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

Herbst 1999
Freitag 24. September 1999

Name: _____
Vorname: _____
Legi-Nummer: _____
Unterschrift: _____

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl	Visum
1	26		
2	16		
3	24		
4	16		
5	16		
6	20		
Total	118		
Note			

Allgemeine Hinweise

- Zugelassene Hilfsmittel: **keine**.
- Legen Sie zu Anfang der Prüfung Ihre Legi neben sich auf den Tisch.
- Die Prüfung besteht aus 6 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, Fehlererkennung, Schleifen, C++-Semantik, Typen *(26 Punkte)*

a) In dieser Aufgabe geht es um die Syntax und die Bedeutung 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 Bestimmen Sie für jedes der folgenden Programmfragmente, ob

A) es *korrekt kompiliert* wird,

B) es *beim Kompilieren* einen Fehler verursacht,

C) es *beim Ausführen* einen Fehler verursacht oder

D) die Ausführung des Programmes *nie beendet* wird.

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

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

iii)
```
1  double a;
2
3  a= 3.0;
4  do {
5    a= a / 2.0;
6  } while (a < 2.0);
```

iv)
```
1  double a;
2
3  a= 3.0;
4  while (a < 2.0) {
5    a= a / 2.0;
6  };
```

Lösungsstruktur Für jedes Programmfragment erwarten wir uns die Klassifizierung in Form eines Grossbuchstabens (A, B, C, D). Zusätzlich erwarten wir

zu A: Geben Sie die Auswirkungen des Programmfragmentes auf die Variable `a` an.

zu B: Geben Sie an, an welcher Stelle der Fehler auftritt.

zu C: Geben Sie eine Begründung.

zu D: Geben Sie eine Begründung.

Die Ausarbeitung der Frage kann auf dem Prüfungsblatt erfolgen.

Massstab Pro korrekter Klassifizierung erhalten Sie 1 Punkt. Pro korrekter zusätzlicher Angabe erhalten Sie 1 Punkt.

$\Longrightarrow$

b) Mit der folgenden Aufgabe testen wir, inwiefern Sie in der Lage sind, Programme zu verstehen.

Aufgabe Wir präsentieren Ihnen drei Programmstücke. Für jedes Programmstück sollen Sie eine einzige Zuweisung finden, welche denselben Effekt auf die Variable `a` hat, wie das gesamte Programmstück.

i)
```
1   int a,b;
2
3   [...] // a>0, b>0
4
5   while (a-b > 0) {
6     a= a - b;
7   }
```

ii)
```
1   int a,b,c;
2
3   [...] // b>0
4
5   c= 0;
6   while (b > 0) {
7     c= c + a;
8     b--;
9   }
10  a= c;
```

iii)
```
1   bool a,b;
2
3   [...]
4
5   if (a && b) {
6     a= true;
7   } else {
8     a= false;
9   }
10
```

Lösungsstruktur Pro Programmstück erwarten wir eine in C++ kodierte Zuweisung für `a`, die denselben Effekt auf die Variable `a` hat, wie das gesamte Programmstück.

Beispiel: Für das Programmstück

```
1   int a,b;
2
3   if (b==1) {
4     a= 1;
5   } else {
6     a= b;
7   }
```

erwarten wir die Lösung

```
a= b;
```

Massstab Pro korrekter Zuweisung erhalten Sie 2 Punkte.

c) In der Vorlesung haben sie verschieden Typen für Variablen in C++ kennengelernt. Diese Aufgabe testet, wie gut Sie damit umgehen können.

Aufgabe Deklarieren Sie die Variablen `a`, `b` und `c` für jeden der folgenden Ausdrücke separat so, dass er fehlerlos kompiliert wird:

i) `a= b % c;`

ii) `a= (b > 1.0) && !(c=='a');`

iii) `a[c]= char(int('b') + b);`

iv) `a->b[a->c]= !c[b];`

Lösungsstruktur Wir erwarten pro Ausdruck eine in C++ gültige Deklaration aller vorkommenden Variablen. Es kann mehrere richtige Antworten geben. Wir erwarten uns nur je eine.

Massstab Pro richtig deklariertem Ausdruck erhalten Sie 3 Punkte. Die gesamte Unteraufgabe ist 12 Punkte wert.

