Lösung Klausur 1 infT

1. (2 Pkte.)
Geben Sie an, wie gross die Zahl - in einem C-Programm auftretend - in dezimal jeweils ist:

a) `0X33` **51** b) `073` **59** c) `01A` f: Hexziffer in Oktalzahl d) `'\021'` **17**

2. (1 Pkt.)
Welche Ausgabe ist von den untenstehenden Codefragmenten an der Konsole zu erwarten?

a)
```
printf("-%d-", strlen("Aloha"));
```
-5-

b)
```
char str[10] = {'b', 'o', 'n', 'j', 'o', 'u', 'r'};
printf("-%d-", strlen(str));
```
-7- Rest des Arrays mit 0 initialisiert

3. (2 Pkte.)
Am `stdout` wird mit
```
printf("%s0x%04X = %d", s, x, d);
```
folgende Ausgabe angezeigt:
```
64 K - 1 = 0xFFFF = 65535
```
Schreiben Sie das der Funktion `printf` vorangehende Codefragment zur Deklarierung und Initialisierung nötiger Variablen.

L:
```
char s[] = "64 K - 1 = ";
int x = 0xFFFF;
int d = x;
```

4. (2 Pkte.)
Die beiden Zeilen
```
sprintf(string, "%s 0x%04X = %d", s, x, d);
printf("%s 0x%04X = %d", s, x, d);
```
bewirken folgende Ausgabe an der Konsole:
```
64_K_-_1_= 0xFFFF = 65535
```
Stellen Sie genügend *neue* Variablen bereit, um diese mit Werten aus `string` zu initialisieren. Wie hat das entsprechende Code-Fragment auszusehen?

L:
```
char t[100];
int y;
int e;
sscanf(string, "%s 0x%04X = %d", t, &y, &e);
```

5. (4 Pkte.)
Mit den Werten `short a = 1000, b = 100; short c = 15; short x;` erfolgten von einem C-Programm Ausgaben wie im Rahmen unten gezeigt. Alle x-Resultate ganz rechts wurden durch das Programm berechnet.

```
Anzahl Bytes fuer int: sizeof(int) = 4
Anzahl Bytes fuer short: sizeof(short) = 2
a) x = a * b / c,                        x = 6666
b) x = (short)((int)a * (int)b) / c,     x = -2071
c) x = ((short)((int)a * (int)b)) / c,   x = -2071
d) x = ((short)((int)a * (int)b) / c),   x = -2071
e) x = (short)((int)a * (int)b / c),     x = 6666
f) x = (short)(a * b) / c,               x = -2071
g) x = (int)(a * b) / c,                 x = 6666
```

Diskutieren Sie die Zeilen `a)` und `b)`.

L:
Die Zeile `a)` liefert ein korrektes Resultat (bis auf Abrundung, da `int`). Da `short` bei Berechnungen (hier nach `int`) promoviert werden, reicht der Bereich.
In Zeile `b)` wird zuerst nach `int` gecastet (eigentlich unnötig) und darauf multipliziert. Mit dem `short`-Cast, der sich wie in Zeile `c)` verhält, kommt es zu einem Informationsverlust, der bei der anschliessenden Promotion nach `int` (wegen Division) zu einem negativen Zähler führt. Dies führt zum fehlerhaften Resultat.

6. (2 Pkte.)
Gegeben ist die Binärzahl `1001'1110`. Wandeln Sie die Zahl mit einem einzigen C-Bitoperatoren so, dass das MSBit und das LSBit (und nur diese) den invertierten Bitwert annehmen.
`short val = 0x9E;`

L:
`val ^= 0x81;`

7. (1 Pkt.)
Welcher Wert ergibt sich für `x` nach Ausführung der folgenden Zeile?
```
int x = 9, y = 2;
x *= y + 1;              x =  27
```

8. (2 Pkte.)
Was tut das folgende Codefragment? Geben Sie eine möglichst knappe, aber präzise Antwort.
```
short n = 0xF6, b;
for (b = 0; n; n >>= 1) {
  if (n & 1) {
    b++;
  }
}
```

L:
Es zählt die Anzahl der Einer in `n` in binärer Darstellung. Oder: Es zählt die Anzahl der Bits, die 1 sind.

9. (3 Pkte.)
Gegeben ist folgendes Code-Fragment:
```
struct A {
  char s[10];
  int len;
};
```

Schreiben Sie eine eigene Funktion `concat`, die zwei Parameter vom Typ `struct A` entgegen nimmt, deren C-Strings verkettet und die Länge anpasst und das Resultat als neue `struct A` zurückgibt.

Hinweis: Keine Tests auf Anomalien vornehmen.

L:
Mit C-String-Funktionen:
Annahme: Struktur für 0-delimited C-Strings mit der Stringlänge len ('\0' nicht mit gezählt).

```
struct A concat(struct A head, struct A tail) {
  struct A res;
  strcpy(res.s, head.s);
  strcat(res.s, tail.s);
  res.len = strlen(res.s);
  return res;
}
```

Die etwas schnellere Art:

```
struct A concat(struct A head, struct A tail) {
  strcat(head.s, tail.s);
  head.len = strlen(head.s);
  return head;
}
```

Variante:

```
struct A concat(struct A head, struct A tail) {
  struct A res;
  sprintf(res.s, "%s%s", head.s, tail.s);
  res.len = strlen(res.s);
  return res;
}
```

Ohne C-String-Funktionen:
Annahme: kein 0-delimited C-String in Struktur.

```
struct A concat(struct A head, struct A tail) {
  struct A res;
  int i, j;

  for (i = 0; i < head.len; ++i) {
    res.s[i] = head.s[i];
  }
  for (j = 0; j < tail.len; ++i, ++j) {
    res.s[i] = tail.s[j];
  }
  res.len = head.len + tail.len;
  return res;
}
```

Schnellere Variante:

```
struct A concat(struct A head, struct A tail) {
  int i = 0, len = head.len;

  for ( ; i < tail.len; ++i) {
    head.s[i + len] = tail.s[i];
  }
  head.len += tail.len;
  return head;
}
```

10. (4 Pkte.)
Gegeben sind folgende Zeilen (Ausschnitt):

```
#define LEN 11
char s[LEN] = "char s[LEN]";
struct A { char s[LEN]; int i; };
struct A a = { "char s[LEN]" , LEN };
```

Schreiben Sie zwei Funktionen. Die eine werde verwendet, um alle Zeichen eines Array in Grossbuchstaben zu wandeln, Parameterübergabe: Array, Array-Grösse. Die andere werde verwendet, um alle Zeichen des Array in einer **`struct A`** - Variablen in Grossbuchstaben zu wandeln. Übergabe: Struktur-Variable.
Notieren Sie auch den Code, der zeigt, wie mit Hilfe der Funktionen das entsprechende Resultat errreicht wird.
Hinweis: **`stdio`**-Funktion die **`char`** in Grossbuchstaben wandelt: **`char toupper(char)`**

```
void arrToUpper(char s[], int len) {
  int i;
  for (i = 0; i < len; i++) {
     s[i] = toupper(s[i]);
  }
}


struct A structToUpper(struct A a) {
  arrToUpper(a.s, a.len);
  return a;
}


/* ... */

char s[] = "abcde";              /* war nicht verlangt */
struct A a = {"tuvwxyz", 7};     /* war nicht verlangt */


arrToUpper(s, 5);
a = structToUpper(a);
```