Friedrich-Alexander-Universität Erlangen-Nürnberg

Schriftliche Prüfung zur Vorlesung
„Einführung in die Programmierung für Fertigungstechniker”

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Gesamt
40	10	10	10	10	80

(Name) (Vorname) (Matrikel-Nr.)

(Fachrichtung) (Semester)

Ihre Scheine werden direkt an das Prüfungsamt gesandt!

Erlangen, den 04.02.1995

Organisatorische Hinweise

Die folgenden Informationen bitte aufmerksam lesen und die Erklärung am Blattende unterschreiben.

- Die Bearbeitungszeit beträgt 120 Minuten.
- Es sind **k e i n e** Hilfsmittel zugelassen.
- Schmierpapier darf nicht mit abgegeben werden. Die Lösung einer Aufgabe soll auf das Aufgabenblatt in den dafür vorgesehenen freien Raum geschrieben werden, der jeweils der Aufgabe folgt. Sollte der Platz nicht ausreichen, so können Sie die Rückseiten der Aufgabenblätter mitverwenden.
- Bei Bedarf sind von der Aufsicht Schmierpapier (farbig) und zusätzliche Lösungsblätter (weiß) zu erhalten. Vermerken Sie vor dessen Verwendung unbedingt Ihren Namen und Ihre Matrikelnummer darauf!
- Fragen zu den Prüfungsaufgaben können grundsätzlich nicht beantwortet werden.
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studienfachrichtung und Fachsemesterzahl auf dem Deckblatt der Klausur ein.
- Bitte legen Sie Ihren Studenten- und einen Lichtbildausweis zur Kontrolle bereit.
- Sie dürfen den Raum nicht verlassen, bevor Ihre Personalien überprüft wurden, und Sie die Klausurunterlagen der Aufsicht zurückgegeben haben.
- Bleiben Sie bitte am Ende der Bearbeitungszeit solange an Ihrem Platz sitzen, bis alle Klausurunterlagen eingesammelt sind.

Erklärung

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen (8 Seiten inkl. Deckblatt)
- die Kenntnisnahme der obigen Informationen.

Ich bin damit einverstanden, daß mein Prüfungsergebnis unter Angabe der Matrikel-Nr. veröffentlicht wird.

☐ ja ☐ nein

Erlangen, den 04.02.1995

(Unterschrift)

Aufgabe 1: (40 Punkte)

Schreiben Sie ein Programm **calc**, welches zwei beim Programmaufruf in der Kommandozeile (Achtung: **nicht** von der Standardeingabe einlesen!!!) übergebene Fließkomma-Zahlen (*operand1* und *operand2*) gemäß eines zusätzlich übergebenen Operators (*operator*) miteinander verknüpft und die Aufrufargumente sowie das Ergebnis (*ergebnis*) der Verknüpfung auf die Standardausgabe ausgibt.

Beispiel: Bei Aufruf Ihres Programms mittels

calc *operand1 operand2 operator* (also z.B.: **calc** 4.5 7 +)

soll Ihr Programm

operand1 operator operand2 = ergebnis (hier also: 4.5 + 7 = 11.5)

ausgeben und terminieren.

Ihr Programm soll folgende Operatoren erkennen und entsprechend handeln (mathematische Funktionen der C-Library wie **pow()** o.ä. dürfen bei der Implementierung von Ihnen **nicht** verwendet werden!):

- \+ : Berechnen von *operand1* + *operand2*
- \- : Berechnen von *operand1* - *operand2*
- * : Berechnen von *operand1* * *operand2*
- / : Berechnen von *operand1* / *operand2*
- ^ : Berechnen von $operand1^{operand2}$ ($operand2 \geq 0$, ganzzahlig)

Zur Wandlung einer in der Aufrufzeile gegebenen Ziffernfolge in eine Fließkomma-Zahl kann die Funktion **double atof(char *str)** verwendet werden (wobei **str** einen Zeiger auf den zu konvertierenden String darstellt).

Bei Fehlaufrufen (z.B. fehlendes Argument, unerlaubter Operator, etc.) soll Ihr Programm mit einer Fehlermeldung und einem Fehlercode ungleich null terminieren.

Erstellen Sie zuerst ein Struktogramm und dokumentieren Sie Ihr Programm mit kurzen, aber aussagekräftigen Kommentaren! Auf das Struktogramm entfällt die Hälfte der Punkte!

Aufgabe 2: (10 Punkte)

Was berechnet das folgende (sinnlose) Programm? Geben Sie die Inhalte der verwendeten Variablen nach jeder ausgeführten Anweisung an! Verwenden Sie dazu die untenstehende Tabelle!

```
int k[] = { 1, 0, 1, 0 };                (1)
void f(int *k)                           (2)
{  int m = *k-2;                         (3)
   do {                                  (4)
      m       =  k[1];                   (5)
      *(k+1) += *k;                      (6)
      k++;                               (7)
   }while(m > 0);                        (8)
   (*k)++;                               (9)
}                                        (10)
void main()                              (11)
{  int *m = k;                           (12)
   f(m);                                 (13)
   m++;                                  (14)
   f(m);                                 (15)
}                                        (16)
```

Zeile							

Aufgabe 3: (10 Punkte)

Markieren und verbessern Sie die Fehler in dem aus Kernighan/Ritchie "Programmieren in C" bekannten Programm, welches einen Text über die Standardeingabe einliest und die Zahl der Zeichen, Worte (ein Wort wird hier definiert als Zeichenfolge, die keine Leerzeichen, Tabulatorzeichen bzw. Zeilentrenner enthält) und Zeilen darin zählt und ausgibt.

Als Fehler markierte korrekte Teile führen zu Punktabzug. Die Tatsache, daß das Programm keine Kommentare enthält und kein Struktogramm beigefügt ist, wird ausnahmsweise nicht als Fehler betrachtet.

```
#include <stdio.h>

#define OUT 0;

int main(void)
{
    int c, nl, nw, state;

    state = OUT;
    nl = nw = nc = 0;

    while((c=getchar() != EOF) {
       ++nc;
       if ( c == '\n' )
         ++nl;
       if ( c == ' ' || c == '\n' || c == '/t' );
         sate = OUT;
       else
         if ( state = OUT )
            state = IN;
            +nw;
    }

    printf("Zeilen: %d Worte: %d Zeichen: %d\n",nl,nw,nc);

    return 0;
}
```


Aufgabe 4: (10 Punkte)

Geben Sie für die folgenden **for**-, **while**- und **do-while**-Schleifen jeweils an, wie oft und mit welchen Werten der Schleifenvariablen diese jeweils durchlaufen werden!

```
int j,k;
```

a)
```
for(j = 10; j > 4; j -= 2) { ..... }
```

b)
```
k=8;
for(j = 8; j <= k; j++) { k=10; ..... }
```

c)
```
j=4;
while(++j < 8) { ..... }
```

d)
```
j=10;
do { ..... }while(j < 10);
```

Aufgabe 5: (10 Punkte)

Entwerfen Sie geeignete Datenstrukturen zur sortierten Speicherung von Personaldaten (ein Personaldatensatz besteht hier vereinfacht aus **Name**, **Vorname**, **Personalnummer** und **Gehalt** (z.B.: "Mustermann", "Hans", 4711, 2345.67)) in Form eines Binärbaumes (sortiert wird lexikographisch nach Namen). Gehen Sie dabei davon aus, daß Strings nicht mehr als 20 Zeichen enthalten, und daß maximal 100 Personaldatensätze zu speichern sein werden!