School of Engineering

INE1
HS 2014

Semesterendprüfung

Dozenten:	A. Aders, E. Bazzi	Studiengänge:	Elektrotechnik, Systemtechnik
Klassen:	ET14a,b,t	Datum:	13.Januar 2015
	ST14a,b	Zeit:	8:00 – 9:30 Uhr

Name, Vorname	Klasse	Punkte	Note

Zeit	90 Minuten
Maximale Punktzahl	60; alle fünf Aufgaben haben dasselbe Gewicht (je 12 Punkte)
Hilfsmittel	Zusammenfassung maximal zwei Blätter A4 beidseitig beschrieben keine anderen Hilfsmittel erlaubt, insbesondere keinerlei elektronische Hilfsmittel
Abgabe	Füllen Sie dieses erste Aufgabenblatt aus und schreiben Sie alle **Lösungsblätter** mit Ihrem **Namen, Vornamen** und Ihrer **Klasse** sowie der entsprechenden **Aufgabennummer** an. Geben Sie alle Aufgaben- und Lösungsblätter ab. Bei einigen Aufgaben können Sie Ihre Lösung direkt auf das Aufgabenblatt schreiben.
Hinweise	Lesen Sie alle Aufgabenstellungen sorgfältig durch, bevor Sie mit der Bearbeitung der für Sie leichtesten Aufgabe beginnen. Bitte schreiben Sie weder mit Bleistift noch mit roter Farbe.

Bewertung

Aufgabe	1	2	3	4	5	Total
Punkte						

Aufgabe 1: Zeichen: Gross- & Kleinbuchstaben [12 Punkte]

Schreiben Sie eine C-Funktion **`toUppercase(...)`**, die in einem als Parameter übergebenen String alle Kleinbuchstaben 'a' bis 'z' durch die entsprechenden Grossbuchstaben 'A' bis 'Z' ersetzt. Alle anderen Zeichen, insbesondere die übrigen Buchstaben (d.h. Umlaute und andere Buchstaben mit diakritischen Zeichen, 'é', 'ç', 'ñ', usw.) sollen unverändert belassen werden. Sie können die Tatsache ausnützen, dass diese Buchstaben in den ASCII- und Unicode-Zeichensätzen jeweils zusammenhängend in derselben Reihenfolge hintereinander stehen.

a) Prototyp (Aufrufschnittstelle, Signatur) dieser Funktion [2 Punkte]

.....

b) Beschreibung der Lösung, evtl. mit Struktogramm oder Pseudocode [3 Punkte]

.....

.....

.....

c) Codieren Sie die Funktion [6 Punkte]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

d) Schreiben Sie einen Aufruf dieser Funktion, wie er z.B. in Ihrem Test/-funktion stehen könnte. [1 Punkt]

.....

Aufgabe 2: Basisumrechnung ganzer Zahlen – Strings [12 Punkte]

Schreiben Sie eine C-Funktion

```
char * intToString ( int i, const int b, char *s );
```

die zu einem als Parameter übergebenen **`int`** ***i*** die Darstellung zur Basis ***b*** als *String* zurückgibt; auch ***b*** wird als **`int`**-Parameter übergeben. Prüfen Sie, ob 1<***b***<37 ist, wenn nicht, soll **`intToString(...)`** einen String der Länge 0 zurückgeben. Die Funktion **`intToString(...)`** muss das Vorzeichen nicht behandeln, d.h. kann voraussetzen, dass ***i*** nicht negativ ist.

Hinweis: bei einer möglichen Lösung wird zuerst die *Länge* (Anzahl Stellen) der gesuchten Darstellung (von ***i*** zur Basis ***b***) bestimmt; dies ist aber nicht die einzige Möglichkeit.

a) Was bedeutet der Parameter `char *s`? Warum ist er notwendig? [0,5 Punkt]

.....

b) Beschreibung der Lösung, evtl. mit Struktogramm oder Pseudocode [3,5 Punkte]

.....

.....

