Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Dr. K. Simon

Studiengang Maschinenbau und Verfahrenstechnik

1. Vordiplom, Informatik I

Herbst 2001

Freitag, 28. September 2001

Name: _____

Vorname: _____

Legi-Nummer: _____

Unterschrift: _____

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl	Visum
1	30		
2	25		
3	29		
4	25		
5	32		
Total	141		
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.
- Benutzen Sie keine rote Farbe!
- 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: Syntax, Typen, Prozeduren und Funktionen *(30 Punkte)*

a) In der folgenden Aufgabe testen wir anhand einer typischen Situation beim Programmieren (wie Sie sie oft genug bei der Lösung Ihrer Übungen erlebt haben) Ihr Vermögen, Fehler zu finden, zu identifizieren und zu korrigieren. Unterstützt werden Sie dabei durch Informationen, die Ihnen der Computer gibt.

Worum geht's? Wir präsentieren Ihnen vier Programmstücke, in denen jeweils ein Fehler enthalten ist. Dieser Fehler tritt entweder beim Kompilieren oder bei der Ausführung auf. Wir geben Ihnen die Fehlermeldung des Compilers oder die Fehlermeldung bei der Ausführung an.

Aufgabe Kennzeichnen Sie in den folgenden Programmen, wo der Fehler genau auftritt, beschreiben Sie, was der Fehler ist, und geben Sie einen Verbesserungsvorschlag an, der das Programm *auf irgendeine Weise fehlerlos* macht.

Lösungsstruktur Ihre Lösung enthält (1) die tatsächliche Zeilenangabe des Fehlers, (2) eine Beschreibung des Fehlers in Ihren eigenen Worten, (3) den Verbesserungsvorschlag.

Massstab Bewertet werden je Unteraufgabe die korrekte Angabe der Zeile (0.5 Punkte), die korrekte Beschreibung des Fehlers (0.5 Punkte), ein korrekter Verbesserungsvorschlag (1 Punkt). Jede Unteraufgabe gibt 2 Punkte. Die gesamte Teilaufgabe ist 8 Punkte wert.

i)
```
1  int main()
2  {
3    int a[100];
4    double i;
5
6    for (i= 0; i<100; i++) {
7      a[i]= int(i);
8    }
9  }
```

Fehler beim Kompilieren

```
 In function 'int main()':
7: invalid types 'int[100][double]' for
   array subscript
```

ii)
```
1  int main()
2  {
3    int i;
4    double b[100];
5
6    for (i= 0; i<100; i--) {
7      b[i]= i/0.5;
8    }
9  }
```

Laufzeitfehler

```
Segmentation fault
```

iii)
```
1   struct A {
2     int a;
3     double b[100];
4   };
5
6   int main()
7   {
8     A c;
9
10    for (A.a= 0; A.a<100; A.a++) {
11      A.b[A.a]= A.a/0.5;
12    }
13  }
```

Fehler beim Kompilieren

```
 In function 'int main()':
10: parse error before '.'
11: parse error before '.'
```

iv)
```
1   struct A {
2     int a;
3     double b[100];
4   };
5
6   int main()
7   {
8     A *c;
9     int i;
10
11    c= (A*)42;
12
13    for (i= 0; i<100; i++) {
14      c->b[i]= i/0.5;
15    }
16    delete(c);
17  }
18
19
```

Laufzeitfehler

```
Bus error
```

$\Longrightarrow$

b) In der Vorlesung haben Sie gelernt, wie man Variablen, deklariert. In dieser Aufgabe sollen Sie diese Fähigkeit demonstrieren.

Aufgabe Wir präsentieren Ihnen zwei korrekt codierte C++-Programmstücke (i-ii), aus denen hervorgeht, welchen Typ die „Namen“ a, b und c 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. Für Feldgrössen setzen sie irgendeine Zahl ein.

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 10 Punkte wert.

Die Programmstücke

i)
```
1   a[b]= (*c%2)>0;
```

ii)
```
1   c.a= int(c.b*3)%2;
2   a= c.a > 3;
3   b= 'a';
```

c) Die folgende Aufgabe behandelt Ihr Wissen im Bereich Prozeduren und Funktionen.

Aufgabe Ergänzen Sie auf sinnvolle Art die Kopfzeile der folgenden C++-Routinen und geben Sie einen sinnvollen Beispielaufruf an.

- Nehmen Sie dabei an, dass keine globalen Variablen definiert sind. Die Übergabe von Parametern erfolgt allein über den Aufruf.
- Wo eine Rückgabe sinnvoll erscheint (weil sonst Werte verloren gehen), *muss* eine Rückgabe erfolgen. Der gegebene Anweisungsteil der Routine darf dabei aber nicht abgeändert werden.
- Falls für eine verwendete Variable kein eindeutiger Typ bestimmbar ist, wählen Sie einen der Kandidaten aus. Achten Sie aber auf die Schlüssigkeit Ihrer Lösung.

