Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Dr. K. Simon

1. Vordiplom Abt. IIIA

Informatik I

Frühling 2000

Freitag, 3. März 2000

Name: _____

Vorname: _____

Legi-Nummer: _____

Unterschrift: _____

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl	Visum
1	22		
2	18		
3	31		
4	26		
5	22		
Total	119		
Note			

Allgemeine Hinweise

- Zugelassene Hilfsmittel: **keine**.
- Legen Sie zu Anfang der Prüfung Ihre Legi neben sich auf den Tisch.
- Die Prüfung besteht aus 5 Aufgaben. Kontrollieren Sie, ob Sie alle Aufgaben erhalten haben.
- Verwenden Sie für jede Aufgabe ein separates Blatt.
- Schreiben Sie auf jedes Blatt Ihre Legi-Nummer (und nur diese!).
- Schreiben Sie deutlich, und benützen Sie keinen Bleistift.
- Pro Aufgabe darf höchstens ein gültiger Lösungsversuch abgegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen sein.
- In jeder Aufgabenstellung ist eine Lösungsstruktur angegeben. Nichtbeachten der Lösungsstruktur führt zu Punktabzug.
- Es empfiehlt sich *unbedingt* zuerst alle Fragen durchzulesen.
- Abzugeben sind das vollständig ausgefüllte Deckblatt, die Aufgabenblätter und Ihre Lösungen.

Aufgabe 1: C++-Syntax, Schleifen, Funktionsdeklaration *(22 Punkte)*

a) In dieser Aufgabe geht es um die Syntax von C++-Programmen. Wir testen mit dieser Aufgabe, ob Sie in der Lage sind zu entscheiden, ob einfache C++-Programme Fehler enthalten und wie diese zu bewerten sind.

Aufgabe Ordnen Sie jedem Programmteil i-iv genau eine korrekte Aussage A,B,C oder D zu. Begründen Sie Ihre Auswahl.

Die Aussagen

A) Der Programmteil wird *fehlerlos kompiliert* und *läuft fehlerfrei*.
B) Das *Kompilieren* des Programmteiles wird durch eine Fehlermeldung *abgebrochen*.
C) Die *Ausführung* des Programmteiles wird durch eine Fehlermeldung *unterbrochen*.
D) Die *Ausführung* des Programmteiles *dauert unendlich lange*.

Lösungsstruktur Wir erwarten pro Programmteil genau eine Klassifizierung in Form eines Grossbuchstabens (A,B,C,D) sowie eine Begründung in ein bis zwei Sätzen. Im Fall A geben Sie die Auswirkungen der Programmteiles auf die Variable `a` an.

Beispiel Der Programmteil

```
1   int a;
2
3   a= 5;
4   if (3 = a) {
5           a= a+1;
6   } else {
7           a= a*2;
8   }
```

führt zur Klassifizierung B. Begründung: In Zeile 4 enthält die if-Anweisung statt einer Bedingung eine Zuweisung zu einer Konstanten.
Die Lösung kann auf dem Prüfungsblatt abgegeben werden.

Massstab Pro korrekter Klassifizierung: 1 Punkt. Prof korrekter Begründung: 1 Punkt. Gesamte Unteraufgabe: 8 Punkte.

Die Programmteile

i)
```
1   double a;
2
3   a= 2.0;
4
5   while (a < 3.0) {
6           a= a/2.0;
7   } else {
8           a= a*2.0;
9   }
```

ii)
```
1   double a;
2
3   a= 2.0;
4
5   if (a < 3.0) {
6           a= a/2.0;
7   } else {
8           a= a*2.0;
9   }
```

iii)
```
1   double *a;
2
3   a= 0;
4   *a= 2.0;
5
6   if (*a < 3.0) {
7           *a= *a/2.0;
8   } else {
9           *a= *a*2.0;
10  }
```

iv)
```
1   double a;
2   int i;
3
4   a= 2.0;
5
6   for (i= 2; i>=0; i=i/2) {
7           a= a/2.0;
8   }
```

$\Longrightarrow$

b) In der Vorlesung haben Sie verschiedene Arten von Schleifen kennengelernt. In den Übungen haben Sie gelernt, wie man Schleifentypen ineinander überführen kann. Diese Kenntnis wird in dieser Teilaufgabe geprüft.

Aufgabe Formulieren Sie die folgenden Programmteile jeweils mit den beiden anderen Schleifenvarianten so, dass der Effekt bezüglich der Variablen `a` derselbe ist.

