Aufgabe 1

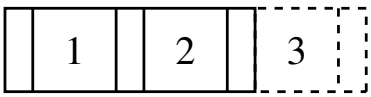
```
IMPLEMENTATION MODULE Zaehler;

   FROM Storage IMPORT ALLOCATE, DEALLOCATE;

   TYPE
      Zaehler = POINTER TO ZaehlerRec;
      ZaehlerRec =
         RECORD
            wert: CARDINAL;
            inkrement: CARDINAL;
         END;

   PROCEDURE Init(VAR zaehler: Zaehler; inkrement: CARDINAL);
      (* Initialisierung eines neuen Zaehlers,
         der zu Beginn den Wert 0 besitzt
      *)
   BEGIN
      NEW(zaehler);
      zaehler^.wert := 0;
      zaehler^.inkrement := inkrement;
   END Init;

   PROCEDURE Wert(zaehler: Zaehler) : CARDINAL;
      (* liefert den aktuellen Wert des Zaehlers zurueck
         und erhoeht den Wert des Zaehlers um den Inkrement-Wert,
         der bei Init mitgegeben worden ist
      *)
      VAR
         wert: CARDINAL;
   BEGIN
      wert := zaehler^.wert;
      INC(zaehler^.wert, zaehler^.inkrement);
      RETURN wert
   END Wert;

   PROCEDURE Wegwerfen(VAR zaehler: Zaehler);
      (* Elimination eines nicht mehr benoetigten Zaehlers *)
   BEGIN
      DISPOSE(zaehler);
   END Wegwerfen;

END Zaehler.
```

Aufgabe 2

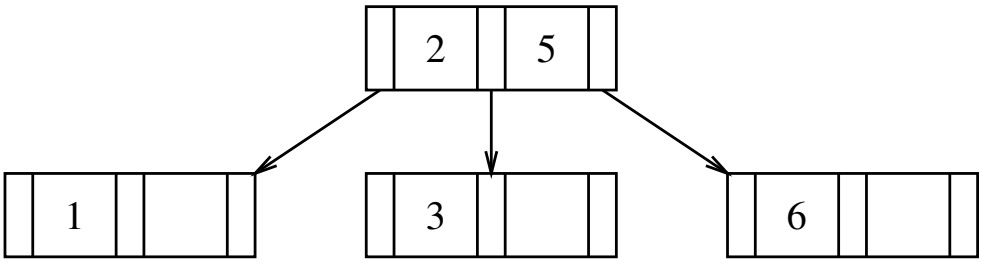
(*a*) [12, z] wird ganz unten rechts in den freien Slot neben [10, f] eingetragen.

(*b*)

Zuerst [2,t] unten links einfuegen:



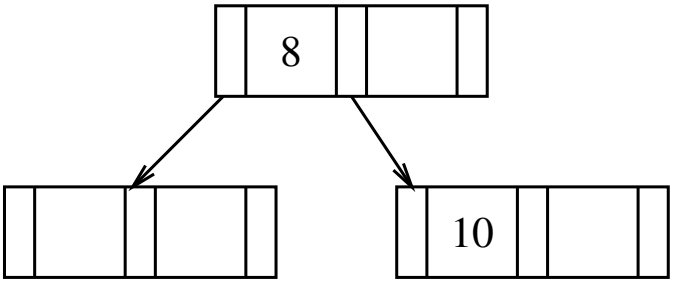
Wegen Ueberlauf muss gesplittet werden:



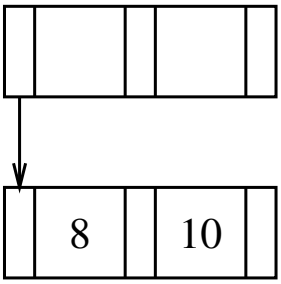
(*c*) Da [5, y] aus einer Schicht stammt, die nicht ganz unten ist, muß ein anderer Satz als Ersatz nach oben hochgezogen werden. Möglich sind [3, d] oder [6, c]. Wenn man [3, d] heraufzieht, sind keine weiteren Aktionen notwendig.

