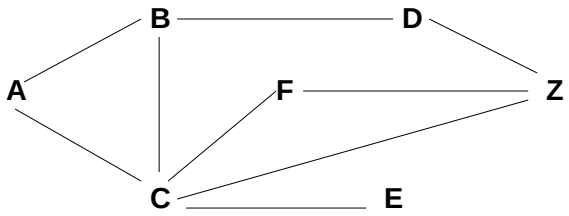
Name: *Matrikelnummer:*

Aufgabe 1: Suchalgorithmen **(12 Punkte)**

(a) (4 Punkte) Gegeben ist folgendes Streckennetz:



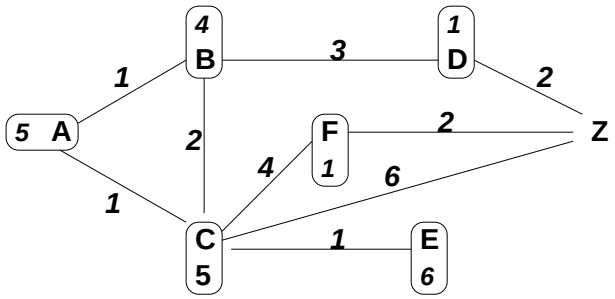
Zeigen Sie durch Handsimulation wie mit dem Breitensuch-Algorithmus ein Weg von A nach Z gefunden wird. Für die Reihenfolge, in der Nachfolger-Knoten besucht werden, sei eine **alphabetische Ordnung** vorgegeben
(A < B < C < D < E < F < Z).

(b) (1 Punkt) Was ist ein Zyklus und wie können Zyklen bei der Breitensuche erkannt und vermieden werden?

Punkte	

Name: *Matrikelnummer:*

(c) (4 Punkte) Nun sind zusätzlich Entfernungen zwischen den Orten (an den Kanten) und untere Schranken für die Restwegkosten (an den Knoten) gegeben:



Zeigen Sie durch Handsimulation wie mit dem A* Algorithmus für bewertete Kanten der Weg von A nach Z gefunden wird. Die Ordnung der Nachfolger-Knoten sei wieder alphabetisch.

(d) (1 Punkt) Was ist die Terminationsbedingung für A* und Branch-and-Bound?

Punkte	

Name: *Matrikelnummer:*

(e) (1 Punkt) In welcher Hinsicht ist Branch-and-Bound als Spezialfall von A* zu sehen?

(f) (1 Punkt) Mit welcher Speicherstruktur und welchem Abarbeitungsprinzip kann Tiefensuche realisiert werden?

Punkte	

Name: *Matrikelnummer:*

Aufgabe 2: Greedy-Algorithmen **(13 Punkte)**

(a) (1 Punkt) Was ist der Unterschied zwischen greedy-Lösbarkeit und greedy-Algorithmus?

(b) (10 Punkte) Ein Fabrikant möchte einige seiner Maschinen erneuern. Für jede Maschine sind die Leistung (Anzahl der produzierten Stücke pro Zeit) und die Betriebskosten (pro Zeit) bekannt. Erneuert werden sollen die unwirtschaftlichsten Maschinen. Die Auswahl soll durch einen *greedy*-Algorithmus erfolgen. Entsprechend wird als *greedy*-Kriterium der Quotient $Kosten/Leistung$ verwendet.

Gegeben seien n Maschinen, wovon m erneuert werden sollen ($m < n$). Die Maschinen werden durch folgenden Datentyp repräsentiert:

```
TYPE Maschtyp = RECORD
                  Kennung   : INTEGER;
                  Kosten    : REAL;
                  Leistung  : REAL;
                  Erneuern  : BOOLEAN;
                END;

TYPE Maschinen = ARRAY[1..n] OF Maschtyp; (* n als Konstante vor-
gegeben *)
```

Gesucht ist eine (MODULA-2) Prozedur

`PROCEDURE erneuere (VAR AlleMasch:Maschinen)`,

die aus den n vorhandenen Maschinen diejenigen m Maschinen auswählt, die bevorzugt erneuert werden sollen. Dabei sei `AlleMasch` bereits vorbelegt (Kennung, Leistung und Kosten sind gegeben; Erneuern ist für alle Maschinen auf `FALSE` gesetzt) und `m` und `n` seien als Konstanten vorgegeben. HINWEIS: Für den *greedy*-Algorithmus sollen die Daten **nicht** vorsortiert werden!

