): KeysetT;
     PROCEDURE search(ks: KeysetT; k: KeyT) : BOOLEAN;
     (* Pre: der Schlüssel k genügt der Definition                  *)
     (* Effekt: keiner;  val: true, wenn k gefunden, false sonst    *)
     PROCEDURE insert(VAR ks: KeysetT; k: KeyT);
     (* Pre: der Schlüssel k genügt der Definition, ks enthält weniger als *)
     (* die maximale Anzahl darstellbarer Schlüssel                 *)
     (* Effekt: ks' = ks ∪ {k}                                      *)
END Menge.
```

Gesucht sind a) die Repräsentation von Mengen als Trie, b) die Abstraktionsfunktion und die konkrete (Repräsentations-) Invariante und c) die Implementierung der Operationen `create, search, insert`.
Hinweis: die den Schlüsseln zugehörigen Datensätze sollen ignoriert werden. Die Blätter enthalten nur je einen Schlüssel; innere Knoten und Blätter können einheitlich mit varianten Records repräsentiert werden.

Aufgabe 2 (60 min , 15 P)
Betrachten Sie die folgende kleine „Welt“:



Es gibt eine Grundfläche aus 6 x 6 Feldern und (beliebig viele) Bauklötze, die nur nach bestimmten Regeln auf der Grundfläche angeordnet werden dürfen. Betrachten Sie zunächst nur die würfelartigen Bausteine, dann gelten die folgenden Regeln:

1.) Ein Baustein muß genau auf einem Feld oder auf einem anderen Baustein liegen.
2.) Wo ein Baustein ist, kann kein anderer sein.

a) Spezifizieren Sie den abstrakten Datentyp “Klotzwelt”. Auf dem Datentyp sind die Operationen `erzeuge`, `lege` und `entferne` definiert:

> `erzeuge: -> Klotzwelt`
> Erzeugt eine leere Grundfläche.
>
> `lege: Klotzwelt x Position -> Klotzwelt`
> Legt einen Baustein an die angegebene Position in der Klotzwelt ab.
>
> `entferne: Klotzwelt x Position -> Klotzwelt`
> Entfernt einen Baustein an der angegebenen Position aus der Klotzwelt.

Geben Sie eine abstrakte Invariante an, die die oben genannten Einschränkungen des Zustandsraums der Klotzwelt beschreibt.
b) Implementieren Sie den abstrakten Datentyp “Klotzwelt” in Modula2. Definitionsmodul und Implementierung der Operation `lege` reicht. Vergessen Sie dabei nicht die konkrete Invariante, sowie die Vor-, Nach-bedingung und den Effekt von `lege` anzugeben. Sie können Implementierungen bekannter ADTen wie Listen, Stapel, Mengen usw. als gegeben annehmen und bei Bedarf importieren.
c) Neben würfelartigen Bausteinen gibt es pyramidenartige Bausteine, die sich dadurch unterscheiden, daß auf einen pyramidenartigen Baustein kein weiterer Baustein gelegt werden darf. Auf allen Bausteinen sind die Operationen `lege` und `entferne` definiert. Definitionsmodul reicht. Begründen Sie ihren Entwurf.

Aufgabe 3 (45 min, 10 P)
Ein Speicher kann als Abbildung von Speicheradressen auf Speicherinhalte modelliert werden. Bevor eine Zelle mit `store` bzw. `load` bearbeitet werden kann, muß sie mit `new` zugänglich („allokiert“) werden. Der folgende ADT soll zu einem gültigen Miranda-abstype ergänzt werden:

```
abstype storage *, address
      with
      emptystorage :: storage *
      new  :: * -> storage * -> (storage *, address)
      store :: * -> address -> ......
      load  :: storage * -> address -> ...
||    Repräsentation
address      == num
storage *    == (num -> *, num)
||    Funktionsdefinitionen
emptystorage = (undef,0)
      ....
```