Universität Augsburg
Institut für Informatik

Deckblatt: Übung zur Vorlesung Informatik 1 WS 2017/18
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Marius Brendle, Johannes Metzger, Lev Sorokin

---

Hinweis: Es sind alle Felder auszufüllen! Abgabe der Übungsblätter immer **mittwochs** (Ausnahme wenn Feiertag: donnerstags) bis **spätestens 12:00 Uhr** in die entsprechend gekennzeichneten Briefkästen der Veranstaltung im Erdgeschoss des Instituts für Informatik (Gebäude N). Zuwiderhandlung wird mit Strafe geahndet! (Punktabzug)

Übungsblatt	

(hier die Nummer des bearbeiteten **Übungsblatts eintragen**)

	Übung 01 (1055 N) Montag 08:15 - 09:45 Uhr (Lennart Eing)
	Übung 02 (1056 N) Montag 08:15 - 09:45 Uhr (Alexander Fuchs)
	Übung 03 (1057 N) Montag 08:15 - 09:45 Uhr (Michelle Lienhart)
	Übung 04 (1055 N) Montag 12:15 - 13:45 Uhr (Henning Cui)
	Übung 05 (1056 N) Montag 12:15 - 13:45 Uhr (Christian Schavitz)
	Übung 06 (1055 N) Montag 14:00 - 15:30 Uhr (Maximilian Demmler)
	Übung 08 (1056 N) Montag 17:30 - 19:00 Uhr (Moritz Feldmann)
	Übung 09 (1057 N) Montag 17:30 - 19:00 Uhr (Dat Le Thanh)
	Übung 10 (1057 N) Dienstag 12:15 - 13:45 Uhr (Alexander Szöke)
	Übung 11 (1057 N) Dienstag 14:00 - 15:30 Uhr (Denise Böhm)
	Übung 12 (1056 N) Dienstag 17:30 - 19:00 Uhr (Marvin Drexelius)
	Übung 13 (1057 N) Dienstag 17:30 - 19:00 Uhr (Tom Wolfskämpf)
	Übung 14 (1055 N) Mittwoch 08:15 - 09:45 Uhr (Jonas Junge)
	Übung 15 (1055 N) Mittwoch 10:00 - 11:30 Uhr (Elisabeth Korndörfer)
X	Übung 16 (1054 N) Donnerstag 14:00 - 15:30 Uhr (Florian Magg)
	Übung 17 (1057 N) Donnerstag 14:00 - 15:30 Uhr (Lukas Lodes)
	Übung 18 (1054 N)* Donnerstag 17:30 - 19:00 Uhr (Patrick Eckert)
	Übung 19 (1058 N) Freitag 08:15 - 09:45 Uhr (Lena Tikovsky)
	Übung 20 (1054 N) Freitag 10:00 - 11:30 Uhr (Felix Fischer)
	Übung 21 (1055 N)* Freitag 14:00 - 15:30 Uhr (Isabell Rücker)
	Übung 23 (1057 N) Freitag 15:45 - 17:15 Uhr (André Schweiger)

(hier die eingeteilte **Übungsgruppe ankreuzen**) *(1056 N bis 03.11.17)

Teamnummer	5

(hier die Nummer des eingeteilten **Teams eintragen**)

Benjamin Ritter
Marina Huber
Anton Lydike

(hier die **Vor- und Nachnamen** aller Teammitglieder **eintragen**)

Aufgabe		
Aufgabe		
Aufgabe		
Aufgabe		
Gesamt		(vom Tutor auszufüllen)

Assignment 9

33)

i)

1. `int * p;` (oder `int * p = NULL;`)
2. `double * q = &x;`
3. `char * r = w+sizeof(char);`
4. `'i'`
5. `*w = 'c';`

ii)

1. Gib die ersten 2 Buchstaben des Wortes "Merry" aus.
2. Gib "er" aus.
3. Gib die ersten 2 Buchstaben des Wortes "Merry" aus.
4. Gib den Buchstaben 'N' ('M' + 1), gefolgt von 'O' ('N' + 1) aus.
5. Gib das Wort "Merry" ab dem zweiten Buchstaben aus.

iii)

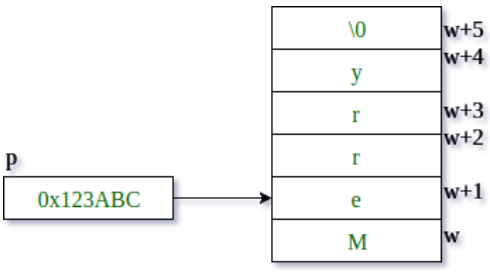
1. Zeile 6: `double` sollte `int` sein.
2. Zeile 7: Arrays (oder Strings) können nicht einfach so kopiert werden. (Es müsste `strcpy` verwendet werden?)
3. Zeile 6: Pointer kann keinen integer Wert annehmen.
4. Zeile 7: `p` ist kein Array, deshalb zugriff auf nicht reservierten / vorhergesehenen Speicher.

iv)

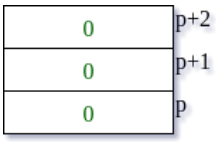
1. Zähle wie oft `x` in `w` vorkommt.
2. Gebe die ersten `n` werte aus dem int array `x` aus.

v)

1.



2.