Die verwendeten Variablen sind dabei wie folgt deklariert:

```
const int n= 100;
double a[n];
int i;
```

Lösungsstruktur Wir erwarten uns pro Programmteil je zwei Umformulierungen in die beiden anderen Schleifentypen. Gelingt Ihnen das nicht, so erwarten wir dafür eine Begründung in ein bis zwei Sätzen.

Massstab Pro korrekter Umformulierung bzw. Begründung: 1 Punkt. Gesamte Unteraufgabe: 6 Punkte.

Die Programmteile

i)
```
1   for (i= n-1; i>0; i--) {
2     a[i]= a[i-1];
3   }
```

ii)
```
1   i= 0;
2   while ((i<n) &&
3          (a[i]<=a[i+1]))
4     i++;
```

iii)
```
1   a[0]= 10;
2   do {
3     a[0]= a[0]/2.0;
4   } while (a[0]>=0);
```

c) In der Vorlesung haben Sie verschiedene Typen für Variablen in C++ kennengelernt. Sie haben gelernt Funktionen und Prozeduren zu deklarieren. Diese Aufgabe testet, wie gut Sie damit umgehen können.

Aufgabe Wir geben Ihnen zwei unvollständige Funktions-/Prozedurdeklarationen vor. Sie sollen in dieser Aufgabe die Köpfe mit Typangaben derart vervollständigen, dass sie korrekt kompiliert werden und lauffähig sind.

Lösungsstruktur Als Lösung abzugeben ist nur der vervollständigte Prozedur-/Funktionskopf. Geben Sie ausserdem an, ob es sich bei der vervollständigten Deklaration um eine Funktion oder eine Prozedur handelt.

Beispiel Gegeben ist folgende Deklaration:

```
1   ... s(...x)
2   {
3     return(exp(x));
4   }
```

Es handelt sich hier um eine Funktion. Der vervollständigte Funktionskopf ist:

```
double s(double x)
```

Massstab Pro richtigem Typ: 1 Punkt. Richtige Klassifizierung Funktion/Prozedur: 1 Punkt. Gesamte Unteraufgabe: 8 Punkte.

Die Deklarationen

i)
```
1   ... f(... phi, ... b)
2   {
3     if (*b < 'a') {
4       return(phi<=2.0);
5     } else {
6       return(phi>2.0);
7     }
8   }
```

ii)
```
1   ... g(... s, ... n)
2   {
3     int i;
4     for (i=0; i<=n; i++)
5       s[i]= i%2*3;
6   }
```

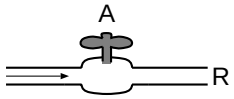
Aufgabe 2: Logik *(18 Punkte)*

a) In der Vorlesung haben Sie logische Ausdrücke und ihre Verknüpfungen sowie mehrere Notationen zu deren Darstellung kennenglernt. In dieser Aufgabe sollen Sie zeigen, ob Sie in der Lage sind, einfache logische Ausdrücke aus einem Sachverhalt herzuleiten.

Aufgabe Gegeben sind Rohrsysteme mit Ventilen. Bei jedem Rohrsystem fliesst von der linken Seite eine Flüssigkeit hinein. Sie sollen nun für jedes Rohrsystem einen logischen Ausdruck entwickeln, der anhand der Ventilstellungen bestimmt, ob auf der rechten Seite Flüssigkeit herausfliesst ($R =$ **true**) oder nicht ($R =$ **false**). Dabei gelten für jedes Ventil folgende Wahrheitswerte: **true** Ventil geschlossen, **false** Ventil geöffnet.

Lösungsstruktur Pro Rohrsystem erwarten wir einen logischen Ausdruck in C++-Notation oder mathematischer Notation nach folgendem Beispiel:

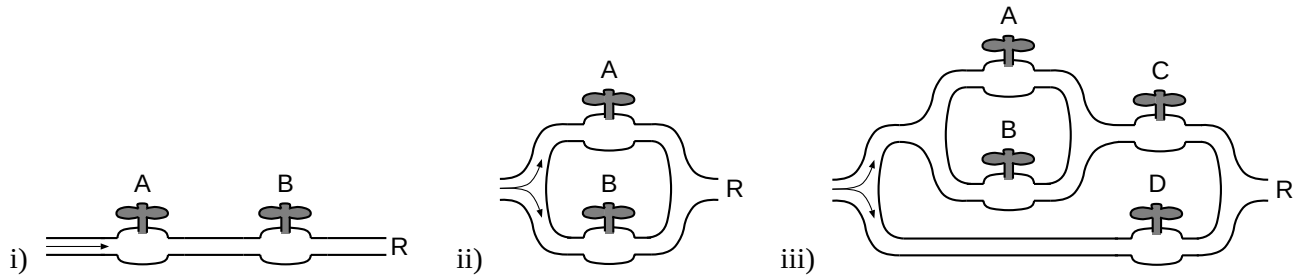
Beispiel



Der logische Ausdruck zu diesem Rohrsystem lautet: `R= !A`

Massstab Pro richtigem logischen Ausdruck: 2 Punkte. Gesamte Unteraufgabe: 6 Punkte.