Lösungsstruktur Ihre Antwort enthält *eine Kopfzeile* für die gegebene C++-Routine und *einen Aufruf* mit Deklaration aller verwendeten Variablen. Für den Aufruf müssen die übergebenen Variablen nicht initialisiert werden. Gegebenenfalls sind auch *Typen zu deklarieren*.

Bewertung Bewertet wird jeder korrekte Parameter (1 Punkt), die korrekte Einordnung und Handhabung von Funktion/Prozedur (1 Punkt), jede notwendige und korrekte Typendeklaration (1 Punkt) und jede notwendige und korrekte Variablendeklaration (0.5 Punkte). Die ganze Unteraufgabe ist 12 Punkte wert.

Die C++-Routinen

i)
```
1   ... maximum(...)
2   {
3     int i;
4     int max;
5
6     max= a[0];
7     for (i= 1; i<n; i++) {
8       if (a[i]>max) max= a[i];
9     }
10    return(max);
11  }
```

ii)
```
1   ... bereich(...)
2   {
3     int i;
4     res.inf= a[0];
5     res.sup= a[0];
6     for (i= 1; i<n; i++) {
7       if (a[i]<res.inf) res.inf= a[i];
8       if (a[i]>res.sup) res.sup= a[i];
9     }
10  }
```

Aufgabe 2: Records und Felder *(25 Punkte)*

In dieser Aufgabe geht es um Datenstrukturen, die Sie selber definieren sollen und auf denen Sie eine C++-Prozedur implementieren sollen.

Datenstrukturen für eine Matrix Die herkömmliche Datenstruktur für eine Matrix ist ein zweidimensionales Feld. Nehmen wir an, eine Matrix ist maximal $N \times N$ Werte gross, so beschreibt folgende Datenstruktur eine herkömmliche Matrix:

```
struct Matrix {     // herkoemmliche Matrix
  int m;            // Anzahl Zeilen
  int n;            // Anzahl Spalten
  doulbe val[N][N]; // Werte
};
```

Wie in der Mathematik üblich, verwenden wir dabei den ersten Index des 2-dimensionalen Feldes für die Zeilen und den zweiten Index des 2-dimensionalen Feldes für die Spalten.

In technischen Anwendungen erhält man häufig Matrizen, die nur ganz wenige Einträge enthalten, die ungleich 0 sind. Die grosse Masse von Einträgen ist 0. Man nennt solche Matrizen *dünn besetzt*. Verwendet man für ihre Speicherung ein zweidimensionales Array, so speichert man sehr viele Nullen. Diese will man nicht speichern. Daher bietet sich eine andere Datenstruktur an, die nur Werte ungleich 0 speichert. Eine solche Datenstruktur muss für jeden Wert der Matrix ungleich 0 *drei Werte* speichern:

- den Zeilenindex
- den Spaltenindex
- den Wert

a) Entwerfen Sie eine Datenstruktur `Duenn` für dünn besetzte Matrizen. Nehmen Sie an, dass eine dünn besetzte Matrix höchstens M Werte ungleich 0 enthält.

Lösungsstruktur Erwartet wird ein kommentiertes Codestück in C++, dass die Datenstruktur für eine dünn besetzte Matrix in der beschriebenen Weise zur Verfügung stellt. Die Datenstruktur soll höchst möglich strukturiert sein!

Massstab Bewertet wird:

- Korrekter Einsatz von benutzerdefinierten Datenstrukturen: 4 Punkte
- Vollständigkeit der zu speichernden Werte: 3 Punkte
- Kommentare: 1 Punkt

Die Unteraufgabe ist 8 Punkte wert.

b) Implementieren Sie eine C++-Prozedur

```
void Matrix2Duenn(Matrix matrix1, Duenn &matrix2)
```

welche eine auf konventionelle Weise gespeicherte Matrix in das Speicherformat einer dünn besetzten Matrix konvertiert. Dabei ist im Parameter `matrix1` die konventionell gespeicherte Matrix gegeben. Das Resultat `matrix2` enthält die selben Werte wie `matrix1`, allerdings im Format `Duenn`, welches Sie in Aufgabe a) definiert haben.

Lösungsstruktur Abzugeben ist eine vollständige und kommentierte C++-Prozedur, welche mit der Datenstruktur aus Aufgabe a) arbeitet und die gestellte Aufgabe erfüllt.

Massstab Bewertet wird

- Korrekte Anwendung der Datenstrukturen: 8 Punkte
- Gleiche Werte in `matrix1` und `matrix2`: 8 Punkte
- Kommentare: 1 Punkt

Die Unteraufgabe ist 17 Punkte wert.