(*d*)

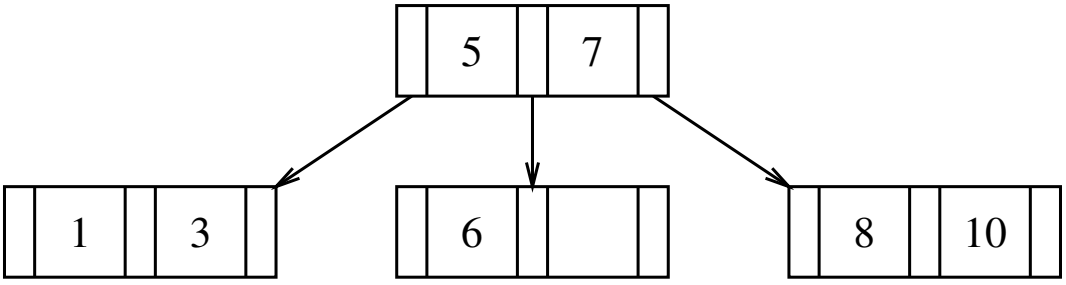
Zuerst die 8 hochziehen:



Wegen Unterlauf zusammenfassen:



Nochmals wegen Unterlauf zusammenfassen,
dabei erhalten wir eine neue Wurzel und die Hoehe des B-Baumes ist nur noch 2:



Aufgabe 3

(*a*)

0	Kurt, 28
1	
2	Alexander, 22
3	Anton, 33
4	Franz, 21
5	
6	Elisabeth, 18
7	
8	Anne, 27
9	
10	Ulrike, 23

(*b*)

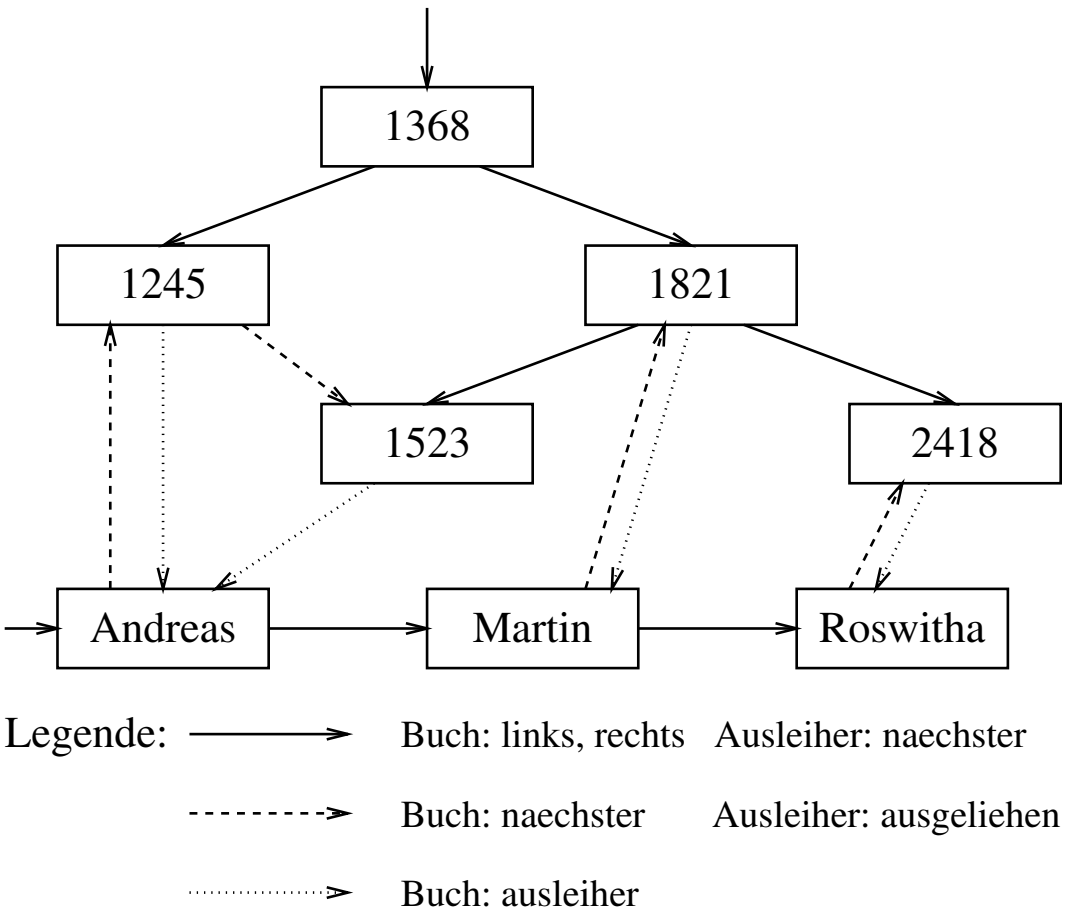
```
PROCEDURE Hash(name: Name) : HashIndex;
   (* liefert einen Hash-Wert fuer den angegebenen Namen zurueck;
      dabei wird vorausgesetzt, dass der Name nur aus Buchstaben besteht;
      der Unterschied zwischen Klein- und Großbuchstaben ist nicht
      signifikant
   *)
   VAR
      index: CARDINAL;
      wert: CARDINAL;
BEGIN
   wert := 0;
   index := 0;
   WHILE (index <= HIGH(name)) AND (name[index] # 0C) DO
      wert := wert * 16 + CAP(name[index]) - ORD("A");
      INC(index);
   END;
   RETURN wert MOD hashtabGroesse
END Hash;
```