Punkte	

Name: *Matrikelnummer:*

```
PROCEDURE erneuere (VAR AlleMasch:Maschinen);
```

```
END erneuere;
```

Punkte	

Name: *Matrikelnummer:*

(c) (2 Punkte) Gegeben ist das Rucksack-Problem:
Ein Dieb will möglichst viel Geld aus einem Münz-Tresor stehlen. Die Münzen im Tresor haben alle dasselbe Gewicht, aber unterschiedliche Wertigkeiten. Der Dieb hat einen Rucksack dabei, in den maximal n Münzen passen.
Was ist das greedy-Kriterium für dieses Problem?

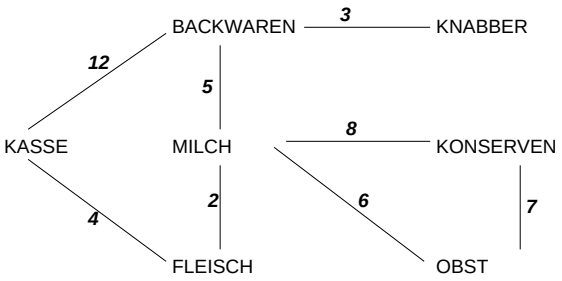
Punkte	

Name: *Matrikelnummer:*

Aufgabe 3: Graphalgorithmen **(13 Punkte)**

(a) (1 Punkt) Was repräsentiert der aufspannende Baum eines Graphen?

(b) (10 Punkt) In einem Supermarkt soll die Anordnung der Waren optimiert werden. Aus diesem Grund wird zunächst für verschiedene Produktsorten (sowie die Kasse) erhoben, wie weit die Wege zwischen diesen Produkten sind. Das Ergebnis der Analyse ist im folgenden Graph abgebildet:



Implementieren Sie den Dijkstra-Algorithmus (in MODULA-2) um zu ermitteln, wie hoch die minimalen Wegekosten von einer fest vorgegebenen Position zu allen anderen Positionen sind.

Vorgaben:

```
CONST MatrixDimension = 7;
      UNENDLICH = 99;

TYPE  Matrix = ARRAY[1..MatrixDimension],[1..MatrixDimension] OF INTEGER;
      ErgebnisArray = ARRAY[1..MatrixDimension] OF INTEGER;
```

Die Knotenbezeichner (“Kasse”, “Backwaren”, etc.) werden dabei durch Zahlen $\{1 \ldots MatrixDimension\}$ angegeben. Die Kosten sind als `INTEGER` vorgegeben.

Gesucht ist die Prozedur `PROCEDURE Dijkstra (VAR kMatrix: Matrix; VAR Ergebnis: ErgebnisArray; startKnoten: INTEGER)`; wobei `kMatrix` bereits initialisiert sei (Adjazenzmatrix des Graphen).

Punkte	

Name: *Matrikelnummer:*

```
PROCEDURE Dijkstra (VAR kMatrix: Matrix; VAR Ergebnis:
ErgebnisArray; startKnoten: INTEGER);
```

```
END Dijkstra;
```

Punkte	

Name: *Matrikelnummer:*

(c) **(1 Punkt)** Wie hoch sind die minimalen Wegekosten von der Kasse zu allen angegebenen Waren?

Backwaren –

Milch –

Fleisch –

Knabber –

Konserven –

Obst –

(d) **(1 Punkt)** Was ist der Aufwand des Dijkstra-Algorithmus?

Punkte	

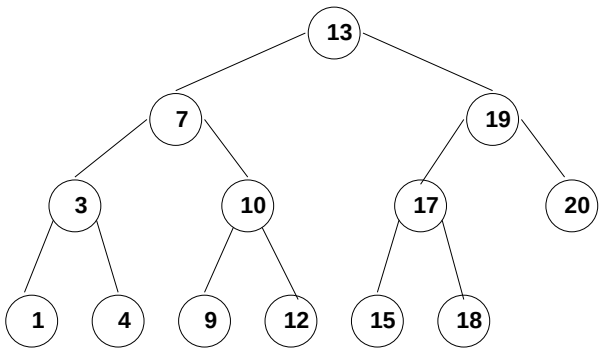
Name: *Matrikelnummer:*

Aufgabe 4: Dynamische Datenstrukturen **(12 Punkte)**

(a) (1 Punkt) Welche Aufwandsklasse hat die Suche nach einem Eintrag in einem maximal ausgewogenen Binärbaum im Vergleich zur Suche in einer ungeordneten Liste (bei annähernd gleichen Zugriffswahrscheinlichkeiten auf die Daten)?