Aufgabe 2: Logik *(16 Punkte)*

In dieser Aufgabe geht es um Logik und ihre Beziehung zu C++ Programmen. Sie sollen zeigen, dass Sie in der Lage sind, mit logischen Ausdrücken umzugehen und sie umzuformen.

Betrachten Sie folgendes C++-Programmstück:

```
1   int a,b,c;
2   bool D;
3
4   [...]
5   if (a > 2) {
6     if (c == -1)
7       D= false;
8     else
9       D= true;
10  } else {
11    if (b < 1)
12      D= false;
13    else
14      D=  (c == -1);
15  }
16  [...]
```

a) **Aufgabe** Füllen Sie aufgrund des Programmes und der darin enthaltenen logischen Bedingungen die folgende Wahrheitstabelle für die boolesche Variable `D` aus:

$(a > 2)$	$(b < 1)$	$(c == -1)$	D	In welcher Zeile wird der Wert von `D` bestimmt?
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Lösungsstruktur Wir erwarten von Ihnen eine vollständig ausgefüllte Wahrheitstabelle für `D`:

- Jede Zeile enthält einen Wahrheitswert für `D` in Form von falsch = 0, wahr = 1.
- Tragen Sie in der letzten Spalte der Wahrheitstabelle ausserdem ein, in welcher Zeile des Programmes der jeweilige Wert von `D` bestimmt wird.

Massstab Pro korrekter Zeile der Wahrheitstabelle erhalten Sie 1 Punkt.

b) **Aufgabe** Leiten Sie aus der Wahrheitstabelle von Aufgabe a) oder direkt aus dem Programmstück einen logischen Ausdruck für `D` her und formen Sie ihn so um, dass er einfacher wird. Formulieren Sie aufgrund des vereinfachten logischen Ausdruckes das C++-Programmstück neu.

Lösungsstruktur

- Wir erwarten einen logischen Ausdruck in C++-Notation mit den Zeichen `&&`, `||`, `!` bzw. in mathematischer Notation mit den Zeichen $\wedge$, $\vee$, $\neg$,
- gefolgt von einer Herleitung eines einfacheren Ausdruckes, der den selben Wahrheitsgehalt hat.
- Wir erwarten die Nennung des einfacheren Ausdruckes.
- Ausserdem erwarten wir ein Programmstück, das diesen einfacheren Ausdruck implementiert und somit das gegebene Programmstück ersetzen kann.

Massstab

- Der von der Wahrheitstabelle oder dem Programm abgeleitete richtige logische Ausdruck ist 2 Punkte wert.
- Die korrekte Herleitung des vereinfachten Ausdruckes ist 2 Punkte wert.
- Der korrekte vereinfachte Ausdruck ist 2 Punkte wert.
- Das aus diesen Überlegungen abgeleitete korrekte Programm ist 2 Punkte wert.

Aufgabe 3: Arrays, Records, Prozeduren, Funktionen *(24 Punkte)*

In der Vorlesung haben Sie gelernt, was Arrays und Records sind. Sie haben sich mit Prozeduren und Funktionen vertraut gemacht. In dieser Aufgabe sollen Sie anhand eines Beispiels zeigen, dass Sie mit diesen Elementen der Programmiersprache C++ umgehen können.

Für beide Teilaufgaben geben wir Ihnen folgende Definitionen vor:

```
1   const int N 100;       // Maximale Groesse der Arrays
2
3   struct Matrix {
4     int m;               // Anzahl Zeilen
5     int n;               // Anzahl Spalten
6     double a[N][N];      // Inhalt der Matrix
7   };
8
9   struct Vector {
10    int n;               // Anzahl Eintraege
11    double a[N];         // Inhalt des Vectors
12  };
```

Setzen Sie für beide Teilaufgaben die Existenz der Funktion

```
double fabs(double x)
```

voraus, die den Absolutwert der Variablen `x` liefert.

a) **Aufgabe** Schreiben Sie in C++ eine Prozedur

```
void ZeilenSumme(Matrix A, Vector &s) { ... }
```

welche für jede Zeile der Matrix `A` die Summe der Absolutwerte ihrer Elemente ermittelt und diese in der korrespondierenden Zeile des Vektors `s` speichert und den Vektor `s` zurückgibt. Eine Matrix `A` hat `A.m` Zeilen und `A.n` Spalten.