Die Rohrsysteme



b) Diese Aufgabe hat das Ziel zu prüfen, ob sie in der Lage sind, einen logischen Ausdruck auszuwerten und ihn zu vereinfachen.

Aufgabe Gegeben ist folgender logischer Ausdruck in C++-Notation:

```
R= (!A && !C) || (A && !C) || (!A && !B) || (A && !B)
```

i) Stellen Sie eine Wahrheitstabelle zu diesem Ausdruck auf.

ii) Vereinfachen Sie den Ausdruck so weit als möglich aufgrund von Erkenntnissen aus der Wahrheitstabelle oder durch algebraische Umformungen.

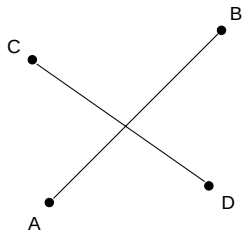
Lösungsstruktur Abzugeben ist einerseits eine vollständige Wahrheitstabelle. Andererseits erwarten wir eine Herleitung eines vereinfachten Ausdruckes entweder durch algebraische Umformungen des gegebenen Ausdruckes oder durch eine Prosabeschreibung in zwei bis drei Sätzen. Ausserdem erwarten wir die Angabe des vereinfachten Ausdruckes in C++- oder mathematischer Notation.

Massstab Pro korrekter Zeile der Wahrheitstabelle: 1 Punkt. Herleitung des vereinfachten Ausdruckes: bis zu 4 Punkte. Gesamte Unteraufgabe: 12 Punkte.

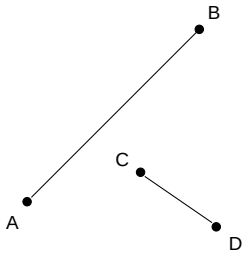
Aufgabe 3: Algorithmik: Geometrie *(31 Punkte)*

Vor allem in den Schnellübungen hatten sie die Gelegenheit, Algorithmen zu entwickeln und Datenstrukturen zu entwerfen. In allen Unteraufgaben wird die Fähigkeit geprüft, selbständig Entscheidungen zur Gestaltung von Funktionen und Datenstrukturen treffen.

Das Problem Gegeben seien 4 Punkte *A, B, C, D* im zweidimensionalen Raum. Die Punkte *A* und *B* sowie die Punkte *C* und *D* bestimmen 2 Strecken. In dieser Aufgabe soll eine C++-Funktion entwickelt werden, welche entscheidet, ob die beiden Strecken einander schneiden oder nicht:



Die Strecken schneiden sich.



Die Strecken schneiden sich nicht.

Aufgaben

a) Entwerfen Sie die Datenstrukturen zu den geometrischen Objekten, die sie benötigen werden.

Lösungsstruktur Wir erwarten eine Ausformulierung der Datenstrukturen in C++. Verwenden Sie aussagekräftige Namen und sparen Sie nicht mit Kommentaren!

Massstab Je strukturierter Ihre Daten, umso besser wird Ihre Lösung bewertet. Gesamte Unteraufgabe: maximal 7 Punkte.

b) Entwickeln Sie eine C++-Funktion, die zu einer gegebenen Strecke und einem Punkt entscheidet, auf welcher Seite der durch die Strecke definierten Geraden der Punkt liegt.
Hinweis: Sei $\vec{n}$ der Normalvektor einer Geraden, $\vec{p}$ ein Punkt auf der Geraden und d der Abstand des Punktes $\vec{x}$ von der Geraden, so gilt folgende Beziehung $(\vec{x} - \vec{p})\frac{\vec{n}}{|\vec{n}|} = d$.

