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

Dr. K. Simon

Studiengang Maschinenbau und Verfahrenstechnik

1. Vordiplom, Informatik I

Herbst 2000

Freitag, 29. September 2000

Name: _____

Vorname: _____

Legi-Nummer: _____

Unterschrift: _____

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl	Visum
1	17		
2	15		
3	21		
4	14		
5	14		
Total	81		
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, Typendeklarationen *(17 Punkte)*

a) In der Vorlesung haben sie die Syntax der Sprachelemente von C++ kennengelernt und ihre Bedeutung erfahren. In dieser Aufgabe können Sie auf unterschiedlichen Ebenen zeigen, ob Sie in der Lage sind, korrekte C++-Sprachelemente von falschen zu unterscheiden.

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

Aussagen

A) Der Programmteil wird *fehlerlos kompiliert*, *läuft fehlerfrei* und produziert eine *Ausgabe*.
B) Der Programmteil enthält einen *syntaktischen Fehler* oder einen *Fehler, der zur Laufzeit auftritt*.
C) Die Programmteil enthält zwar keine Fehler, jedoch dauert die *Ausführung* des Programmteiles *unendlich lange*.

Lösungsstruktur Wir erwarten pro Programmteil *genau eine* Klassifizierung in Form eines Grossbuchstabens (A,B,C). Als „Begründung“ erwarten wir je nach Auswahl des Grossbuchstabens:

A) Geben Sie die *Ausgabe* des Programmteiles an.
B) Geben Sie den *Ort des Fehlers* und *eine Beschreibung des Fehlers* an: Was genau ist falsch?
C) Geben Sie eine *Begründung* an: Warum läuft das Programmstück unendlich lange?

Die Lösung kann auf dem Prüfungsblatt abgegeben werden.

Beispiel

```
1   int a;
2
3   a= 5;
4   if (3 = a) {
5           a= a+1;
6   } else {
7           a= a*2;
8   }
9
10  cout << a << endl;
```

Dieser Programmteil führt zur Klassifizierung B.
Begründung: In Zeile 4 enthält die if-Anweisung statt einer Bedingung eine fehlerhafte Zuweisung zu einer Konstanten.

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

Die Programmteile

i)
```
1   int i, a;
2
3   a= 1;
4   i= 0;
5   do {
6     a= a*2;
7     i= i;
8   } while(i<2);
9
10  cout << a << endl;
```

ii)
```
1   int i, a;
2
3   a= 1;
4   i= 0;
5   do {
6     a= a*2;
7     i++;
8   } while(i<2);
9
10  cout << a << endl;
```

iii)
```
1   struct A {
2     int a,b;
3   };
4   A *a;
5
6   a= 0;
7   a->a= 2;
8   a->b= 3;
9
10  cout << a->a << endl;
```

iv)
```
1   struct A {
2     int a,b;
3   };
4   A *a;
5
6   a= new A;
7   a.a= 2;
8   a.b= 3;
9
10  cout << a.a << endl;
```

⟹

b) In der Vorlesung haben Sie gelernt, wie man Variablen, Funktionen und Prozeduren deklariert. In dieser Aufgabe sollen Sie *deklarieren* und zeigen, dass Sie das können.

Aufgabe Wir präsentieren Ihnen zwei korrekt codierte C++-Programmstücke (i-ii), aus denen klar hervorgeht, welchen Typ die „Namen“ `a`, `b` und `i` haben. Sie sollen für jedes Programmstück diese Programmelemente deklarieren.

Lösungsstruktur Wir erwarten uns pro Programmteil je eine Deklaration für *alle* `a`, `b` und `c` in korrekter C++-Syntax.

Beispiel

```
1   a= 3;
2   b= '2';
3   c= (char(a)=='2');
```

Dieser Programmteil verlangt folgende Deklaration der Objekte `a`, `b`, und `c`:

```
1   int a;
2   char b;
3   bool c;
```

Massstab Bewertet werden nur *notwendige* Deklarationen. Korrekte einfache Datentypen erhalten 1 Punkt. Korrekte erweiterte Datentypen werden mit bis zu 3 Punkten bewertet. Die gesamte Teilaufgabe ist 9 Punkte wert.

