Fakultät Elektrotechnik/Informatik

Klausur Einführung in die Informatik I für Elektrotechniker 20. Februar 2003

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung
(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	4	
2	6	
3	4	
4	4	
5	6	
6	10	
7	10	
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 (4 Punkte) Theorie.**

1. (1 Punkt) Was versteht man unter einem *Algorithmus*?

2. (1 Punkt) Was ist eine *ALU* und wozu dient sie?

3. (2 Punkte) Was verstehen Sie unter *Klassen*? Wodurch sind sie charakterisiert und wie verhalten sie sich zu *Objekten*?

• **AUFGABE 2 (6 Punkte) JAVA.**

1. (2 Punkte) Geben Sie den Aufwand des unten stehenden Programms in Abhängigkeit der Eingabe sowie den Aufwand der Methoden `k` und `l` in *Big-O-Notation* an.

```
class Foo {
    public static void main(String[] args) {
        Bar b = new Bar();
        b.run();
    }
}

class Bar {
    void run() {
        int n = Terminal.askInt("n = ");
        int ergebnis = k(n);
        Terminal.println("ergebnis = " + ergebnis);
    }

    int k(int p) {
        int erg = 0;
        for (int i=p; i>0; i--) {
            erg = erg + l(p);
        }
        return erg;
    }

    int l(int x) {
        if (x < 2) {
            return 0;
        } else {
            return 1 + l( x/2 );
        }
    }
}
```

2. (1 Punkt) Geben Sie die Kommandozeile an, mit der man das obige Java-Programm starten kann, nachdem es bereits erfolgreich compiliert wurde.

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 Foo {
 2     float do_strange(int x) {
 3         int i;
 4         int N = Math.max(x, 20);
 5         double[] a = new double[N];
 6         Bar b = new Bar();
 7         i = 0;
 8         b.value = i;
 9         while (i < N) {
10             a[i] = Terminal.askDouble("a[" + i + "] = ");
11             b.foo.value = i;
12             i++;
13         }
14         return a[0];
15     }
16 }
17
18 class Bar {
19     int value;
20     Foo foo;
21
22     newBar(int value) {
23         this.value = value;
24     }
25 }
```

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

1. (2 Punkte) Wandeln Sie die reelle Zahl 0.65 in eine Dualzahl mit 4 binären Nachkommastellen um und nehmen Sie dann die Umwandlung noch einmal in Gegenrichtung vor.

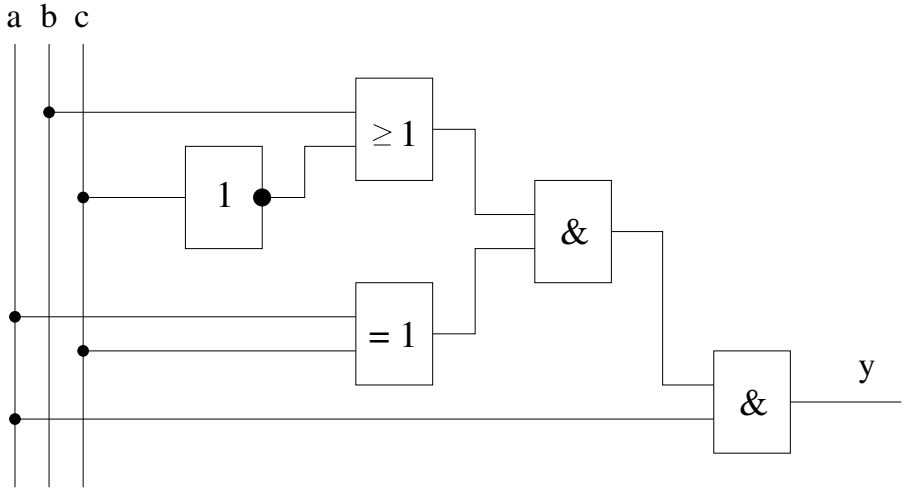
   Geben Sie den dabei auftretenden Fehler an. Lassen Sie den Lösungsweg erkennen.

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*?

   - $4 + 4$
   - $-3 + 7$
   - $-6 - 5$

• **AUFGABE 4 (4 Punkte) Schaltungen.**

1. (2 Punkte) Stellen sie die Wertetabelle für die folgende Schaltung auf.



2. (2 Punkte) Geben Sie dann eine einfachere Schaltung an, die das Gleiche berechnet.

• AUFGABE 5 (6 Punkte) Größter gemeinsamer Teiler.

Der Algorithmus von Euklid zur Bestimmung des ggT (größten gemeinsamen Teilers) zweier natürlicher Zahlen n und m funktioniert folgendermaßen (für n und m größer 0):

Wenn beide Zahlen gleich sind, ist dies der ggT der beiden ursprünglichen Zahlen. Ansonsten subtrahiere die kleinere der beiden Zahlen von der größeren. Mit der kleineren der beiden ursprünglichen Zahlen und mit der eben gebildeten Differenz ist das Verfahren zu wiederholen.

1. (4 Punkte) Schreiben Sie eine Methode

   ```
   int ggT(int n, int m)
   ```

   welche den größten gemeinsamen Teiler zweier Zahlen berechnet.

   Beispiele:

   - `ggT(6,5) = 1`
   - `ggT(6,2) = 2`
   - `ggT(6,10) = 2`
     da *ggT(6,10) = ggT(6, 10-6) = ggT(6,4) = ggT(6-4, 4) = ggT(2, 4) = ggT(2, 4-2) = ggT(2,2) = 2*

2. (2 Punkte) Schreiben Sie ferner eine Methode

   ```
   boolean teilerFremd(int n, int m)
   ```

   welche angibt, ob zwei Zahlen teilerfremd sind oder nicht. Teilerfremd sind Zahlen, wenn ihr ggT gleich 1 ist.

   Beispiele:

   - `teilerFremd(6,5) = true`
   - `teilerFremd(6,10) = false`

• AUFGABE 6 (10 Punkte) Kundenverwaltung eines Versandhändlers.

In dieser Aufgabe soll die Kundenverwaltung eines Versandhändlers modelliert werden. Dazu dienen zwei Klassen `Ware` und `Kunde`. Eine `Ware` ist durch ihre *Bezeichnung* und den *Preis* charakterisiert, während ein `Kunde` einen *Namen* sowie ein Feld von Waren als *Bestellung* besitzt.