.....

c) Codieren Sie die Funktion [6 Punkte]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

d) Schreiben Sie einen Aufruf dieser Funktion, wie er z.B. in Ihrem Test / Testfunktion stehen könnte (inkl. Deklaration und Initialisierung der benötigten Variablen). [2 Punkte]

.....

.....

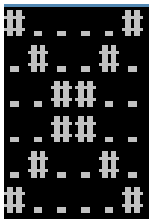
.....

Aufgabe 3: 2D-Arrays und Bit-Arithmetik [12 Punkte]

Die Funktion **`zeichne(…)`** erzeugt aus dem 2D Array **`a[][]`** die daneben stehende Ausgabe auf dem Bildschirm. Werte grösser als Null werden als Raute, der Wert Null hingegen wird als Punkt ausgegeben.

```
char a[X][Y] = {
        {1,0,0,0,0,1},
        {0,1,0,0,1,0},
        {0,0,1,1,0,0},
        {0,0,1,1,0,0},
        {0,1,0,0,1,0},
        {1,0,0,0,0,1},
        };
```

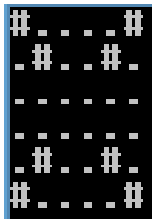
Ausgabe von `zeichne(…);`



a) Ermitteln Sie Werte von **`a[][]`** damit die Ausgabe von **`zeichne(…)`** wie unten rechts erscheint. [1 Punkt, korrekte Werte]

```
char a[X][Y]={

        {___,___,___,___,___,___},

        {___,___,___,___,___,___},

        {___,___,___,___,___,___},

        {___,___,___,___,___,___},

        {___,___,___,___,___,___},

        {___,___,___,___,___,___},
};
```

Ausgabe von **`zeichne(…);`**



b) Gibt es mehrere Lösungen? Wenn ja beschreiben Sie sie. [2 Punkte, korrekte Antwort]

c) Die Funktion **aAndB(…)** verknüpft die jeweiligen Elemente des **Array a** mit den zugehörigen Elementen des **Array b** auf Bit-Ebene mit der **&** Operation.
Das Resultat wird in **Array a** geschrieben (d.h. Es wird für jedes Element der Arrays die Operation **a[m][n] = a[m][n] & b[m][n]** durchgeführt).

```
char a[X][Y] = {
            {1,0,0,0,0,1},
            {0,1,0,0,1,0},
            {0,0,1,1,0,0},
            {0,0,1,1,0,0},
            {0,1,0,0,1,0},
            {1,0,0,0,0,1}
            };
```

```
char b[X][Y] = {
            {1,2,3,3,2,1},
            {1,2,3,3,2,1},
            {1,2,3,3,2,1},
            {1,2,3,3,2,1},
            {1,2,3,3,2,1},
            {1,2,3,3,2,1}
            };
```

Welche Werte enthält der Array a nach Durchführung der Funktion aAndB?
Tragen Sie die Werte unten ein. [2 Punkte, korrekte Antwort]

```
char a[BREITE][HOEHE]={

            {___,___,___,___,___,___},

            {___,___,___,___,___,___},

            {___,___,___,___,___,___},

            {___,___,___,___,___,___},

            {___,___,___,___,___,___},

            {___,___,___,___,___,___},
    };
```

d) Anschliessend finden Sie ein“Skelett“ des Programms mit den Funktionen **`main()`** und **`aAndB()`**. Bitte vervollständigen Sie das Programm.

[7 Punkte, korrekte Kopfzeile, Logik, Returnwerte, Syntax]

```
#include <stdio.h>
#define BREITE 6
#define HOEHE 6

______  aAndB (_____________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________


int main (void) {
  char a[BREITE][HOEHE] = {
         {1,0,0,0,0,1}, {0,1,0,0,1,0}, {0,0,1,1,0,0},
         {0,0,1,1,0,0}, {0,1,0,0,1,0}, {1,0,0,0,0,1}    };

  char b[BREITE][HOEHE] = {
         {1,2,3,3,2,1}, {1,2,3,3,2,1}, {1,2,3,3,2,1},
         {1,2,3,3,2,1}, {1,2,3,3,2,1}, {1,2,3,3,2,1}    };

  zeichne (a);
  // Funktionsaufrufes mit Übergabe von a und b

  _______ aAndB (_____________________________________________);

  zeichne (a);
  return 0;
}
```

Aufgabe 4: Steuerprogression (Arrays) [12 Punkte]

Die direkte Bundessteuer (Einkommenssteuer) ist wie in den meisten Ländern progressiv, d.h. hohe Einkommen werden stärker besteuert als tiefe Einkommen.