Die Programmstücke

i)
```
1   a= 'b';
2   b[c]= a;
```

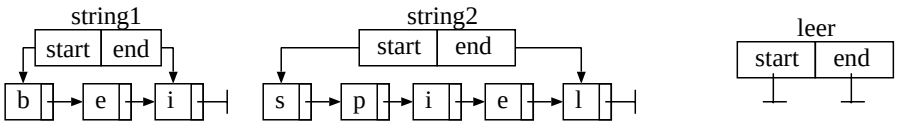
ii)
```
1   c.a= (a=='b');
2   c.b= float(b%2);
```

Aufgabe 2: Lineare Listen *(15 Punkte)*

In der Vorlesung haben Sie gelernt, was Zeigervariablen (Pointer) sind, wie man dynamische Datenstrukturen definiert und wie man mit ihnen umgeht. In dieser Aufgabe sollen Sie diese beiden Kenntnisse mithilfe von Listen unter Beweis stellen.

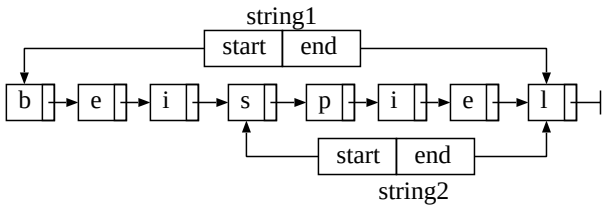
Die Idee für eine dynamische Datenstruktur Zeichenketten (strings) haben Sie in der Vorlesung als Arrays von Zeichen kennengelernt. Man kann sie auch als Listen realisieren. Dabei machen wir das so, dass jedes Listenelement jeweils *nur ein* Zeichen speichert. Diese Zeichen werden in Listenform miteinander verkettet.

Beispiele Folgende Graphik zeigt Zeichenketten – *string1*, *string2* und eine leere Zeichenkette – als Listen realisiert:



Concat Eine Operation, die häufig für Zeichenketten benötigt wird, ist das Zusammenhängen zweier Zeichenketten. Diese Operation nimmt als Eingabeparameter zwei Zeichenketten – *string1* und *string2*. Sie hängt *string2* an *string1*. Dabei wird in *string1* das Ergebnis zurückgegeben (also: *string1* wird verändert).

Beispiel Die folgende Graphik zeigt das Ergebnis von *concat* für die beiden anfangs gegebenen Zeichenketten *string1* und *string2*:



Aufgaben

a) Definieren Sie den graphisch dargestellten Zeichenkettendatentyp (wir nennen ihn in der Folge `Zeichenkette`) in C++ als lineare Liste.

Lösungsstruktur Abzugeben ist eine in C++ formulierte Definition einer Datenstruktur für Zeichenketten, die aus den gegebenen Beispielgraphiken abgeleitet ist (*Listenkopf* und *Listenelemente*). Sparen Sie nicht mit Kommentaren!

Bewertungskriterien Bewertet wird, ob Sie in der Lage sind, eine solche Datenstruktur vollständig aufgrund der Ihnen gegebenen Erläuterung in C++ zu definieren. 1 Punkt wird für erläuternde Kommentare vergeben. Pro schwerem Syntaxfehler gibt es 1 Punkt Abzug. Die gesamte Unteraufgabe ist 8 Punkte wert.

b) Implementieren Sie in C++ eine Prozedur

```
void concat(Zeichenkette &s1, Zeichenkette s2)
```

die die beschriebene Operation *Concat* auf die beiden Zeichenketten *s1* und *s2* implementiert.

Lösungsstruktur Abzugeben ist eine in C++ formulierte Prozedur, die gestellte Aufgabe erfüllt. Sparen Sie nicht mit Kommentaren. Eine gute vorbereitende Zeichnung, aus der der Ablauf der Prozedur ersichtlich ist, hat den Stellenwert eines Kommentars.

Bewertungskriterien Bewertet wird der Umgang mit Pointervariablen und dynamischen Datenstrukturen, sowie die korrekte Abfolge der Anweisungen. 1 Punkt wird für erläuternde Kommentare vergeben. Pro schwerem Syntaxfehler gibt es 1 Punkt Abzug. Die gesamte Unteraufgabe ist 7 Punkte wert.

Aufgabe 3: Rekursion *(21 Punkte)*

In der Vorlesung haben Sie erfahren, wie man die Fakultät und den grössten gemeinsamen Teiler zweier ganzer Zahlen rekursiv berechnen kann. In dieser Aufgabe wird Ihre Fähigkeit getestet, eine rekursive Funktion in C++ zu entwerfen.

