Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Jörg Derungs
Matthias Müller

Institut für Computersysteme
Jürg Gutknecht
Matteo Corti

1. Vordiplom D-INFK

Prüfung Informatik I + II

3. Oktober 2002

Name, Vorname: _____

Stud.-Nummer: _____

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die untenstehenden Hinweise gelesen und verstanden habe.

Unterschrift: _____

Hinweise:

- Ausser einem Wörterbuch dürfen Sie keine Hilfsmittel verwenden.
- Bitte schreiben Sie Ihre StudentInnen-Nummer auf **jedes** Blatt.
- Melden Sie sich bitte **sofort**, wenn Sie sich während der Prüfung in irgendeiner Weise bei der Arbeit gestört fühlen.
- Bitte verwenden Sie für jede Aufgabe ein neues Blatt. Pro Aufgabe kann nur eine Lösung angegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen werden.
- Bitte schreiben Sie **lesbar**! Wir werden nur bewerten, was wir lesen können.

Viel Erfolg!

Stud.-Nummer: _____

Informatik I:

Aufgabe	1	2	3	4	**Σ**
Mögl. Punkte	9	9	9	9	36
Punkte					

Informatik II:

Aufgabe	5	6	7	8	**Σ**
Mögl. Punkte	10	7	9	10	36
Punkte					

9 P **AUFGABE 1** *(Informatik I)*:

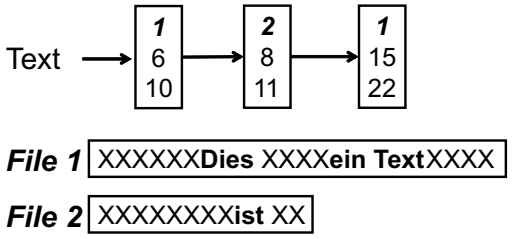
Wir betrachten einen mit `M` × `N` Platten belegten Boden. Ausserdem nehmen wir an, dass sich eine Küchenschabe zu Beginn der Beobachtung in einer Ecke befinde und sich während jeder Zeiteinheit zufällig auf eine benachbarte Platte begebe. Erstellen Sie ein Oberonprogramm, welches den Gang der Küchenschabe simuliert und die Zeit berechnet, die vergeht, bis die Küchenschabe jede Platte mindestens einmal betreten hat. Nehmen Sie dabei an, dass ein Zufallszahlengenerator existiere, der bei jedem Aufruf `Random.Next()` eine zufällige, nichtnegative `INTEGER` Zahl liefert).

9 P **AUFGABE 2** *(Informatik I)*:

Eine *Unordnung* sei eine Permutation p von $\{0, ..., n-1\}$, in welcher kein Element am richtigen Platz steht, d. h. dass für alle `i`, `0` $\leq$ `i` $\leq$ `n-1` , `p[i]` $\neq$ `i` gilt. Schreiben Sie ein effizientes Branch-and-Bound Programm, welches alle Unordnungen von $\{0, ..., n-1\}$ konstruiert.

9 P **AUFGABE 3** *(Informatik I)*:

Wir denken uns einen Text als Liste von *Pieces* dargestellt, wobei ein Piece durch ein File sowie Anfangs- und Endposition innerhalb des Files dargestellt sei (siehe separate Figur):




```
TYPE
  Piece = POINTER TO RECORD
    next: Piece;
    file: INTEGER; (*file number*)
    beg, end: INTEGER
 END;
```

a) Erstellen Sie eine Oberonprozedur, welche zu einer gegebenen Position `a` innerhalb eines Textes das entsprechende Piece und die Position innerhalb desselben eruiert.

b) Erstellen Sie nun eine Oberonprozedur, welche einen Ausschnitt [a, b] aus einem Text lscht. Berücksichtigen Sie die Spezialfälle.

9 P **AUFGABE 4** *(Informatik I)*:

Wir betrachten ein hierarchisches Filesystem, dessen Verzeichnis durch einen Mehrwegbaum mit folgendem Knotentyp dargestellt sei:

```
TYPE
  Item = POINTER TO RECORD
    down, next: Item;
    isdir: BOOLEAN; (*true <=> is directory*)
    name: ARRAY 20 OF CHAR
 END;
```

a) Erstellen Sie ein Oberonprogramm, welches nachprüft, ob das Verzeichnis tatsächlich eine Baumstruktur besitzt (d. h. dass kein Item in mehr als einem (Unter-)Verzeichnis enthalten ist).