Aufgabe 4

(*a*)

```
TYPE
   Signatur = CARDINAL;
   Autor = ARRAY [0..31] OF CHAR;
   Titel = ARRAY [0..79] OF CHAR;
   Name = ARRAY [0..31] OF CHAR;
   Ausleiher = POINTER TO AusleiherRec;
   Buch = POINTER TO BuchRec;
   BuchRec =
      RECORD
         signatur: Signatur; (* Schluessel *)
         autor: Autor;
         titel: Titel;
         links, rechts: Buch; (* nach der Signatur sortierter Binaerbaum *)
         ausleiher: Ausleiher; (* = NIL, wenn es bei uns steht *)
         naechster: Buch; (* Liste der Buecher, die einer ausgeliehen hat *)
      END;
   AusleiherRec =
      RECORD
         name: Name; (* Schluessel *)
         naechster: Ausleiher; (* sortierte lineare Liste der Ausleiher *)
         ausgeliehen: Buch; (* Liste der ausgeliehenen Buecher *)
      END;
```

(*b*)



Aufgabe 5

```
PROCEDURE BestimmeSortierteListe(baum: Baum);

   PROCEDURE Traverse(knoten: Knoten);
   BEGIN
      IF knoten # NIL THEN
         (* zuerst diejenigen aufsuchen, die nach uns kommen *)
         Traverse(knotenˆ.rechts);
         (* aktuellen Knoten vorne in die Liste einhaengen *)
         knotenˆ.naechster := baumˆ.liste;
         baumˆ.liste := knoten;
         (* und jetzt diejenigen aufsuchen, die vor uns kommen *)
         Traverse(knotenˆ.links);
      END;
   END Traverse;

BEGIN
   baumˆ.liste := NIL;
   Traverse(baumˆ.wurzel);
END BestimmeSortierteListe;
```

Aufgabe 6

(*a*)

```
PROCEDURE Anhaengen(liste: Liste; info: Info);
   (* lege ein neues Element an mit der gegebenen info-Komponente
      und haenge es an das Ende der Liste 'liste'
   *)
   VAR
      element: Element (* neues Element *)
BEGIN
   NEW(element); element^.info := info;
   element^.naechster := NIL;
   IF liste^.ende # NIL THEN
      liste^.ende^.naechster := element;
   ELSE
      liste^.anfang := element;
   END;
   liste^.ende := element;
END Anhaengen;
```

(*b*)

```
PROCEDURE Ausgabe(liste: Liste);
   VAR
      element: Element;
BEGIN
   element := liste^.anfang;
   WHILE element # NIL DO
      WriteString(element^.info); WriteLn;
      element := element^.naechster;
   END;
END Ausgabe;
```