Fakultät Elektrotechnik/Informatik

Klausur Einführung in die Informatik I für Elektrotechniker 15. Juli 2004

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung
(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	8	
2	8	
3	4	
4	6	
5	4	
6	5	
7	9	
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 und **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!
- Kommentare kosten Zeit; kommentieren Sie ihr Programm nur da, wo der Code alleine nicht verständlich wäre.
- Wir weisen noch einmal darauf hin, daß die Benutzung von Taschenrechnern und anderen elektronischen Hilfsmitteln nicht gestattet ist.

Viel Erfolg!

• **AUFGABE 1 (8 Punkte) Theorie.**

1. (1 Punkt) Was verstehen Sie unter einer *Methode*?

2. (2 Punkte) Was muss alles bei einer *Methodendefinition* angegeben werden?

3. (1 Punkt) Was ist ein *Flipflop* und wozu dient es?

4. (1 Punkt) Weshalb wird heutzutage fast ausschließlich mit dem *Zweierkomplement* gearbeitet und nicht mit dem *Einerkomplement*?

5. (1 Punkt) Erläutern Sie den Unterschied zwischen einem *Halbaddierer* und einem *Volladdierer*.

6. (2 Punkte) Ein Programm benötigt für die Verarbeitung von 10 Eingabedaten 1 Sekunde, für 20 Eingabedaten 4 Sekunden und für 30 Eingabedaten 9 Sekunden.

   (a) Welchen Aufwand (in $\mathcal{O}$-Notation) würden Sie für dieses Program abschätzen? Begründen Sie Ihre Antwort.

   (b) Was müsste man tun, um Ihre Antwort auf Frage (a) nachweisen zu können?

• **AUFGABE 2 (8 Punkte) Java.**

1. (2 Punkte) Welche Werte haben die Variablen `x` und `y` nach der Ausführung des folgenden Programmteils:

```
int x = 2;
int y = 6;
while (x < y) {
    x = x + 2;
    y++;
}
// x = ?, y = ?
```

2. (2 Punkte) Nennen Sie die Unterschiede zwischen einer `while (...) {...}`-Schleife und einer `do {...} while (...)`-Schleife .

3. (1 Punkt) Schreiben Sie den Java-Code auf, der ein Objekt mit dem Namen `person` der folgenden Klasse erzeugt. (Die Person soll 32 Jahre alt sein und Wilhelm heißen.)

```
class Mensch {
    int alter;
    String name;
    Mensch (int alter, String name) {
        this.alter = alter;
        this.name = name;
    }
}
```

4. (3 Punkte) Welche Fehler enthält folgendes Java-Codefragment? Geben Sie jeweils die Zeilennummer an und beschreiben Sie den Fehler. Folgefehler werden ignoriert.

```
 1 class Auto {
 2     int kilometer = 0;
 3
 4     public static void main (String[] args) {
 5         Wohnwagen w = new Wohnwagen ("B-TB 244");
 6         w.fahren ();
 7         Auto a = new Auto ();
 8         a.kilometer = 10000.0 / a.kilometer;
 9     }
10 }
11
12 class Wohnwagen {
13     void fahren () {
14         Terminal.println ("Brumm!");
15     }
16 }
```

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

1. (2 Punkte) Wandeln Sie die reelle Zahl 0.55 in eine Dualzahl mit 4 binären Nachkommastellen um und nehmen Sie dann die Umwandlung noch einmal in Rückrichtung vor. Geben Sie den dabei auftretenden Fehler an und lassen Sie den Lösungsweg erkennen.

2. (2 Punkte) Berechnen Sie die folgenden Aufgaben unter Verwendung der Zweikomplementdarstellung mit 4 Bit. Lassen Sie den Lösungsweg erkennen. Wo findet ein Über- bzw. Unterlauf statt? Woran erkennen Sie einen Über- bzw. Unterlauf?

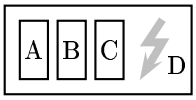
   (a) $7 + 3$

   (b) $6 + (-2)$

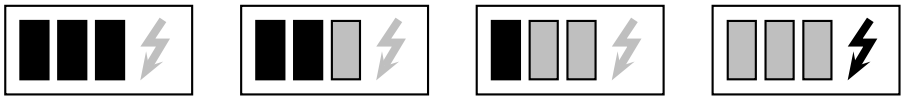
   (c) $-5 + (-5)$

• AUFGABE 4 (6 Punkte) Schaltungen.