34)

a)

```c
int * read_pos()
{
        int *px;

        px = malloc(sizeof(int));
        if (px == NULL) {
                return NULL;
        }

        if (scanf("%i", px) != 1) {
                free(px);
                return NULL;
        }

        return px;
}
```

b)

```c
int main(int argc, char *argv[])
{
        int i;
        char *w = malloc((strlen(argv[0]) + 1) * sizeof(char));

        if (w == NULL) {
                /* nothing to free */
                return 1;
        }

        for (i = 1; i < argc; ++i) {
                char *neu = realloc(w, (strlen(w) + strlen(argv[i]) + 1) * sizeof(char));

                if (neu == NULL) {
                        /* w is still allocated */
                        free(w);
                        return 1;
                }

                w = neu;
        }
        printf("%s\n", w);
        free(w);
        /* free w, it's no longer used, freeing w, frees neu automatically */
        return 0;
}
```

c)

Anweisung	f[0]	f[1]	f[2]	r	s
`char f[3], *r, *s;`	`-`	`-`	`-`	`?`	`?`
`r = f;`	`-`	`-`	`-`	`4000`	`?`
`s = r;`	`-`	`-`	`-`	`4000`	`4000`
`*r = 'A';`	`A`	`-`	`-`	`4000`	`4000`
`*(r+1) = *f + 1;`	`A`	`B`	`-`	`4000`	`4000`
`r++;`	`A`	`B`	`-`	`4001`	`4000`
`(*s)++;`	`B`	`B`	`-`	`4001`	`4000`
`f[2] = *r;`	`B`	`B`	`B`	`4001`	`4000`
`s = &f[2];`	`B`	`B`	`B`	`4001`	`4002`
`(*f)++;`	`C`	`B`	`B`	`4001`	`4002`
`(*s)--;`	`C`	`B`	`A`	`4001`	`4002`
`*(++r) = 'B';`	`C`	`B`	`B`	`4002`	`4002`

d)

Anweisung	a	b	r	s
`double a = 1.3, *r;`	`1.3`	`-`	`?`	`-`
`int b = 2, *s;`	`1.3`	`2`	`?`	`?`
`r = &a - 1;`	`1.3`	`2`	`1992`	`?`
`a--;`	`0.3`	`2`	`1992`	`?`
`s = &b + 1;`	`0.3`	`2`	`1992`	`3004`
`b++`	`0.3`	`3`	`1992`	`3004`
`r--;`	`0.3`	`3`	`1984`	`3004`
`s--;`	`0.3`	`3`	`1992`	`3000`
`*(r + 2) = 2.5;`	`2.5`	`3`	`1992`	`3000`
`b = b / a;`	`2.5`	`1`	`1992`	`3000`
`*s = 2 * b;`	`2.5`	`2`	`1992`	`3000`

35)

a)

```c
#include <stdio.h>
#include <ctype.h>

int read_alpha(char *in)
{
        char buff;

        scanf("%c", &buff);

        if (buff >= 'a' && buff <= 'z') {
                *in = buff;
                return 1;
        }

        return 0;
}

int main()
{
        char c;
        int success;

        success = read_alpha(&c);

        if (success == 0) {
                printf("Invalid character.\n");
                return 1;
        }

        printf("Read: %c\n", c);
        return 0;
}
```

b)

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>


int main (int argc, char *argv[])
{
        char * str;
        int n, * random, l = 0;
        time_t t;

        if (argc != 2) {
                printf("WHY WOULD YOU BE THIS CRUEL?!\n");
                return 1;
        }

        n = 0;
        str = argv[1];

        if (*str == '-') {
                printf("It's a retard!\n");
                return 3;
        }

        while (*str != '\0') {
                if (*str >= '0' && *str <= '9') {
                        if (n != 0) {
                                n *= 10;
                        }

                        n += *str - '0';
                } else {
                        printf("Hey! Do you know what a fucking number is?!\n");
                        return 2;
                }

                str++;
        }

        printf("requested %d random numbers.\n", n);

        if (n == 0) return 0;

        random = calloc(n, sizeof(int));

        if (random == NULL) {
                printf("Well, I guess you should by memory. Yes, I know it's stupidly expensive atm.");
                return 1;
        }

        srand((unsigned) time(&t));

        /* initialize array backwards, then print forward */

        l = n;
        for (; n > 0; n--) {
                random[n - 1] = rand();
        }

        for (; n < l; n++) {
                printf("%d\n", random[n]);
        }

        free(random);
        return 0;

}
```

36)

a)

```
char * crypt(char *in, char c, char *out)
{
        while (*in != '\0')
                *out++ = *in++ + c;

        *out = '\0';

        return out;
}
```

b)

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

int main (int argcount, char *args[])
{
        int c = 0;
        char *newhaystack = args[1];

        if (argcount != 3) {
                printf("Dude. Two arguments. Is it that hard?!\n");
                return 1;
        }

        while ((newhaystack = strstr(newhaystack, args[2])) != NULL) {
                c++;
                newhaystack++;
        }

        printf("%d\n", c);

        return 0;
}
```

c)

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

int main (int argcount, char *args[])
{
        int i;
        char * str = calloc(1, sizeof(char));

        for (i = 1; i < argcount; i++) {
                str = realloc(str, strlen(str) + strlen(args[i]) + 2);

                if (str == NULL) {
                        printf("I came here to concat strings and reserve memory. And I am all out of
memory!");
                        return 1;
                }

                if (i != 1) {
                        strcat(str, " ");
                }

                str = strcat(str, args[i]);
        }

        printf("%s\n", str);

        free(str);

        return 0;
}
```