Lösungsstruktur Wir erwarten

- eine Beschreibung des Verhaltens der Funktion in drei bis vier Sätzen (auch graphisch) oder als Kommentar in der C++-Funktion. Erklären Sie insbesondere die Bedeutung der Eingabe- und Ausgabewerte (die Bedeutung von „der Punkt liegt auf der linken Seite“, „der Punkt liegt auf der rechten Seite“, „der Punkt liegt auf der Gerade“).
- eine vollständig in C++ formulierte Funktion, die mit der Beschreibung konsistent ist und die gestellte Aufgabe erfüllt.

c) Entwickeln Sie nun eine C++-Funktion, die entscheidet, ob zwei gegebene Strecken einander schneiden oder nicht. Setzen Sie dazu die die Existenz der Funktion aus Teilaufgabe b) voraus und benutzen Sie sie.

Lösungsstruktur Wir erwarten

- eine Beschreibung des Verhaltens der Funktion in drei bis vier Sätzen oder als Kommentar in der C++-Funktion. Erklären Sie insbesondere die Bedeutung der Eingabe- und Ausgabewerte (die Bedeutung von „Sie schneiden sich“, „Sie schneiden sich nicht“) sowie die grundlegende Idee der Funktion!
- eine vollständig in C++ formulierte Funktion, die mit der Beschreibung konsistent ist und die gestellte Aufgabe erfüllt.

Massstab für die Unteraufgaben b) und c) Vernünftige Beschreibung: 6 Punkte. Korrekt implementierte Funktion: 6 Punkte. Pro schwerem Syntaxfehler: 1 Punkt Abzug. Jede Unteraufgabe: 12 Punkte.

Aufgabe 4: Felder *(26 Punkte)*

In dieser Aufgabe wird anhand eines Beispiels geprüft, wie gut Sie mit Feldern (Arrays) umgehen können.

Gegeben ist die Verknüpfungstabelle einer binären algebraischen Operation $\circ$ über einer Menge $M = \{0, \ldots, n-1\}$.

Beispiel Die Zahlen $0, 1, 2, 3, 4$ werden wie folgt Verknüpft.

$\circ$	0	1	2	3	4
0	4	2	0	1	3
1	3	0	1	4	2
2	0	1	2	3	4
3	1	4	3	2	0
4	2	3	4	0	1

Die Tabelle ist wie folgt zu interpretieren: $0 \circ 0 = 4$, $0 \circ 1 = 2 \ldots$

In den folgenden zwei Teilaufgaben sollen Funktionen geschrieben werden, die überprüfen, ob diese Operation abgeschlossen und assoziativ ist.

Abgeschlossenheit Für alle Zahlen a und b aus M muss gelten: $a \circ b \in M$. In unserem Beispiel ist die Operation abgeschlossen.

Assoziativität Für alle Zahlen a, b und c aus M muss gelten: $a \circ (b \circ c) = (a \circ b) \circ c$. In unserem Beispiel ist die Operation nicht assoziativ, weil: $(0 \circ 4) \circ 3 = 3 \circ 3 = 2 \neq 4 = 0 \circ 0 = 0 \circ (4 \circ 3)$

Aufgaben

a) Implementieren Sie folgende C++-Funktion:

```
bool IstAbgeschlossen(const int o[n][n], const int n);
```

Diese Funktion soll genau dann den Wert **true** zurückgeben, wenn die binäre algebraische Operation, repräsentiert durch ihre Verknüpfungstabelle im Feld `int o[n][n]`, *abgeschlossen* ist.

Lösungsstruktur Abzugeben ist eine C++-Funktion (inkl. Funktionskopf), die die Aufgabenstellung erfüllt. Versehen Sie das Programm mit Kommentaren.

Massstab Zielgerichteter, korrekter Umgang mit Arrays: bis zu 6 Punkten. Gesamte Teilaufgabe (vollständige, syntaktisch korrekte Prozedur): 12 Punkte.

b) Implementieren Sie folgende weitere C++-Funktion:

```
bool IstAssoziativ(const int o[n][n], const int n);
```

Diese Funktion soll genau dann den Wert **true** zurückgeben, wenn die binäre algebraische Operation, repräsentiert durch ihre Verknüpfungstabelle im Feld `int o[n][n]`, *assoziativ* ist. Sie dürfen in Teilaufgabe b) voraussetzen, dass die Operation abgeschlossen ist.

Lösungsstruktur Abzugeben ist eine C++-Funktion (inkl. Funktionskopf), die die Aufgabenstellung erfüllt. Versehen Sie das Programm mit Kommentaren.

Massstab Zielgerichteter, korrekte Umgang mit Arrays: 8 Punkte Gesamte Teilaufgabe (vollständige, syntaktisch korrekte Prozedur): 14 Punkte.