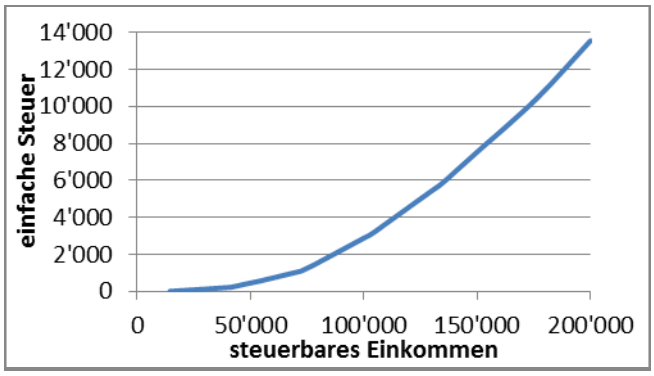
Die Progression ist in einer Tabelle festgelegt (Beispiel Tarif A für Alleinstehende):

Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 14'500	übersteigen	Fr. 0.77
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 31'600	übersteigen	Fr. 0.88
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 41'400	übersteigen	Fr. 2.64
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 55'200	übersteigen	Fr. 2.97
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 72'500	übersteigen	Fr. 5.94
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 78'100	übersteigen	Fr. 6.60
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 103'600	übersteigen	Fr. 8.80
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 134'600	übersteigen	Fr. 11.00
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 176'000	übersteigen	Fr. 13.20
Die einfache Steuer beträgt je Fr. 100 steuerbares Einkommen, die	Fr. 755'200	übersteigen	Fr. 11.50

D.h. von Fr. 14‘500 bis Fr. 31‘600 beträgt die Steuer Fr. 0.77 pro Fr. 100 Einkommen,
von Fr. 31‘600 bis Fr. 41‘400 beträgt die Steuer Fr. 0.88 pro Fr. 100 Einkommen, usw.
Für die Berechnung der Steuer wird das steuerbare Einkommen auf ganze Fr. 100 abgerundet.

Daraus ergeben sich an den Stufengrenzen folgende Steuerbeträge:

Einkommen	Steuer
14'500	0.00
31'600	131.67
41'400	217.91
55'200	582.23
72'500	1'096.04
78'100	1'428.68
103'600	3'111.68
134'600	5'839.68
176'000	10'393.68
755'200	86'848.08



Schreiben Sie eine C-Funktion, die zu einem als **`int`**-Parameter übergebenen steuerbaren Einkommen den Steuerbetrag zurückgibt *in Rappen ganzzahlig* gemäss obigen Tabellen.

Hinweis:

Verwenden Sie Arrays für die Einkommensgrenzen der Progressionsstufen, die Steuersätze sowie die Steuerbeträge an den Stufengrenzen:

```
const int eg[] = { 14500, 31600, 41400, 55200, 72500,
    78100, 103600, 134600, 176000, 755200 };     // Einkommensgrenzen
const int ss[] = { 77, 88, 264, 297, 594, 660, 880, 1100, 1320, 1150 };
    // Steuersatz in Rp. pro 100.-- Fr.
int sadg [ANZ] = { 0 };    // Steuer an der Grenze -- initialisieren!!
```

Die Werte im dritten Array berechnen sich aus den Werten in den ersten beiden Arrays. Zwischen den Stufengrenzen ist die Steuer eine lineare Funktion. Diese Arrays können Sie als globale Variable deklarieren und z.B. im Hauptprogramm ein für allemal initialisieren. –

N.B. Falls Sie solche Arrays verwenden, gehören deren Deklaration und Initialisierung auch zur Lösung!

a) Prototyp (Aufrufschnittstelle, Signatur) dieser Funktion [2 Punkte]

.....

b) Beschreibung der Lösung, evtl. mit Struktogramm oder Pseudocode [3 Punkte]

.....

