Fakultät Elektrotechnik/Informatik

Klausur Einführung in die Informatik I für Elektrotechniker (Version A)

18. Juli 2002

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung
(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	5	
2	7	
3	4	
4	2	
5	4	
6	10	
7	12	
Summe	44	

Spielregeln (**Jetzt lesen!**):

- Benutzen Sie für die Lösung der Aufgaben **nur** das mit diesem Deckblatt ausgeteilte Papier. Lösungen, die auf anderem Papier geschrieben werden, können **nicht** bewertet werden. Schreiben Sie ihre Lösung auch auf die Rückseiten der Blätter; benötigen Sie für eine Lösung mehr als ein Blatt, finden Sie am Ende der Klausur Leerblätter. Zusätzliches Papier können Sie von den Tutoren bekommen.
- Tragen Sie jetzt (vor Beginn der eigentlichen Bearbeitungszeit !!!) auf *allen* Blättern ihren Namen und ihre Matrikelnummer ein.
- Schreiben Sie deutlich! Unleserliche oder zweideutige Lösungen können nicht gewertet werden.
- Schreiben Sie **nicht** mit Bleistift.
- Bitte schreiben Sie nicht mit rotem oder grünem Stift (das sind die Farben für die Korrektur).
- Lesen Sie die Aufgaben jeweils bis zum Ende durch; oft gibt es hilfreiche Hinweise!
- Wir weisen noch einmal darauf hin, daß die Benutzung von Taschenrechnern und anderen elektronischen Hilfsmitteln nicht gestattet ist.

Viel Erfolg!

• **AUFGABE 1 (5 Punkte) Theorie.**

1. (1 Punkt) Was versteht man unter dem *Einerkomplement* und mit welchem Problem ist es behaftet?

2. (2 Punkte) Was ist eine *CPU* und wie gelangen die Daten aus dem Speicher dorthin?

3. (2 Punkte) Was verstehen Sie unter *Objekten*? Wodurch sind sie charakterisiert und wie werden sie beschrieben?

• AUFGABE 2 (7 Punkte) JAVA.

1. (1 Punkt) Geben Sie die Kommandozeile ein, mit der man eine Java Quellcodedatei mit dem Namen `Foo.java` compilieren kann!

2. (3 Punkte) Geben Sie den Aufwand der Methoden `f` und `g` in Abhängigkeit der Eingabe in *Big-O-Notation* an.

```
int f(int n) {
    int i = 0;
    int s = 0;
    while (i < n * n) {
        s = s + 3;
        i++;
    }
    return s;
}

int g(int k) {
    int s = 0;
    int p = f(k);
    for (int i = 1; i < p / (k * k); i++) {
        int j = k - i;
        do {
            j--;
            s = s + f(j);
        } while (j >= 0);
    }
    return s;
}
```

3. (3 Punkte) Wo sind die Fehler im folgenden *JAVA*-Code? Beschreiben Sie die Fehler und geben Sie die Zeilenzahlen des Auftretens an. (Folgefehler werden wie üblich ignoriert.)

```
 1 class PairArray {
 2     Pair[] p;
 3
 4     newPairArray() {
 5         this.p = new Pair[0];
 6     }
 7
 8     int init() {
 9         int i, j;
10         final byte b = 10;
11         p = new Pair[b+1];
12         i = b;
13         do {
14             p[j] = new Pair(i);
15             i--;
16             j = i;
17             p[j].b = b;
18         } while (i > 0);
19         p[i] = new Pair(b);
20         return p[0];
21     }
22 }
23
24 class Pair {
25     int a, b;
26
27     Pair(int a) {
28         this.a = a;
29     }
30 }
```

• **AUFGABE 3 (4 Punkte) Zahlsysteme.**

1. (2 Punkte) Wandeln Sie die reelle Zahl 0.85 in eine Dualzahl mit 4 binären Nachkommastellen um und nehmen Sie dann die Umwandlung noch einmal in Gegenrichtung vor.

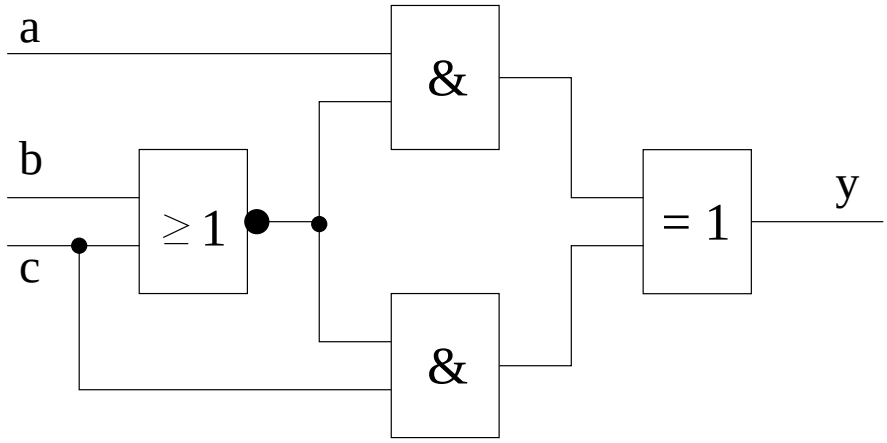
   Geben Sie den dabei auftretenden Fehler an. Lassen Sie den Lösungsweg erkennen.