Aufgabe 3: Logik *(29 Punkte)*

In dieser Aufgabe geht es um eine *logische Verknüpfung*, die mithilfe von *Wahrheitstabellen* untersucht und in C++ *implementiert* werden soll.

Die XOR-Verknüpfung $\oplus$, die sie in den Übungen kennengelernt haben, ist durch folgende Wahrheitstabelle definiert:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Das entspricht der logischen Formel $A \oplus B = (!A \;\&\&\; B) \| (A \;\&\&\; !B)$.

a) Zeigen Sie mithilfe von Wahrheitstabellen, dass gilt: $A \oplus (B \oplus C)$ und $(A \oplus B) \oplus C$ sind äquivalente Ausdrücke.

Lösungsstruktur Wir erwarten Wahrheitstabellen zu jedem der zu vergleichenden Ausdrücke und eine *sichtbare Hervorhebung* des Resultates.

Massstab Die Korrektheit der zu erstellenden Wahrheitstabellen wird insgesamt mit 8 Punkten bewertet. Die Kennzeichnung des richtigen Resultates ist 1 Punkt wert. Die Unteraufgabe gibt 9 Punkte.

b) Erstellen Sie eine weitere Wahrheitstabelle zum Ausdruck $A \oplus B \oplus C \oplus D$.

Lösungsstruktur Wir erwarten eine *vollständige* Wahrheitstabelle zum gegebenen Ausdruck.

Massstab Die Unteraufgabe gibt 8 Punkte.

c) Stellen sie eine allgemeine Regel für den Wert einer XOR-Verknüpfung mit beliebig vielen Operanden auf.

Lösungsstruktur Wir erwarten einerseits eine *Beschreibung* der gefundenen Regel in ein bis zwei *umgangssprachlichen Sätzen*. Formulieren Sie ausserdem (umgangssprachlich, in ein bis zwei Sätzen) Ihre Überlegungen, die Sie zur Aufstellung dieser Regel angestellt haben.

Massstab Sowohl korrekte Regel als auch Ihre Überlegungsbeschreibung werden mit je 2 Punkten bewertet. Die Unteraufgabe ist 4 Punkte wert.

d) Implementieren Sie in C++ eine Funktion

```
bool XOR(bool X[], int n)
```

welche den Ausdruck $X[0] \oplus \ldots \oplus X[n-1]$ berechnet.

Lösungsstruktur Wir erwarten eine vollständige in C++ verfasste Funktion, die mit Kommentaren versehen ist.

Massstab Bewertet werden je nach eingeschlagenem Lösungsweg

- die Strategie zur Berechnung des Ausdruckes mit bis zu 3 Punkten
- die Korrektheit der implementierten logischen Ausdrücke mit bis zu 3 Punkten
- die Kürze der implementierten logischen Ausdrücke mit bis 2 Punkten
- korrekte erläuternde Kommentare mit 1 Punkt

Aufgabe 4: Rekursion *(25 Punkte)*

In dieser Aufgabe geht es um einen Sachverhalt, den man sowohl *rekursiv* als auch *iterativ* implementieren kann. Sie sollen beides tun.

Catalan'sche Zahlen Die Catalan'schen Zahlen $c_n, n = 1, 2, \ldots$ sind wie die Fibbonacci-Zahlen Elemente einer Folge, die durch folgende Rechenvorschrift erzeugt wird:

$$\begin{aligned} c_0 &= 1 \\ c_n &= \sum_{i=0}^{n-1} c_i c_{n-1-i} \quad \text{für } n = 1, 2, \ldots \end{aligned}$$

a) Berechnen Sie die Catalan'schen Zahlen c_2, c_3, c_4.

Lösungsstruktur Zu jeder Zahl erwarten wir *den Wert* und einen Weg, der beschreibt, welche *Produkte von früheren Catalan'schen Zahlen* zusammengezählt werden müssen. Folgen Sie dabei folgendem Muster:

$$c_1 = c_0 c_0 = 1 \cdot 1 = 1$$

Massstab Jeder korrekte Wert und jeder korrekte Weg wird mit 1 Punkt bewertet. Die Unteraufgabe ist 6 Punkte wert.

b) Implementieren Sie die Funktion

```
int catalanr(int n)
```

welche die n-te Catalan'sche Zahl c_n auf *rekursive* Art berechnet. Beschränken Sie dabei die Anzahl der rekursiven Aufrufe auf ein Minimum.

Lösungsstruktur Abzugeben ist eine C++-Funktion, die die gestellte Aufgabe erfüllt. Versehen Sie Ihren Code mit Kommentaren.

