Fachbereich Informatik

Klausur Einführung in die Informatik I für Elektrotechniker 21. April 2001

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung
(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	6	
2	7	
3	4	
4	2	
5	8	
6	8	
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.
- 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 nicht gestattet ist.

Viel Erfolg!

• **AUFGABE 1 (6 Punkte) Theorie.**

1. (3 Punkte) Was bedeuten die Begriffe *Objekt*, *Klasse*, *Typ* und *Wert* und wie stehen Sie zueinander in Beziehung?

2. (1 Punkt) Was verstehen Sie unter *Attributen* ?

3. (2 Punkte) Die Methode `int m1(int n)` habe linearen Aufwand $O(n)$, die Methode `int m2(int n)` habe quadratischen Aufwand $O(n^2)$. Welcher der beiden Funktionsaufrufe `m1(1)` bzw. `m2(1)` liefert schneller das Ergebnis?

   o `m1(1)`

   o `m2(1)`

   o Die Angaben reichen nicht aus, um diese Frage zu beantworten.

   Begründen Sie Ihre Antwort!

• **AUFGABE 2 (7 Punkte) JAVA.**

1. (2 Punkte)

   Gegeben sei die Methode

```
int foo(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * foo(n - 1);
    }
}
```

   Terminiert diese Methode immer?

   o Ja.

   o Nein.

   o Die Angaben reichen nicht aus, um diese Frage zu beantworten.

   Begründen Sie Ihre Antwort!

2. (2 Punkte) Geben Sie den Aufwand der Methoden `f` und `g` in Abhängigkeit der Parameter `i` bzw. `n` in *Big-O-Notation* an.

```
int f(int i) {
    int res = 0;
    do {
        res = res + 1;
        i = i - 1;
    } while (i > 0);
    return res;
}

int g(int n) {
    int i, res = 0;
    for (i = 0; i < n; i++) {
        res = res + f(2);
    }
    return res;
}
```

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

```
1 String s = "NULL";
2 byte b = 11;
3 final int i = b / 2;
4 double[] d = new double[b];
5 for (i = b; i > 0; i--) {
6     d[i] = 2.0;
7     s = s + 1;
8 }
9 b = i * 2;
```

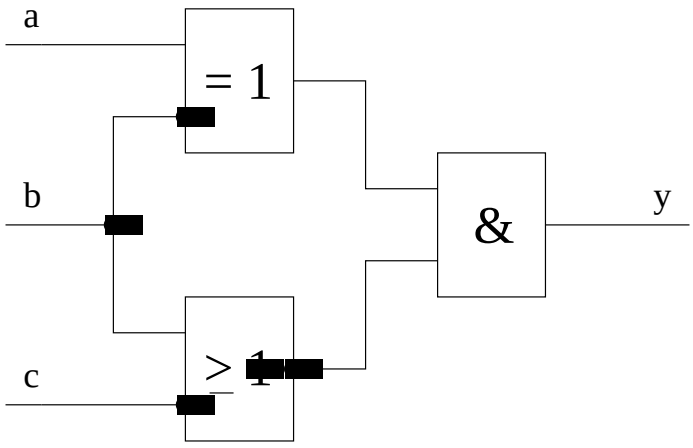
• **AUFGABE 3 (4 Punkte) Zahlsysteme.**

1. (1 Punkt) Wandeln Sie folgende Dezimalzahlen in Dualzahlen um und führen Sie die Berechnungen im Dualsystem aus. Lassen Sie den Lösungsweg erkennen.
   - $10 + 6$
   - $17 + 15$

2. (3 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*?
   - $5 + 6$
   - $1 - 7$
   - $-2 - 4$

• **AUFGABE 4 (2 Punkte) Schaltungen.**

Stellen sie die Wertetabelle für die folgende Schaltung auf.