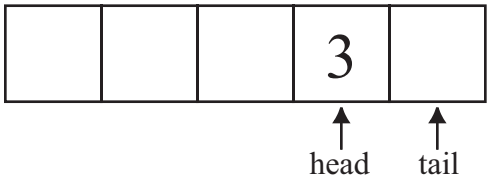
b) Erstellen Sie dann ein Oberonprogramm, welches das Verzeichnis ausdruckt und zwar so, dass jeder Name auf eine eigene Zeile geschrieben wird, und dass die Namen der in einem Verzeichnis enthaltenen Items gegenüber diesem um eine Position eingerückt geschrieben werden.

AUFGABE 5 *(Informatik II)*:

In dieser Aufgabe sollen Sie **nur** die Ergebnisse angeben. Diese können Sie direkt bei den Aufgaben notieren. Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung Informatik II verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch andere Methoden benutzen, müssen Sie diese **kurz** soweit erklären, dass Ihre Ergebnise verständlich und nachvollziehbar sind.
Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

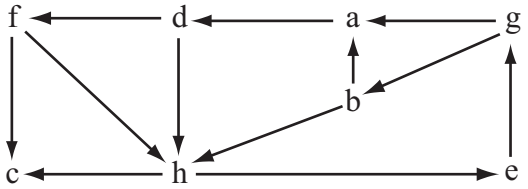
1 P

a) Gegeben ist ein zyklischer Buffer (queue), der nur ein Element (3) enthält. Wie sieht der Buffer und die Zeiger aus, nachdem ein Element 8 eingefügt und dann zweimal das vorderste Element entfernt wurde? Hinweis: Head zeigt immer auf das vorderste und Tail auf die Stelle hinter dem letzten Element.



1 P

b) Der folgende gerichtete Graph wird mit Tiefensuche traversiert. Die Suche startet beim Knoten 'a'. Geben Sie eine Reihenfolge an, in der die Knoten erreicht werden können.



1 P

c) Geben Sie für untenstehende Funktionen eine **Reihenfolge** an, so dass folgendes gilt: Wenn Funktion f links von Funktion g steht, so gilt $f \in O(g)$.

Beispiel: Die drei Funktionen n^3, n^7, n^9 sind bereits in der entsprechenden Reihenfolge, da $n^3 \in O(n^7)$ und $n^7 \in O(n^9)$ gilt.

- n^5
- n MOD 5
- 5^n
- $5/n$
- $n + 5$
- $\log_5 n$
- $n \cdot 5$

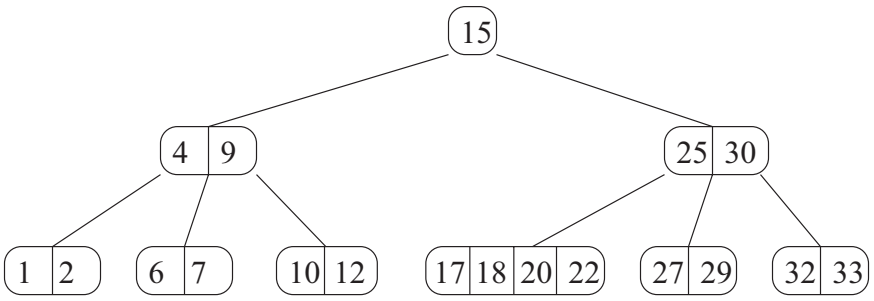
1 P d) Markieren Sie in der folgenden Liste genau die Aussagen, die korrekt sind:

- ☐ $n^5 - n \in O(n^4)$
- ☐ $\log_3 n + \log_2 n \in O(\log_2 n)$
- ☐ $\sum_{i=1}^{n}(ni) \in \Omega(n^3)$
- ☐ $(n+1)! \in \Theta(n!)$

1 P e) Zeichnen Sie den binären Suchbaum, dessen Knoten in preorder-Reihenfolge traversiert diese Zahlenfolge ergibt:

$5, 4, 2, 1, 3, 9, 6, 8, 7, 11, 10, 12$

1 P f) Zeichnen Sie den Baum, der entsteht, wenn im folgenden B-Baum der Ordnung 5 zuerst der Schlüssel 21 eingefügt und danach der Schlüssel 7 gelöscht wird.