Massstab Bewertet wird:

- Behandlung der Rekursivität: 4 Punkte
- Beschränkung der rekursiven Aufrufe: 2 Punkte
- Behandlung der Funktion: 2 Punkte
- algorithmische Umsetzung der Summe: 2 Punkte
- Kommentare: 1 Punkt

Die Unteraufgabe ist 11 Punkte wert.

c) Implementieren Sie die Funktion

```
int catalani(int n)
```

welche die n-te Catalan'sche Zahl c_n auf *iterative* Art berechnet. Dabei dürfen Sie annehmen dass n nicht grösser als 100 ist.

Lösungsstruktur Abzugeben ist eine C++-Funktion, die die gestellte Aufgabe erfüllt. Versehen Sie Ihren Code mit Kommentaren.

Massstab Bewertet wird:

- algorithmische Umsetzung der Aufgabe: 5 Punkte
- Datenstrukturen: 1 Punkt
- korrekte Behandlung der Funktion: 1 Punkt
- Kommentare: 1 Punkt

Die Unteraufgabe ist 8 Punkte wert.

Aufgabe 5: Dynamische Datenstrukturen: Sortierte Listen *(32 Punkte)*

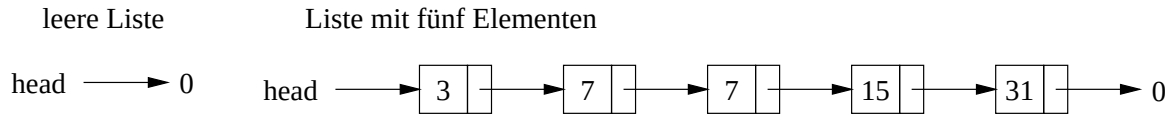
In dieser Aufgabe geht es um Zeigervariablen und dynamische Datenstrukturen.

Sortierte Listen Betrachten Sie die folgende Deklaration

```
1   struct Node {    // Listenelement
2     int key;       // Schluessel
3     Node *next;    // naechstes Listenelement
4   };
5
6   Node *head;      // Listenanfang
```

Eine sortierte Liste ist eine Liste von Elementen vom Typ `Node`, deren Elemente nach dem Schlüssel *aufsteigend sortiert* sind. Sie ist durch einen Zeiger (`head`) auf ihren Anfang geben. Eine sortierte Liste kann *mehrere Elemente mit gleichem Schlüssel* enthalten.

Beispiele



a) In die *nicht leere Liste* des obigen Beispiels sollen folgende Zahlen eingefügt werden:

i) 2 ii) 8 iii) 32

Erstellen Sie zu jeder einzufügenden Zahl eine Zeichnung resp. eine Zeichnungsfolge. Ausgangspunkt für jede Einfügung ist die nicht leere Liste des obigen Beispiels.

Lösungsstrukur Jede Zeichnung macht klar,

- an welchem Ort die Zahl eingefügt wird,
- welche Zeiger umgebogen werden,
- die Reihenfolge der Aktionen (durch Nummerierung).

Massstab Für jede richtige elementare Aktion vergeben wir 1 Punkt. Die Unteraufgabe ist 6 Punkte wert.

b) Implementieren Sie eine C++-Prozedur

```
void insert(Node *&head, int key)
```

welche die Zahl `key` als neues Element in eine sortierte Liste einfügt. Diese Liste ist durch einen Zeiger `head` auf ihren Anfang gegeben.

Lösungstruktur Abzugeben ist eine kommentierte C++-Prozedur.

Massstab Bewertet wird:

- Algorithmus: 5 Punkte
- Manipulation der dynamischen Datenstruktur: 4 Punkte
- Umgang mit Zeigervariablen: 3 Punkte
- Kommentare: 1 Punkt

Die Unteraufgabe ist 13 Punkte wert.

Hinweis: Um ein Element an der richtigen Stelle einfügen zu können, müssen sie seinen *Vorgänger* speichern!

$\Longrightarrow$

c) Implementieren Sie eine C++-Funktion

```
Node *delete(Node *&head, int key)
```

welche das Element mit dem Schlüssel `key` in der durch `head` gegebenen Liste sucht und aus der Liste löscht. Das gefundene Element wird von der Funktion als Funktionswert zurückgegeben. Wird kein Element gefunden, so ist das Resultat der Funktion der Null-Zeiger. Die Funktion *muss* bei der Suche ausnützen, dass die Liste sortiert ist!

Lösungstruktur Abzugeben ist eine kommentierte C++-Funktion.

Massstab Bewertet wird:

- Algorithmus: 6 Punkte
- Umgang mit Zeigervariablen: 3 Punkte
- Manipulation der dynamischen Datenstruktur: 2 Punkte
- Behandlung der Funktion: 1 Punkt
- Kommentare: 1 Punkt

Die Unteraufgabe ist 13 Punkte wert.

Hinweis: Um ein Element an der richtigen Stelle löschen zu können, müssen sie seinen *Vorgänger* speichern!

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

lj, September 2001