1. (3 Punkte) Entwickeln Sie eine Schaltung zur Steuerung einer Batterieanzeige, wie sie z.B. in Mobiltelefonen verwendet wird, um den Ladezustand der Batterie darzustellen. Die Anzeige besteht aus drei Balkensegmenten A, B und C sowie einem Blitzsymbol D.



Die Anzeige soll folgende vier Zustände annehmen, die durch die Eingabesignale x und y gesteuert werden (grau dargestellte Segmente sind ausgeschaltet, schwarz dargestellte eingeschaltet):

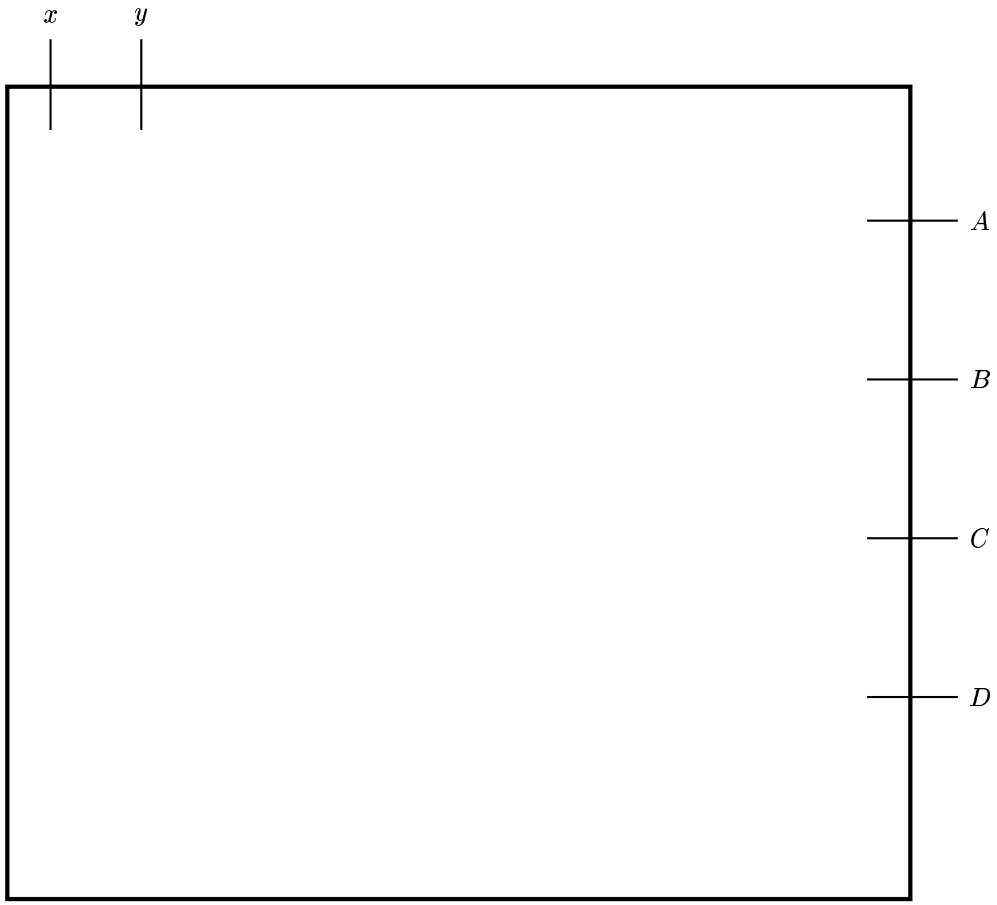


$x = 1, y = 1$ $x = 1, y = 0$ $x = 0, y = 1$ $x = 0, y = 0$

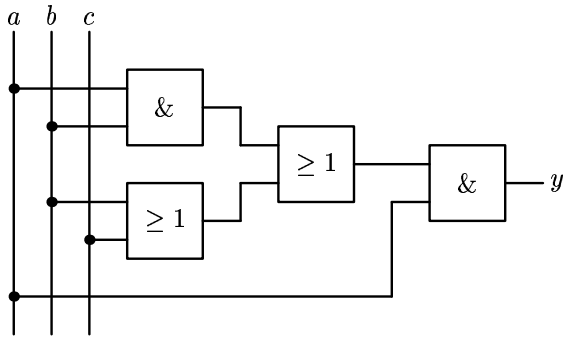
Die Ausgabesignale der Schaltung sollen die Bezeichnungen des jeweils angesteuerten Segments bekommen.

