Ingenieurinformatik – Diplom-FA
(Teil 2, C-Programmierung)

Name	Vorname	Matrikelnummer	Sem.-Gr.:	Hörsaal	Platz

Zulassung geprüft:

_____

(Grundlagenteil)	Aufgabe 1	Aufgabe 2	Aufgabe 3	Summe

Die Prüfung ist nur dann gültig, wenn Sie die erforderliche Zulassungsvoraussetzung erworben haben (erfolgreiche Teilnahme am Praktikum). Dies wird vom Aufgabensteller überprüft.

FA-Diplom

Aufgabensteller: Dr. Reichl, Dr. Küpper und Kollegen

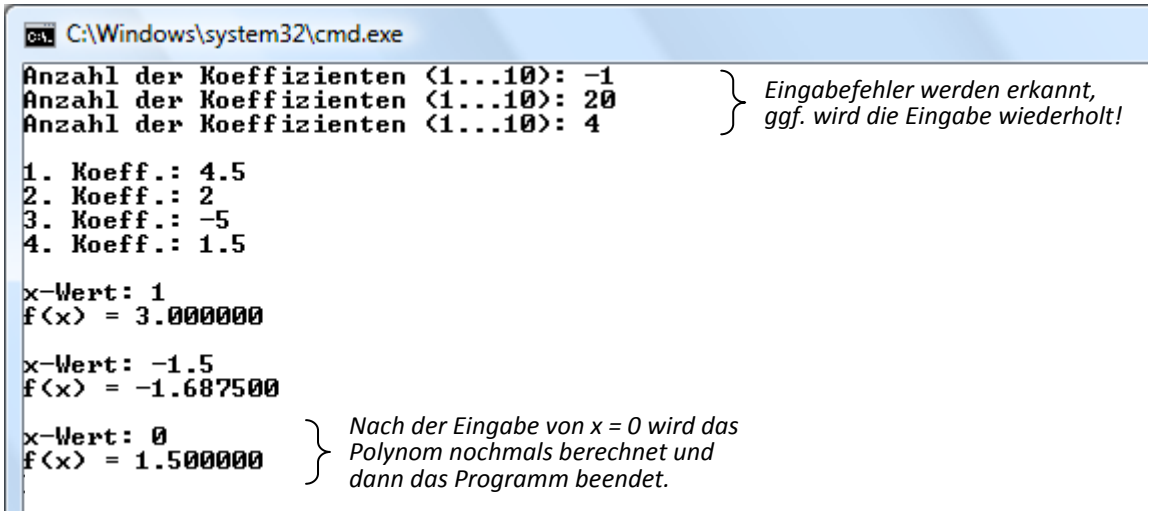
Bearbeitungszeit: 60 Minuten

Hilfsmittel:
- Taschenrechner nicht zugelassen
- PC/Notebook nicht zugelassen
- Sonstige eigene Hilfsmittel sind erlaubt
- Bearbeitung mit Bleistift ist erlaubt

Aufgabe 1: (ca. 24 Punkte)

Schreiben Sie ein C-Programm zur Berechnung der Funktionswerte von reellen Polynomen:

- Nach dem Start des Programms muss der Anwender die Anzahl der Koeffizienten des Polynoms eingeben (maximal 10).
- Wenn eine ungültige Anzahl eingegeben wird (kleiner als null oder größer als 10), dann soll die Eingabe wiederholt werden – siehe Bildschirmfoto!
- Es folgt die Eingabe aller Koeffizienten. Auf die Überprüfung von Eingabefehlern (zum Beispiel: Eingabe von Buchstaben statt Zahlen) wird hier verzichtet.
- Schließlich kann der Anwender beliebig viele x-Werte eingeben, für die der Wert des Polynoms jeweils berechnet wird. Wenn der Anwender als x-Wert null eingibt, wird zunächst der Wert des Polynoms an der Stelle x = 0 berechnet und dann das Programm beendet.
- Achten Sie darauf, dass alle Eingabeaufforderungen so wie in der Abbildung gezeigt auf dem Bildschirm ausgegeben werden.



Im abgebildeten Beispiel wird der Wert des Polynoms $y = 4{,}5 \cdot x^3 + 2 \cdot x^2 - 5 \cdot x + 1{,}5$ an den Stellen x = 1 und x = -1,5 berechnet. Danach gibt der Anwender den Wert x = 0 ein, das Programm berechnet nochmals den Wert des Polynoms für x = 0 und wird dann beendet.