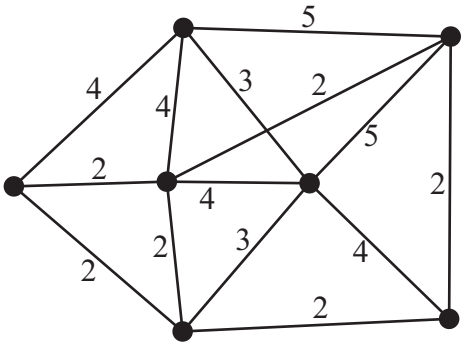


1 P g) In der folgenden Tabelle ist ein Heap in der üblichen impliziten Form gespeichert:

[3,4,7,5,11,9,13,8,29,12,15,10]

Wie sieht die Tabelle aus, nachdem das Element 2 hinten angefügt und die Heap-Bedingung wiederhergestellt wurde?

1 P h) Markieren Sie im untenstehenden gewichteten Graphen die Kanten eines **minimalen** Spannbaums.



1 P i) Geben Sie die Laufzeiten in Oh-Notation für die Suche in den folgenden Datenstrukturen an:

	best case	worst case
Sortierte verkettete Liste		
Sortierter Array		
AVL-Baum		

1 P j) Wieviele Schlüsselvergleiche benötigt eine mit Transpose-Regel selbstanordnende Liste insgesamt, um nacheinander die Elemente 3,1,1,1 zu finden?

5	8	1	3	4	7

AUFGABE 6 *(Informatik II)*:

2 P a) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 2T(\frac{n}{2}) + 2n & n > 1 \\ 0 & n = 1 \end{cases}$$

Beweisen Sie mit vollständiger Induktion, dass $2n \log_2(n)$ eine geschlossene Formel für $T(n)$ ist. *Hinweis:* Sie können annehmen, dass n eine Potenz von 2 ist.

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 4T(\frac{n}{2}) - 5 & n > 1 \\ 3 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) Formel für $T(n)$ an und beweisen Sie diese.

Hinweise: (1) Sie können annehmen, dass n eine Potenz von 2 ist. (2) $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$

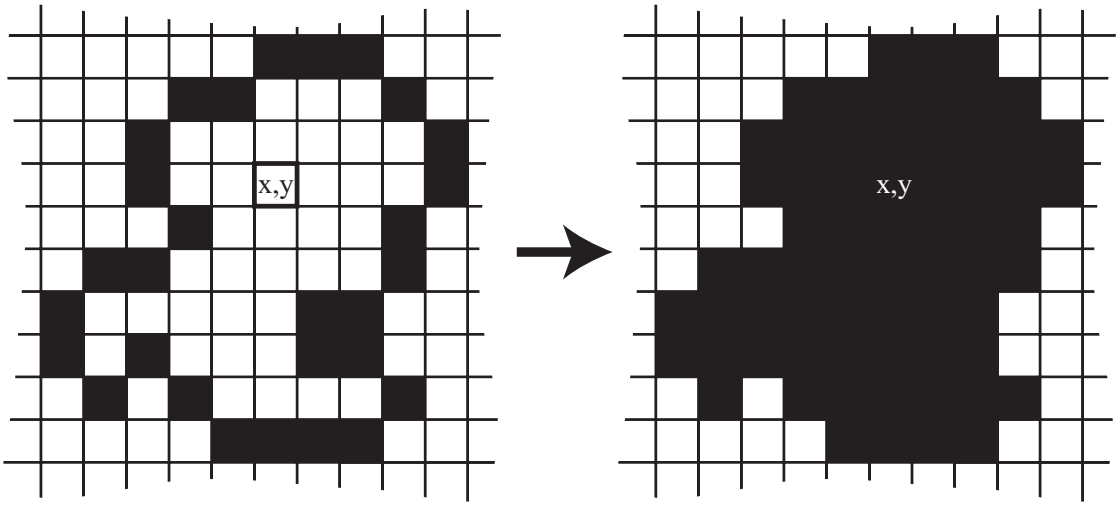
1 P c) Geben Sie eine möglichst genaue asymptotische Laufzeit für folgenden Algorithmus an (in Oh-Notation):

```
i := n;
WHILE i > 1 DO
    i := i DIV 2;
END
```

1 P d) Geben Sie eine möglichst genaue asymptotische Laufzeit für folgenden Algorithmus an (in Oh-Notation):

```
i := n;
WHILE i > 1 DO
    j := 1;
    WHILE j < i DO
        j := j + 1;
    END;
    i := i DIV 2;
END
```

AUFGABE 7 *(Informatik II)*:
Gegeben sei ein unendlich grosses Schachbrett, dessen Zellen beliebig schwarz oder weiss eingefärbt sind. Wir möchten nun, beginnend bei der Zelle (x,y), alle weissen zusammenhängenden Zellen schwarz einfärben (wie die Funktion *flood fill* in Zeichnungsprogrammen). Zwei Zellen sind zusammenhängend, wenn sie eine gemeinsame Kante besitzen.
Hinweis: Wir nehmen an, dass der einzufärbende Bereich nicht unendlich gross ist.