• **AUFGABE 5 (8 Punkte) Binomischer Lehrsatz.**

1. (4 Punkte) Der Binomialkoeffizient $\binom{n}{k}$ ist darstellbar als

$$\binom{n}{k} = \frac{n*(n-1)*(n-2)*\ldots*(n-k+1)}{k*(k-1)*(k-2)*\ldots*1} = \prod_{i=0}^{k-1} \frac{n-i}{k-i}$$

   Schreiben Sie eine *JAVA*-Methode

   ```
   long binom(long n, long k)
   ```

   welche den Binomialkoeffizienten ohne Verwendung von Rekursion oder der Fakultätsfunktion berechnet.

2. (4 Punkte) Aus dem binomischen Lehrsatz ergibt sich als Speziallfall die Gleichung

$$2^n = \sum_{k=0}^{n} \binom{n}{k}$$

   Schreiben Sie eine Methode

   ```
   boolean test(long n)
   ```

   welche die obige Gleichung für eine natürliche Zahl `n` überprüft.

   Ihre Methode `test` soll also das Ergebnis `true` liefern, wenn für den Parameter `n` die Summe $\sum_{k=0}^{n} \binom{n}{k}$ sowie 2^n den selben Wert besitzen, `false` anderenfalls. Falls `n` eine negative Zahl ($n < 0$) ist, soll ebenfalls `false` als Ergebnis zurückgegeben werden.

• AUFGABE 6 (8 Punkte) Operationen auf Feldern.

1. (4 Punkte) Schreiben Sie eine Methode

   ```
   int[] scan(int[] A)
   ```

   welche die Teilsummen aller Präfixe eines Feldes berechnet.

   Beispiele:

   - `scan( [1] ) = [1]`
   - `scan( [1, 7] ) = [1, 1+7] = [1, 8]`
   - `scan( [1, 7, -2] ) = [1, 1+7, 1+7-2] = [1, 8, 6]`
   - `scan( [1, 7, -2, 4] ) = [1, 1+7, 1+7-2, 1+7-2+4] = [1, 8, 6, 10]`

2. (4 Punkte) Schreiben Sie eine Methode

   ```
   int[] filtereven(int[] A)
   ```

   welche alle *geraden* Zahlen in einem Feld zurückliefert (und die ungeraden Zahlen entfernt).

   Beispiel:

   - `filtereven( [1, 4, -8, -5, 12] ) = [4, -8, 12]`

• **AUFGABE 7 (9 Punkte) Matrizen.**

1. (5 Punkte) Schreiben Sie eine Methode

   ```
   double[][] insertcolumn(double[][] M, double[] V, int column)
   ```

   welche einen Vektor `V` in eine Matrix `M` an Spalte `column` einfügt.

   Dabei bezeichnet `column` den Index der Spalte, welche die Elemente von `V` in der Ergebnismatrix enthalten soll, die restlichen Spalten der Ausgangsmatrix sind gegebenfalls entsprechend zu verschieben.

   Der Spaltenindex `column` kann somit zwischen 0 und der Spaltenzahl von `M` liegen. Falls er außerhalb dieses Bereiches liegt, oder die Zeilenzahl von `M` und `V` verschieden sind, das Ergebnis der Operation `insertcolumn` also keine korrekte Matrix mehr wäre, soll die Matrix `M` unverändert zurückgegeben weden.

   Beispiel: Für $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ und $V = [10, 20]$ ergibt sich

   - `insertcolumn(M, V, 0)` $= \begin{pmatrix} 10 & 1 & 2 & 3 \\ 20 & 4 & 5 & 6 \end{pmatrix}$
   - `insertcolumn(M, V, 2)` $= \begin{pmatrix} 1 & 2 & 10 & 3 \\ 4 & 5 & 20 & 6 \end{pmatrix}$

2. (4 Punkte) Schreiben Sie eine Methode

   ```
   double[][] transpose(double[][] M)
   ```

   zum Transponieren einer Matrix, in der Sie mittels obiger Methode `insertcolumn` schrittweise die einzelnen Zeilen der Ausgangsmatrix `M` als Spalten der Transpositionsmatrix hinzufügen.

   Beispiel: `transpose` $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$