Übungsblatt 3

Willkommen zum Praktikum zu Programmieren 3.

Aufgabe 1. Zeichnen Sie einen Speicherbereich von `0x100` inklusive bis `0x120` exklusive byteweise adressierbar je Wort eine Zeile. Nehmen Sie an, dass konsekutiv Speicher verwendet wird. Jeder Wert soll innerhalb eines Wortes liegen, der Anfang wird also gegebenenfalls verschoben (wortweise aligned). Zeichnen Sie bei folgendem Quelltext ein, wo jede Variable liegt und wie sich der Speicher verändert hat jedes mal wenn `showmem()` aufgerufen wird.

```
1  int i=42;
2  char c='a';
3  char ca[2] = {'b', 'c'};
4  short sa[3] = {0, 8, 15};
5  char *cp = &c;
6  int *ip = &i;
7
8  showmem();
9  *ip = 17;
10 ip = (int *) sa;
11 *ip= 0x12121212;
12 showmem();
13 *cp++ += 1;
14 *cp++ = 'x';
15 *cp++ = 'y';
16 *cp++ = 'z';
17 showmem();
18 printf("i:%d c:%c ca:%c,%c sa:%d,%d,%d\n",
19      i, c, ca[0], ca[1], sa[0],sa[1],sa[2]);
```

Aufgabe 2. Zeichnen Sie die Belegung des Speichers für die folgenden Definitionen (jede separat).

```
1  char *a[2];
2  char (*b)[2];
3  char c[2][3];
4  char *d[2] = {"hi", "world"};
5  char e[2][6] = {"hi", "world"};
```

Wieviel Byte Speicher belegt jede Definition? Tragen Sie bei den Bereichen, die initialisiert wurden die Werte in die Speicherblöcke.

Aufgabe 3. Schreiben Sie ein Programm `komprimier`, das eine Zeichenkette ohne Ziffern als ersten Kommandozeilenparameter erwartet und komprimiert, beziehungsweise

wenn das Programm `dekomprimier` heißt dekomprimiert und auf der Standardausgabe wieder ausgibt. Das Kompressionsverfahren ersetzt jedes Zeichen das mindestens zweimal hintereinander vorkommt durch die Anzahl der Vorkommen als Dezimalzahl gefolgt von dem Zeichen.

Schreiben Sie dazu Funktionen `komprimier` und `dekomprimier`, die einen Zeiger auf eine konstante Zeichenkette erhalten und einen Zeiger auf eine neu angelegte Zeichenkette zurückgeben. Allokieren Sie so wenig Speicher so wenig häufig wie möglich. Stellen Sie sicher, dass das Programm ohne Speicherfehler durchläuft.

Aufgabe 4. Schreiben Sie ein Programm `split`, das auf der Kommandozeile mindestens zwei Strings erwartet. Jeder außer dem ersten String soll gesplittet werden an jedem Vorkommen des ersten Strings. Schreiben Sie dazu eine Funktion

```
split(char *str, char *delim)
```

die ein Struct zurückgibt in dem auf die Anzahl der gesplitteten Strings und die gesplitteten Strings selbst zurückgegriffen werden kann. Platz für die gesplitteten Strings ist immer neu. Schreiben Sie auch eine Funktion `clear`, die ein Struct übergeben bekommt und Platz für die erzeugten Strings entfernt. Versuchen Sie so wenig wie möglich so wenig häufig wie möglich zu allokieren und frei zu geben. Verwenden Sie ein Feld mit dem sie alle Structs für den Programmablauf verwalten. Splitten Sie erst alle Strings, dann geben Sie die gesplitteten Strings aus, dann geben Sie den allokierten Platz frei. Stellen Sie sicher, dass das Programm fehlerfrei und ohne Speicherfehler läuft.

Aufgabe 5. Was ist das Ergebnis des folgenden Programms.

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4    char a = 7;
5    char b = 3;
6    printf("%d, ", a | b);
7    printf("%d, ", a ^ b);
8    printf("%d, ", ~(~a & ~b));
9    printf("%d, ", a << 1);
10   printf("%d\n", a >> 2);
11   return 0;
12 }
```

Lösen Sie die Aufgabe erst auf Papier und überprüfen Sie dann Ihre Antwort am Rechner.

http://www.mi.hs-rm.de/~barth/hsrm/prog3