(b) (1 Punkt) Was ist ein c-höhen-balancierter Baum?

(c) (3 Punkte) Gegeben ist der unten abgebildete 1-höhenbalancierte, geordnete Binärbaum (angegeben sind nur die Schlüssel). In diesen Baum soll ein neues Element mit Schlüssel "11" eingefügt werden, so daß wieder ein 1-höhenbalancierter, geordneter Binärbaum entsteht. Zeichnen Sie den resultierenden Baum auf.



Punkte	

Name: *Matrikelnummer:*

(d) (2 Punkte) Wie kann beim offenen Hashing vermieden werden, daß alle Schlüssel dem selben Indexplatz zugewiesen werden?

Punkte	

(e) **(3 Punkte)** Die Verwaltung einer Studierenden-Kartei soll von geschlossenem auf offenes Hashing umgestellt werden. Zukünftig soll der Zugriff über die erste Ziffer der fünfstelligen Matrikelnummer erfolgen. Bisher wurde die Hash-Tabelle durch folgende Datenstruktur realisiert:

```
CONST
  N = 10000; (* Laenge der Hash-Tabelle *)

TYPE
  ziffer = 0..9;

  daten = RECORD
            nachname: String;
            vorname: String;
            matnummer: ARRAY [1..5] OF ziffer;
            semesterzahl: CARDINAL;
          END;

  element = RECORD
              marke: BOOLEAN;  (* FALSE bei geloeschtem Eintrag *)
              student: daten;
            END;

  tabelle = ARRAY [0..N-1] OF element;
```

Modifizieren Sie die Datenstruktur so, daß offenes Hashing realisiert werden kann. (Notation wieder in MODULA-2, **nur** Datenstruktur, nicht Hash-Funktion)

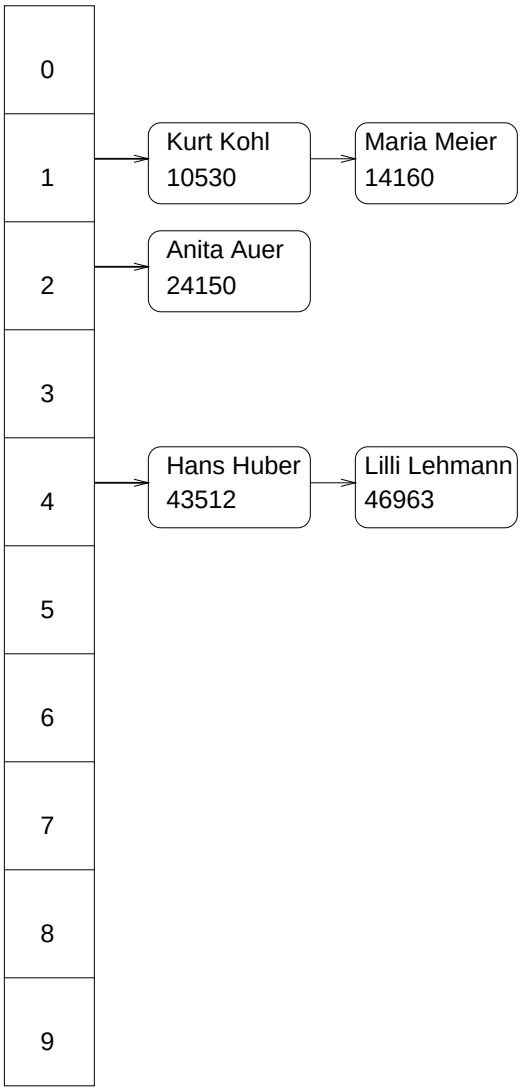
Punkte	

Name: *Matrikelnummer:*

(f) (2 Punkte) In einer Mini-Datenbank sind die Daten von fünf Studierenden mit offenem Hashing gespeichert. Geben Sie an, wie diese Daten bei geschlossenem Hashing in einer Tabelle mit 10 Speicherplätzen abgelegt waren:

- Eintragsreihenfolge: nach dem ersten Buchstaben des Nachnamens; bei Kollision wird der nächste freie Speicherplatz verwendet.
- Schlüssel k: Quersumme der 5-stelligen Matrikelnummer
- Hashfunktion: Divisionsmethode: $h'(k) = k \ mod \ N$.

Tragen Sie die Werte für h' und die Matrikelnummern in die vorgegebene Tabelle ein.



j	h'(k)	Mat-Nr.
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

Punkte	