```
class Ware {
    String bezeichnung;
    float preis;        // in Euro
}
```

```
class Kunde {
    String name;
    Ware[] bestellung;

    // weiteres siehe unten
}
```

Erweitern Sie nun die Klasse `Kunde` um die folgenden Methoden:

Hinweis: Wenn Sie für eine Unteraufgabe keine Lösung gefunden haben, können Sie bei der Bearbeitung darauf aufbauender Teilaufgaben diese Methode trotzdem verwenden.

1. (3 Punkte) Erweitern Sie die Klasse `Kunde` um eine Methode

   ```
   float warenWert()
   ```

   welche den Gesamtwert aller bestellten Waren des Kunden berechnet.

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

   ```
   float versandKosten()
   ```

   welche die Versandkosten des Kunden berechnet.

   Dabei fallen bei Waren im Gesamtwert unter 10 Euro Versandkosten von 10 Euro an. Beträgt der Warenwert 100 Euro oder mehr, fallen keine Versandkosten an. In allen anderen Fällen kostet der Versand 4,50 Euro.

3. (1 Punkt) Erweitern Sie die Klasse `Kunde` um eine Methode

   ```
   boolean isStammkunde()
   ```

   die ermittelt, ob der Kunde ein Stammkunde ist. Als Stammkunde wird geführt, wer Waren für einen Gesamtwert von über 200 Euro bestellt.

4. (4 Punkte) Schreiben Sie ferner eine Methode

   ```
   Kunde[] stammkunden(Kunde[] klist)
   ```

   welche aus einem Feld von Kunden alle Stammkunden zurückliefert.

• AUFGABE 7 (10 Punkte) Magische Quadrate.

1. (5 Punkte) Schreiben Sie eine Methode

   ```
   int[][] makeSquare(int[] A)
   ```

   welche ein Feld ganzer Zahlen in eine (größtmögliche) quadratische Matrix umwandelt.

   Dabei sollen die Elemente des Array in die Ergebnismatrix zeilenweise beginnend links oben (Zeile 0, Spalte 0) bis rechts unten eingetragen werden. Falls das Feld A nicht quadratisch viele Elemente enthält, sollen die zusätzlichen Arrayelemente ignoriert werden.

   Beispiel: `makeSquare( [1,2,3,4,5,6,7,8,9,10,11] )` $= \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

2. (5 Punkte) Schreiben Sie nun eine Methode

   ```
   boolean isMagic(int[][] A)
   ```

   welche testet, ob die übergebene quadratische Matrix einem (einfachen) magischen Quadrat entspricht. Bei einem einfachen magischen Quadrat ist die Summe in jeder Zeile und jeder Spalte gleich.

   Hinweis: Die Summe hängt vom Wert der Matrixelemente ab!

   Beispiel:

16	1	2	15	= Summe 34
3	14	13	4	= Summe 34
5	12	11	6	= Summe 34
10	7	8	9	= Summe 34
=	=	=	=	
34	34	34	34	