Aufgabe 2: (ca. 21 Punkte)

Das folgende C-Programm berechnet die reellen und komplexen Nullstellen der quadratischen Gleichung $x^2 + px + q = 0$. Vor Beginn der Berechnung wird der Anwender zur Eingabe der Parameter p und q aufgefordert. Nach der Ergebnisausgabe wird der Anwender gefragt, ob er eine weitere Berechnung wünscht (Eingabe von 1) oder das Programm beenden möchte (Eingabe von 2).

```
#include <studio.h>
#include <math.h>

int main(void)
{
    int weiter;
    double p, q, w, x;
    do
    {
        printf("p: "); scanf("%lf", &p);
        printf("q: "); scanf("%lf", &q);
        x = -p / 2.0; w = x * x - q;
        if(w == 0)
        {
            printf("x = %d\n", x);
        }
        else if(w >= 0)
        {
            printf("x1 = %f\n", x + sqrt(w));
            printf("x2 = %f\n", x - sqrt(w));
        }
        else
        {
            printf("x1 = %f + %f i\n", x, sqrt(-w));
            printf("x2 = %f - %f i\n", x, sqrt(-w));
        }

        do
        {
            printf("Nochmal (1) oder Ende (2)?");
            scanf("%d", &weiter);
        }
        while(weiter < 1 && weiter > 2)
    }
    while(weiter == 1);
}
return 0;
```

2.1. Korrigieren Sie die fünf Fehler, die sich in den Quelltext eingeschlichen haben.

2.2. Zeichnen Sie für das (korrigierte) Hauptprogramm ein Struktogramm, welches den genauen Ablauf der Funktion **main** beschreibt.

2.3. Warum nennt man die hier verwendete do-while-Schleife auch „nicht-abweisende Schleife“?

int-Variable definieren:	weiter
double-Variablen definieren:	p, q, w, x

Aufgabe 3: (ca. 22 Punkte)

```c
#include <stdio.h>
void hex2bin(char *hex);

int main(void)
{
    hex2bin("FF"); hex2bin("100"); hex2bin("0");
    return 0;
}

void hex2bin(char *hex)
{
    int idx;
    for(idx = 0; hex[idx] != 0; ++idx)
    {
        switch(hex[idx])
        {
        case '0': printf("0000"); break;
        case '1': printf("0001"); break;
        case '2': printf("0010"); break;
        case '3': printf("0011"); break;
        case '4': printf("0100"); break;
        case '5': printf("0101"); break;
        case '6': printf("0110"); break;
        case '7': printf("0111"); break;
        case '8': printf("1000"); break;
        case '9': printf("1001"); break;
        case 'A': printf("1010"); break;
        case 'B': printf("1011"); break;
        case 'C': printf("1100"); break;
        case 'D': printf("1101"); break;
        case 'E': printf("1110"); break;
        case 'F': printf("1111"); break;
        }
    }
    printf("\n");
}
```

3.1. Wie lautet die Ausgabe des nebenstehenden C-Programms?

3.2. Mittels **sizeof(x)** kann die Anzahl der Speicherzellen (Bytes) ermittelt werden, die von der Variablen x im Hauptspeicher des Rechners belegt werden. Wie lautet die Ausgabe des folgenden C-Programms, wenn es auf einem Windows-PC im Praktikumsraum ausgeführt wird?

Ausgabe des Programms:

```c
#include <stdio.h>

int main(void)
{
    int i;
    float f;

    printf("%d\n", sizeof(i));
    printf("%d\n", sizeof(f));
    return 0;
}
```

3.3. Zwei Vektoren werden als **double v1[3][2];** und als **char str[10];** definiert. Wie viele Bytes belegen diese beiden Vektoren im Hauptspeicher des Rechners?

Vektor **v1**: ..... Bytes Vektor **str**: ..... Bytes

3.4. Werden in einer Matrix A die Zeilen gegen die Spalten vertauscht (also die Matrixelemente an der Hauptdiagonalen gespiegelt), erhält man die transponierte Matrix A^T.

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix} \qquad A^T = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}$$

Beispiel: transponierte 3x3-Matrix

In einem C-Programm sind die Matrizen a und b global definiert und bereits mit Werten belegt. Schreiben Sie eine Funktion **check_trans**. Diese Funktion soll überprüfen, ob a = b^T ist, also ob die Matrix **a der transponierten Matrix b^T** entspricht (Rückgabewert = 1) oder nicht (Rückgabewert = 0).

```
#define DIM 10
int a[DIM][DIM], b[DIM][DIM];

int check_trans(void)
{









}
```

***** *Viel Erfolg!!!* *****

(Platz für Notizen und Nebenrechnungen)