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

Massstab Der zielgerichtete, korrekte Umgang mit Records und Arrays wird mit der 6 Punkten bewertet. Die gesamte Teilaufgabe (vollständige, syntaktisch korrekte Prozedur) ist 12 Punkte wert.

b) **Definition 1** *Eine* quadratische *Matrix A heisst* diagonaldominant, *wenn für jede Zeile i der Matrix gilt*

$$\left(\sum_{\substack{j=0 \\ j \neq i}}^{j<n} |A_{i,j}| \right) < |A_{i,i}|$$

Aufgabe Schreiben Sie in C++ eine Funktion

```
bool diagonaldominant(Matrix A) { ... }
```

welche bestimmt, ob die angegebene Matrix `A` diagonaldominant ist (`true`) oder nicht (`false`). Verwenden Sie zur Berechnung der Zeilensumme die Prozedur aus Aufgabe a. Setzen Sie in dieser Teilaufgabe voraus, dass sie Teilaufgabe a gelöst haben (d.h. die Prozedur `Zeilensumme` existiert).

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

Massstab Der zielgerichtete, korrekte Umgang mit Prozeduren und Funktionen wird mit 6 Punkten bewertet. Die gesamte Teilaufgabe (vollständige, syntaktisch korrekte Funktion) ist 12 Punkte wert.

Aufgabe 4: Maximale Teilfolge *(16 Punkte)*

Mit dieser Aufgabe wollen wir Ihre algorithmischen Fähigkeiten testen.

Gegeben ist eine Folge A von (eventuell negativen) ganze Zahlen:

```
int A[N];
```

Definition 2 *Die Folge A enthält Teilfolgen, die an der Position i beginnen und an der Position j enden. Von jeder dieser Teilfolgen kann man die Summe ihrer Elemente mit* $s = \sum_{k=i}^{j} A_k$ *berechnen. Die Teilfolge mit dem grössten Wert s heisst* maximale Teilfolge von *A.*

Die Summe über eine leere Teilfolge ist 0. Daraus folgt, dass die Summe der maximalen Teilfolge 0 ist, wenn die Folge A nur negative *Zahlen enthält.*

Aufgabe Schreiben Sie in C++ eine Prozedur

```
void maxteilfolge(const int A[], int n, int &i, int &j, int &s)
```

welche die maximale Teilfolge von `A` findet und ihre Startposition `i`, ihre Endposition `j` sowie die Summe ihrer Elemente `s` zurückgibt. Die Variable *n* gibt dabei die tatsächliche Länge des Arrays `A` an. Es wird von Ihnen nicht erwartet, dass Sie ein effizientes Programm schreiben.

Lösungsstruktur Abzugeben ist eine vollständige C++-Prozedur (inkl. Prozedukopf), die die Aufgabenstellung erfüllt. Versehen Sie Ihr Programm mit Kommentaren.

Massstab Uns ist wichtig, dass Sie imstande sind, eine Methode zu finden, die die Aufgabenstellung erfüllt. Wenn wir Ihren Weg nachvollziehen können und er richtig ist, erhalten Sie 8 Punkte. Für eine vollständige, syntaktisch korrekte, richtige Prozedur erhalten Sie 16 Punkte.

Aufgabe 5: Rekursive Umwandlung von Zahlen *(16 Punkte)*

Mit dieser Aufgabe testen wir anhand eines Beispiels Ihre Fähigkeit, Rekursion zu erkennen und sie in C++ zu beschreiben.

Um eine natürliche Zahl in einem Zahlensystem auszugeben, verwendet man folgende Vorgangsweise (die auf dem Hornerschema basiert):

- Man dividiert die Zahl solange durch die Basis des Zahlensystems bis 0 herauskommt. Man merkt sich dabei die Reste dieser Divisionen.
- Die Reste der Divisionen ergeben die neue Zahlendarstellung in umgekehrter Reihenfolge.