.....

.....

.....

.....

c) Codieren Sie die Funktion [7 Punkte]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Aufgabe 5: File I/O [12 Punkte]

Schreiben Sie ein Programm „KeineDoppelzeichen.c“ mit folgendenden Eigenschaften:

1. Datei einlesen: die Datei "InFile.txt" wird vom Programm eingelesen
2. Inhalt verarbeiten:
   - das Programm löscht alle doppelten (bzw. mehrfachen) Buchstaben

     (Bsp: „`Das ist eeein Test...`“ → „`Das ist ein Test.`“)
   - der verarbeitete Text wird auf dem Bildschirm ausgegeben
   - die Anzahl eingelesener, gelöschter und ausgebener Zeichen wird auf dem Bildschirm ausgegeben:

     ```
     Der Eingang hatte 92 und der Ausgang 76 Zeichen.
     16 doppelte Zeichen wurden geloescht.
     ```
3. Datei ausgeben: Die verarbeiteten Daten werden ohne Zwischenspeicherung in die Datei "OutFile.txt" geschrieben.

Verwenden Sie dazu das vorliegende Programgerüst und setzten Sie die Funktionen fopen(), fclose(), fgetc() und fputc() ein.

Aufgaben:

a) Nennen Sie die drei Standardstreams von C und beschreiben Sie kurz deren „Charakteristik“. [1 Punkt]

#0_____

#1_____

#2_____

b) Das mischen von Lese- und Schreiboperationen auf derselben Datei kann, bedingt durch die im System vorhandenen Zwischenpuffer zu unvorhergesehenen Resultaten führen. Nennen Sie zwei mögliche Massnahmen, um das Problem zu lösen. [3 Punkte]

_____

_____

_____

c) Vervollständigen Sie nun das folgende Gerüst „KeineDoppelzeichen.c“. [8 Punkte]

```
#include <stdio.h>
int main()
{
   FILE * pEingang;                  // Input Dateipointer
   FILE * pAusgang;                  // Output Dateipointer

// Definition der notwendigen Variablen




// Oeffnen von inFile.txt zum lesen; Fehlerprüfung:




// Oeffnen von outFile.txt zum schreiben; Fehlerprüfung:




// Zeichen einlesen, verarbeiten und ausgeben:




// Dateien schliessen, Statistik ausgeben:
```

Nützliche Funktionen

fputc

```
int fputc ( int character, FILE * stream );
```

Write character to stream
Writes a *character* to the *stream* and advances the position indicator.

The character is written at the position indicated by the *internal position indicator* of the *stream*, which is then automatically advanced by one.

Parameters

character
: The int promotion of the character to be written.
The value is internally converted to an unsigned char when written.

stream
: Pointer to a FILE object that identifies an output stream.

Return Value

On success, the *character* written is returned.
If a writing error occurs, EOF is returned and the *error indicator* (ferror) is set.

fgetc

```
int fgetc ( FILE * stream );
```

Get character from stream
Returns the character currently pointed by the internal file position indicator of the specified *stream*.
The internal file position indicator is then advanced to the next character.

If the stream is at the end-of-file when called, the function returns EOF and sets the *end-of-file indicator* for the stream (feof).
If a read error occurs, the function returns EOF and sets the *error indicator* for the stream (ferror).

fgetc and getc are equivalent, except that getc may be implemented as a macro in some libraries.

Parameters

Stream : Pointer to a FILE object that identifies an input stream.

Return Value

On success, the character read is returned (promoted to an int value).
The return type is int to accommodate for the special value EOF, which indicates failure:
If the position indicator was at the *end-of-file*, the function returns EOF and sets the *eof indicator* (feof) of *stream*.
If some other reading error happens, the function also returns EOF, but sets its *error indicator* (ferror) instead.