Aufgabe 5: Dynamische Datenstrukturen und Rekursion *(22 Punkte)*

In der Vorlesung und den Übungen haben Sie gelernt mit Zeigervariablen umzugehen. Sie haben dynamische Datenstrukturen wie Listen und Bäume kennengelernt. Ausserdem wurden Sie mit dem Prinzip der Rekursion vertraut gemacht. In dieser Aufgabe prüfen wir Ihr Vermögen, mit Zeigervariablen, dynamischen Datenstrukturen und Rekursion umzugehen.

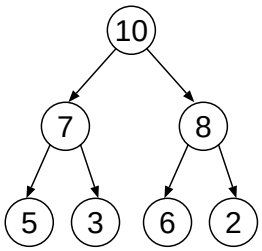
Gegeben ist ein binärer Baum mit Knoten

```
struct TreeNode { int key; TreeNode *left, *right; };
```

die der folgenden Eigenschaft genügen:

> *Der Schlüssel beider Nachfolger ist kleiner oder gleich dem Schlüssel des Vorgängers.*
> Formal: `v->left->key <= v->key` und `v->right->key <= v->key`

Beispiel Folgende Baumstruktur genügt dieser Eigenschaft:



Beachten Sie, dass dieser Baum *kein Suchbaum* ist!

Die Einfüge-Operation Will man einen neuen Schlüssel in diesen Baum einfügen, macht man das folgendermassen:

1. Man wählt zufällig ein Blatt des Baumes aus und hängt an dieses ein neues Blatt mit dem neuen Schlüssel.
2. Die zufällige Auswahl des Blattes passiert wie folgt: Man geht von der Wurzel des Baumes zu den Blättern. Bei jeder Entscheidung – links oder rechts – wirft man eine Münze, bis man zu einem Blatt gekommen ist.
3. Nun geht man vom neuen Blatt im Baum von unten nach oben und vertauscht so lange Nachfolger und Vorgänger bis der neue Schlüssel an der richtigen Stelle ist.

Aufgaben

a) Stellen Sie das Einfügen des Schlüssels 9 anhand des angegebenen Beispielbaumes graphisch dar.

Lösungsstruktur Abzugeben sind mehrere Baumskizzen, die den Baum nach jeder elementaren Operation (Blatt anhängen, vertauschen) darstellen. Machen Sie mit Pfeilen und Kommentaren kenntlich, wie die Baumskizzen zusammenhängen.

Massstab Korrekter Weg: 3 Punkte, Nachvollziehbarkeit Ihrer Gedanken: 1 Punkt. Gesamte Teilaufgabe: 4 Punkte.

$\Longrightarrow$

b) Die Einfüge-Operation soll rekursiv implementiert werden: Geben Sie die Rekursionsbasis sowie den Rekursionsschritt an.

Hinweis: Bedenken Sie, dass auch nach der eigentlichen Rekursion noch Dinge passieren können.

Lösungsstruktur Für die Beschreibung der Rekursionsbasis und des Rekursionsschrittes erwarten wir jeweils eine Erklärung in drei bis vier Sätzen. Geben Sie insbesondere die Bedingungen für die Rekursionsbasis und den Rekursionsschritt an.

Massstab Korrekte Beschreibung Rekursionsbasis, Rekursionsschritt: je 3 Punkte (Bedingung: 1 Punkt, Beschreibung: 2 Punkt). Gesamte Teilaufgabe: 6 Punkte.

c) Implementieren Sie folgende *rekursive* C++-Prozedur:

```
void Insert(int z, TreeNode* &r);
```

Diese Prozedur fügt den Schlüssel `z` in den Baum `r` ein.

Hinweis: Für die Zufallsentscheidung steht Ihnen die Funktion `double drand48(void)` zur Verfügung. Diese Funktion nimmt kein Argument und liefert eine Zufallszahl im Intervall $[0;1)$.

Lösungsstruktur Abzugeben ist eine vollständige C++-Prozedur (ink. Prozedurkopf), welche die Aufgabenstellung erfüllt. Versehen Sie Ihre Prozedur mit Kommentaren!

Massstab Korrekter und zielgerichteter Umgang mit Zeigervariablen und dynamischen Datenstrukturen: 7 Punkte. Rekursion: 4 Punkte. Kommentare: 1 Punkt. Pro schwerem syntaktischem Fehler: 1 Punkt Abzug. Gesamte Teilaufgabe: maximal 12 Punkte.

***** *Viel Erfolg!* *****

lj, Februar 2000