Aus der Mathematik Binomialkoeffizienten kennen Sie vielleicht schon aus der Schule oder Ihren mathematischen Fächern an der ETH. Man schreibt sie als $\binom{n}{k}$ wobei $0 \leq n$ und $0 \leq k \leq n$. Jeder Binomialkoeffizient repräsentiert einen ganzzahligen Wert. Es gibt folgende (rekursive) Beziehung zwischen den Binomialkoeffizieten:

$$\binom{n}{0} = 1 \tag{1}$$

$$\binom{n}{n} = 1 \tag{2}$$

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \tag{3}$$

Aufgaben

a) Führen Sie den Binomialkoeffizienten $\binom{4}{2}$ aufgrund der Beziehungen (1), (2) und (3) auf einen ganzzahligen Wert zurück.

Lösungsstruktur Ihre Lösung besteht aus einer Kette von Gleichungen. Dabei repräsentiert jedes Gleichheitszeichung *genau eine* Ersetzung eines Binomialkoeffizienten gemäss Gleichung (1), (2) oder (3). Jedes Gleichheitszeichen versehen Sie mit der Nummer der verwendeten Gleichung (1), (2) oder (3) gemäss folgendem Beispiel:

Beispiel

$$\binom{4}{2} \overset{(3)}{=} \binom{3}{1} + \binom{3}{2} \overset{(...)}{=} \dots$$

Bewertung Jede Ersetzung eines Binomialkoeffizienten gemäss der gegebenen Beziehungen wird mit 1 Punkt bewertet. Die gesamte Unteraufgabe ist 8 Punkte wert.

b) Wir wollen den Binomialkoeffizienten $\binom{n}{k}$ auf rekursive Weise berechnen. Betrachten Sie die Gleichungen (1), (2) und (3). Welche der Gleichungen stellen die *Abbruchbedingung* für die Rekursion dar, welche enthalten den eigentlichen *Rekursionsschritt*?

Lösungsstruktur Erwartet wird eine *deutliche* Zuordnung der Gleichungszahlen zu den Begriffen *Rekursionsschritt* und *Abbruchbedingung*. Diese Zuordnung kann auf dem Aufgabenblatt geschehen.

Bewertung Jede richtige Zuordnung wird mit 1 Punkt bewertet. Die gesamte Teilaufgabe ist 3 Punkte wert.

c) Entwerfen Sie in C++ eine Funktion

```
int binomial(int n, int k)
```

welche den Binomialkoeffizienten $\binom{n}{k}$ auf rekursive Weise berechnet und das Ergebnis zurückgibt.

Lösungsstruktur Abzugeben ist eine korrekt formulierte C++-Funktion, die die gestellte Aufgabe erfüllt. Sparen Sie nicht mit Kommentaren.

Bewertung Bewertungskriterien sind

- die korrekte Übersetzung der rekursiven Elemente in die Programmiersprache C++: ingesamt 5 Punkte.
- die korrekte Verwendung von syntaktischen Elementen für Funktionen in C++: insgesamt 4 Punkte.

Schwere syntaktische Fehler werden mit jeweils 1 Punkt Abzug bewertet. Sinnvolle erläuternde Kommentare sind 1 Punkt wert. Gesamte Teilaufgabe b): 10 Punkte.

Aufgabe 4: Felder und Records *(14 Punkte)*

In Vorlesung und Übungen haben sie Records und Felder kennengelernt. Diese Aufgabe prüft anhand eines Beispiels, ob Sie mit diesen erweiterten Datentypen umgehen können.

Problem Aus der Mathematik kennen Sie Matrizen und Vektoren. Realisiert werden soll eine Prozedur, die eine Matrix-Vektor-Multiplikation durchführt.

Wiederholung Matrix-Vektor-Multiplikation Gegeben ist eine Matrix A der Dimension $m \times n$ und ein Vektor b der Dimension n. Die Multiplikation $x = Ab$ (x hat Dimension m) ist für die i. Komponente von x wie folgt definiert

$$x_i = \sum_{j=1}^{n} A_{i,j} b_j$$

Beispiel

$$A = \begin{bmatrix} 2 & -1 & 7 \\ 3 & 2 & 0 \\ -5 & 4 & 1 \\ 2 & 6 & -3 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \text{dann} \quad x = Ab = \begin{bmatrix} 2\cdot 1 + (-1)\cdot 1 + 7\cdot 1 \\ 3\cdot 1 + 2\cdot 1 + 0\cdot 1 \\ (-5)\cdot 1 + 4\cdot 1 + 1\cdot 1 \\ 2\cdot 1 + 6\cdot 1 + (-3)\cdot 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \\ 0 \\ 5 \end{bmatrix}$$