Beispiele

Basis=10:

$$\begin{aligned} 1999 &= 199*10+\boxed{9} \\ 199 &= 19*10+\boxed{9} \\ 19 &= 1*10+\boxed{9} \\ 1 &= 0*10+\boxed{1} \end{aligned}$$

Basis=2:

$$\begin{aligned} 1999 &= 999*2+\boxed{1} \\ 999 &= 499*2+\boxed{1} \\ 499 &= 249*2+\boxed{1} \\ 249 &= 124*2+\boxed{1} \\ 124 &= 62*2+\boxed{0} \\ 62 &= 31*2+\boxed{0} \\ 31 &= 15*2+\boxed{1} \\ 15 &= 7*2+\boxed{1} \\ 7 &= 3*2+\boxed{1} \\ 3 &= 1*2+\boxed{1} \\ 1 &= 0*2+\boxed{1} \end{aligned}$$

Basis=16:

$$\begin{aligned} 1999 &= 124*16+\boxed{15(=\mathrm{F}_{16})} \\ 124 &= 7*16+\boxed{12(=\mathrm{C}_{16})} \\ 7 &= 0*16+\boxed{7} \end{aligned}$$

$$1999_{10} = 7\mathrm{CF}_{16} = 11111001111_2$$

Wenn wir die Zahl umwandeln wollen, stehen wir vor folgendem Problem: Die letzte Ziffer wird zuerst berechnet, die erste zuletzt. Die erste Ziffer wollen wir aber zuerst ausgeben, die letzte zuletzt. Dieses Problem kann mit Rekursion gelöst werden.

Aufgabe Schreiben Sie in C++ eine *rekursive* Prozedur

```
void Convert(int z, int b) { ... }
```

welche die natürliche Zahl `z` in der Zahlendarstellung zur Basis $b, 2 \leq b \leq 16$ auf dem Bildschirm ausgibt.

Benützen Sie zur Darstellung der Ziffern 0–15, die Zeichen ‘0’,‘1’,‘2’,‘3’,‘4’,‘5’,‘6’,‘7’,‘8’,‘9’,‘A’,‘B’,‘C’,‘D’,‘E’,‘F’.

Lösungsstruktur Wir erwarten eine vollständige, in C++ geschriebene Prozedur (inkl. Prozedurkopf), die eine *Rekursion* enthält. Trennen Sie scharf zwischen *Rekursionsbasis* und *Rekursionsvorschrift.* Versehen Sie Ihre Prozedur mit Kommentaren.

Massstab Der zielstrebige Einsatz von Rekursion wird mit 8 Punkten bewertet. Eine syntaktisch korrekte, richtige Prozedur erhält 16 Punkte.

Aufgabe 6: Dynamische Datenstrukturen: Bäume *(20 Punkte)*

Diese Aufgabe testet anhand eines Beispiels Ihre Kenntnisse von dynamischen Datenstrukturen. Es wird getestet, ob Sie mit Pointern vertraut sind und mit der Datenstruktur des Baumes umgehen können.

Gegeben ist ein Baum. Seine Knoten sind definiert durch

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

a) Entwickeln Sie eine rekursive Funktion

```
int Gewicht(TreeNode* r) { ... }
```

welche die Anzahl der Knoten im Baum `r` zurückgibt.

Lösungsstruktur Abzugeben ist eine vollständige, in C++ verfasste Funktion (inkl. Funktionskopf). Versehen Sie Ihre Funktion mit Kommentaren.

Massstab Für den richtigen, zielgerechten Umgang mit der Baumstruktur sowie für eine korrekte Strategie zur Berechnung des Gewichtes erhalten Sie 5 Punkte. Für eine vollständig korrekte Funktion erhalten Sie 10 Punkte.

b) Ändern Sie die Funktion `Gewicht` so ab, dass die Schlüsselwerte (`key`) aller Knoten `w` im Baum `r` ausgegeben werden, für die gilt

$$\frac{1}{3} \leq \frac{\texttt{Gewicht(w->left)}}{\texttt{Gewicht(w)}} \leq \frac{2}{3}$$

Lösungsstruktur Abzugeben ist eine vollständige, in C++ verfasste Funktion (inkl. Funktionskopf). Versehen Sie Ihre Funktion mit Kommentaren.

Massstab Für den richtigen, zielgerechten Umgang mit der Baumstruktur sowie für eine korrekte Strategie zur Ausgabe der richtigen Knoten erhalten Sie 5 Punkte. Für eine syntaktisch korrekte, effiziente Lösung (die jeden benötigten Wert nur einmal berechnet) erhalten Sie 10 Punkte.

* * * * * Viel Erfolg * * * * *

lj, September 1999