2. (2 Punkte) Führen Sie folgende Berechnungen unter Verwendung der 2-Komplementdarstellung auf einem imaginären 4-Bit-Rechner aus. Lassen Sie den Lösungsweg erkennen. Wo findet ein *Over-* oder *Underflow* statt? Woran erkennt man das Auftreten eines *Over-* bzw. *Underflows*?

   - $-7 + (-3)$
   - $2 + 6$
   - $1 - 4$

• AUFGABE 4 (2 Punkte) Schaltungen.

Stellen sie die Wertetabelle für die folgende Schaltung auf.



• AUFGABE 5 (4 Punkte) Quersumme.

Schreiben Sie eine Methode

```
int qsum(int n)
```

welche die Quersumme einer ganzen Zahl berechnet, d.h. die Summe ihrer Ziffern.

Hinweis: `n % 10` ergibt die Einerstelle von `n`.

Beispiele:

- `qsum(4) = 4`
- `qsum(1024) = 7`
- `qsum(-58) = 13`

• AUFGABE 6 (10 Punkte) Raumreservierung.

Für die Feierlichkeiten am 3. Oktober vermietet die TU Berlin Räume. Damit ein Raum nicht versehentlich gleichzeitig an zwei Veranstalter vermietet wird, sollen die Reservierungen rechnergestützt bearbeitet werden.

Eine Reservierung wird vorgenommen unter Angabe

- der Raumnummer,
- des Namens des Veranstalters,
- des Startzeitpunkts und
- des Endzeitpunkts.

Wir nehmen zur Vereinfachung an, daß Anfang und Ende einer Reservierung immer zur vollen Stunde erfolgen.

```
class Reserv {
    String raum;
    String veranstalter;
    int start;              // Startzeitpunkt
    int ende;               // Endzeitpunkt

    // weiteres siehe unten
}
```

1. (1 Punkt) Wir wollen im Folgenden mit zulässigen Reservierungen arbeiten. Eine Reservierung ist zulässig, wenn der Startzeitpunkt nicht nach dem Endzeitpunkt liegt. Erweitern Sie die Klasse `Reserv` um eine Methode

   ```
   boolean zulaessig()
   ```

   um zu testen, ob eine Reservierung zulässig ist.

2. (1 Punkt) Erweitern Sie die Klasse `Reserv` um eine Methode

   ```
   int dauer()
   ```

   zur Berechnung der Zeitdauer, welche die jeweilige Reservierung den Raum belegen würde. Sollte die Reservierung nicht zulässig sein, geben Sie `0` zurück.

3. (2 Punkte) Erweitern Sie die Klasse `Reserv` um eine Methode

   ```
   boolean vertraeglich(Reserv other)
   ```

   die genau dann `true` liefert, wenn die Reservierung zulässig ist und sich nicht mit einer anderen zulässigen Reservierung überschneidet. (Die Zeit des Wechsels zwischen zwei Veranstaltungen wird dabei vernachlässigt.)

4. (3 Punkte) Schreiben Sie nun eine Methode

   ```
   Reserv[] entferneName(String name, Reserv[] a)
   ```

   die den Namen eines Veranstalters und ein Feld von Reservierungen bekommt und alle Reservierungen auf diesen Namen aus der Liste entfernt.

   Hinweis: Zwei Strings können mit der Methode `boolean String.equals(String other)` verglichen werden.

5. (3 Punkte) Schreiben Sie ferner eine Methode

   ```
   int belegung(Reserv[] a, String raum)
   ```

   die anhand eines Belegungsfeldes bestimmt, wie viele Stunden der angegebene Raum belegt ist.

• AUFGABE 7 (12 Punkte) Matrixmultiplikation.

Bei der Multiplikation zweier Matrizen A und B ergeben sich die Elemente $C_{i,j}$ der Produktmatrix als Skalarprodukt der i-ten Zeile von A mit der j-ten Spalte von B.

$$C_{i,j} = \sum_k A_{i,k} * B_{k,j}$$

Fehlerfälle wie ungültige Spaltennummern etc. können – außer in Teilaufgabe 4 – ignoriert werden.

1. (3 Punkte) Schreiben Sie eine Methode

   float[] getSpalte(float[][] A, int spalte)

   welche die Spalte mit der Nummer `spalte` aus der Matrix liefert.

   Beispiel: Für $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ gilt `getSpalte(M, 0) = [1, 4]` und `getZeile(M, 0) = [1, 2, 3]`.

2. (2 Punkte) Schreiben Sie eine Methode

   float[] getZeile(float[][] A, int zeile)

   welche analog die entsprechende Zeile liefert.

3. (3 Punkte) Schreiben Sie nun eine Methode

   float skalarprod(float[] A, float[] B)

   die das Skalarprodukt zweier Felder berechnet, d.h. die Summe der elementweisen Produkte $A_i * B_i$.

   Beispiel: `skalarprod([1, 2, 3], [4, 5, 6]) = 1 * 4 + 2 * 5 + 3 * 6 = 22`

4. (4 Punkte) Schreiben Sie ferner eine Methode

   `float[][] matmult(float[][] A, float[][] B)`

   welche die Matrixmultiplikation $A * B$ berechnet. Falls die Spaltenzahl von `A` nicht mit der Zeilenzahl von `B` übereinstimmt, d.h. die Multiplikation nicht ausführbar ist, geben Sie eine leere Matrix zurück.