3 P a) Beschreiben Sie (in Pseudo-Code) eine rekursive Prozedur FloodFill(x,y: INTEGER), die alle zusammenhängenden weissen Zellen, beginnend bei (x,y), schwarz einfärbt. Das Schachbrett ist als unendlich grosser Array c[x,y] of (white, black) gegeben.

3 P b) Sei n die Anzahl der zusammenhängenden weissen Zellen, beginnend bei Zelle (x,y). Geben Sie die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von n im besten und im schlechtesten Fall an (mit Begründung).

3 P c) Den Speicherbedarf messen wir als die maximale Rekursionstiefe (Stackgrösse) während des Füllvorgangs. Welchen asymptotischen Speicherbedarf hat Ihr Algorithmus in Abhängigkeit von n im besten und im schlechtesten Fall? Zeichnen Sie für beide Fälle ein jeweiliges Schachbrettmuster, das den Fall erzeugt.

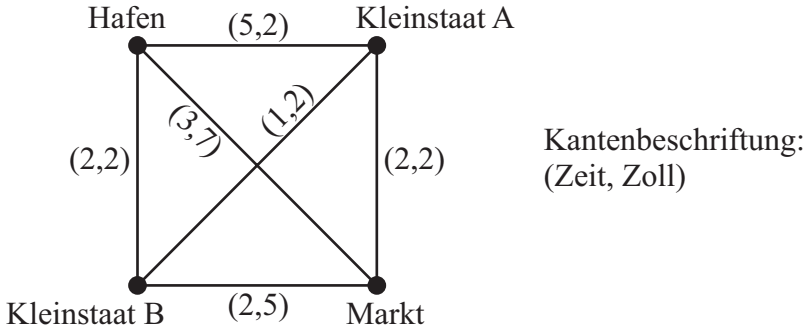
AUFGABE 8 *(Informatik II)*:
Ein Fischhändler will seine Ware vom Hafen zum Markt bringen. Dazu muss er ein Gebiet mit vielen Kleinstaaten durchqueren. An jeder Grenze muss er natürlich Zoll bezahlen.
Da er ein guter Händler ist, will er die Route so wählen, dass er möglichst wenig Zoll zahlen muss. Andererseits muss er in einer bestimmten Zeit am Markt sein, damit seine Fische nicht verderben.

Die Anzahl der Staaten (inklusive Hafen und Markt) ist N. Die Staaten werden als Zahlen codiert, wobei '0' der Hafen ist und '$N-1$' der Markt. Die Reisezeiten und die Zölle sind beide als `ARRAY N, N OF INTEGER` gegeben.

Beispiel:

Reisezeit	Hafen	A	B	Markt
Hafen	0	5	2	3
A	5	0	1	2
B	2	1	0	2
Markt	3	2	2	0

Zoll	Hafen	A	B	Markt
Hafen	0	2	2	7
A	2	0	2	2
B	2	2	0	5
Markt	7	2	5	0



Bei einer maximalen Zeit von 6 Stunden führt der günstigste Weg vom Hafen über B und dann A zum Markt. Die Zölle kosten 6 Goldmünzen. Hat der Händler jedoch 7 Stunden Zeit, dann führt der günstigste Weg vom Hafen über A zum Markt, und die Zölle kosten 4 Goldmünzen.

4 P a) Schreiben Sie einen rekursiven Algorithmus, der die Gesamtkosten für alle Zölle auf dem günstigsten Weg bestimmt. Die Reisezeit darf höchstens `T` sein.

Bestimmen Sie die asymptotische Laufzeit Ihres Algorithmus, gemessen in N (der totalen Anzahl der Staaten).

4 P b) Entwerfen Sie einen Algorithmus nach dem Muster der dynamischen Programmierung, der die Gesamtkosten für alle Zölle auf dem günstigsten Weg berechnet. Wie oben darf die Reisezeit höchstens T sein. Geben Sie auch die Induktionsgleichung an, und wo in der Tabelle das Resultat steht.

Bestimmen Sie die asymptotische Laufzeit Ihres Algorithmus, gemessen in N.

2 P c) Beschreiben Sie in Worten, wie aus der Lösung gemäss der dynamischen Programmierung die entsprechende Route durch Rückverfolgen gefunden werden kann.

Bestimmen Sie die asymptotische Laufzeit für das Finden der Route, gemessen in N.