Die Datenstrukturen Die Datenstrukturen für Matrix und Vektor in C++ geben wir Ihnen vor:

```
1   const int N= 100;      // Maximale Dimension
2
3   struct Matrix {
4     int m,n;             // Dimension der Matrix: m Zeilen, n Spalten
5     double werte[N][N];  // Die Werte der Matrix
6   };
7
8   struct Vektor {
9     int n;               // Dimension des Vektors: n Zeilen
10    double werte[N];     // Die Werte des Vektors
11  };
```

Aufgabe Entwickeln Sie eine Prozedur

```
void MatVecMult(Matrix A, Vektor b, Vektor &x, bool &error)
```

die die Matrix-Vektor-Multiplikation $x = Ab$ in C++ realisiert und im Parameter `x` zurückgibt. Der Parameter `error` soll auf **true** gesetzt werden, falls die Dimensionen von `A` und `b` eine Multiplikation nicht möglich machen. Andernfalls soll im Parameter `error` der Wert **false** zurückgegeben werden.

Lösungsstruktur Abzugeben ist eine Prozedur in C++, die die gestellte Aufgabe erfüllt. Sparen Sie nicht mit Kommentaren.

Bewertungskriterien Bewertet werden der Umgang mit Feldern (7 Punkte), der Umgang mit Records (3 Punkte), die Umsetzung des Algorithmus (3 Punkte) und erläuternde Kommentare (1 Punkt). Pro schwerem Syntaxfehler gibt es 1 Punkt Abzug.

Aufgabe 5: Logik *(14 Punkte)*

In der Vorlesung haben Sie logische Ausdrücke, ihre Verknüpfungen sowie Wahrheitstabellen kennengelernt. In dieser Aufgabe sollen Sie zeigen, ob Sie in der Lage sind, diese Dinge zu verstehen und auf C++-Programme anzuwenden.

Problem Gegeben ist ein Programmstück in C++, das je nach Inhalt der Variablen `a`, `b` und `c` unterschiedliche Ausgaben macht:

```
1   int a,b,c;
2
3   ...
4
5   if (b<=2) {
6     if (c==15) {
7       if (a<0)
8         cout << "vielleicht" << endl;
9       else
10        cout << "wahr" << endl;
11    } else {
12      if (a>=0)
13        cout << "falsch" << endl;
14      else
15        cout << "vielleicht" << endl;
16    }
17  } else {
18    if (c!=15) {
19      if (a>=0)
20        cout << "falsch" << endl;
21      else
22        cout << "vielleicht" << endl;
23    } else {
24      if (a<0)
25        cout << "vielleicht" << endl;
26      else
27        cout << "wahr" << endl;
28    }
29  }
```

Das gegebene Programm entspricht einem logischen Ausdruck in den Bedingungen $a \geq 0$, $b \leq 2$ und $c = 15$. Diese können die Wahrheitswerte „wahr“ und „falsch“ annehmen. Die Ausgabe des Programmes jedoch kann *drei* Werte – „wahr“, „falsch“ und „vielleicht“ – annehmen.

Aufgaben

a) Erstellen Sie zum gegebenen Programm eine Wahrheitstabelle mit drei anstelle der üblichen zwei Resultatwerte.

Lösungsstruktur Ihre Tabelle enthält *alle möglichen Kombinationen von Bedingungen*, die jeweilige *Ausgabe des Programms* und die *Zeilennummer*, in der die Ausgabe geschieht.

Bewertungskriterien Bewertet wird, ob Sie in der Lage sind, eine solche Wahrheitstabelle zu erstellen. Die Teilaufgabe ist 8 Punkte wert.

b) Aufgrund der Erkenntnisse aus der Tabelle aus Teilaufgabe a) fassen Sie Bedingungen zusammen und vereinfachen das gegebene Programm derart, dass jede der Ausgaben „wahr“, „falsch“ und „vielleicht“ nur noch einmal im Programm vorkommt.

Lösungstruktur Abzugeben ist ein syntaktisch korrektes C++-Programm, das die gestellten Anforderungen erfüllt.

Bewertungskriterien Bewertet wird, ob Sie in der Lage sind aufgrund der Tabelle Vereinfachungen im gegebenen Programm vorzunehmen: Jede korrekt zusammengefasste Bedingung in Verbindung mit einer korrekten Ausgabe ist 2 Punkte wert. Für die korrekt gelöste Teilaufgabe erhalten Sie 6 Punkte.

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

lj, September 2000