Stellen Sie zunächst eine Wertetabelle auf:

Vervollständigen Sie nun die entsprechende Schaltung:



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



3. (1 Punkt) Geben Sie eine äquivalente, aber einfachere Schaltung für die Wertetabelle der vorigen Aufgabe an.

• AUFGABE 5 (4 Punkte) Folgenberechnung.

Gegeben sei folgende rekursive Definition der Folge f:

$$\begin{aligned} f(0) &= 1 \\ f(n) &= 1 + \sum_{i=0}^{n-1} f(i) \qquad\qquad n > 0 \end{aligned}$$

Eine Folge kann in Java als Methode mit einem `int`-Parameter programmiert werden. Da der Zahlbereich von `int` auch negative Zahlen umfaßt, erweitern wir die Definition von f:

$$f(-n) = \frac{1}{f(n)}$$

Schreiben Sie eine Methode `double f(int n)`, welche $f(n)$ berechnet.

• **AUFGABE 6 (5 Punkte) Array-Operationen.**

1. (2 Punkte) Schreiben Sie eine Methode

   `float[] jedeZweite (float[] f)`

   welche ein Array zurückgibt, das nur jedes zweite Element aus `f` enthält.

   Beispiel:

   `jedeZweite({3.14f, -11.5f, 2.7f, 3.9f, 4.6f, 6.8f})` $\rightsquigarrow$ `{-11.5f, 3.9f, 6.8f}`

2. (3 Punkte) Schreiben Sie eine Methode

   `double[] mittel (double[] d)`

   welche zwischen je zwei Elemente des Arrays `d` den Mittelwert der beiden Nachbarelemente einfügt.

   Beispiel:

   `mittel({3.0, 5.0, 2.0, 6.0})` $\rightsquigarrow$ `{3.0, 4.0, 5.0, 3.5, 2.0, 4.0, 6.0}`

• AUFGABE 7 (9 Punkte) Polygone.

Ein Polygon mit den n Eckpunkten $p_0, \ldots, p_{n-1}$ kann in Java durch zwei gleichlange Arrays dargestellt werden. Zur Darstellung von Polygonen ist folgende Klassendefinition gegeben:

```
class Polygon {
  double[] x;
  double[] y;
}
```

Die x-Koordinate des Punktes p_i steht dabei im Array `x` an der Stelle i; die y-Koordinate an der entsprechenden Stelle im Array `y`.

1. (3 Punkte) Erweitern Sie die Klasse `Polygon` um folgende Methoden:
   - `Polygon(int n)` — erzeugt ein Polygon mit n Eckpunkten. Die einzelnen Koordinaten müssen dabei nicht initialisiert werden.
   - `int ecken()` — liefert die Anzahl der Eckpunkte.
   - `void setEcke(int i, double px, double py)` — setzt die Koordinaten des Eckpunktes p_i.

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

   ```
   boolean kreuztX()
   ```

   Diese soll `true` zurückliefern, falls eine Seite des Polygons die x-Achse kreuzt. Dies ist dann der Fall, wenn es mindestens einen Eckpunkt mit positiver und einen Eckpunkt mit negativer y-Koordinate gibt.

3. (4 Punkte) Erweitern Sie die Klasse `Polygon` um folgende Methoden:

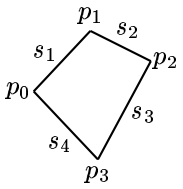
   - `double abstand(int i, int j)` — liefert den Abstand zwischen den Punkten p_i und p_j.

     Dieser berechnet sich zu $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

   - `double seitenlaenge(int i)` — liefert die Länge der i-ten Seite des Polygons.

     Die i-te Seite ist dabei definiert als die Strecke $\begin{cases} (p_{i-1}, p_i) & \text{für } 1 \leq i < n \\ (p_{n-1}, p_0) & \text{für } i = n \end{cases}$

     Beispiel: Bei einem viereckigen Polygon sind die Eckpunkte p_j und Seiten s_i wie im Bild dargestellt definiert:

   - `boolean gleichseitig()` — liefert `true`, falls alle Seiten des Polygons gleich lang sind.

     Sie können dabei voraussetzen, dass $n \geq 1$ gilt.

   Hinweis: Sie dürfen eine Methode in darauffolgenden Aufgabenteilen auch dann benutzen, wenn Sie für diese Methode keine eigene Lösung angegeben haben.