

Assessment I

Vorname: _____

Punkte: ____ / 90, Note: ____

Name: _____

Frei lassen für Korrektur.

Klasse: 4ibb2

Hilfsmittel:

- Ein A4-Blatt handgeschriebene Zusammenfassung.
- Lösen Sie die Aufgaben jeweils direkt auf den Prüfungsblättern.
- Zusatzblätter, falls nötig, mit Ihrem Namen und Fragen-Nr. auf jedem Blatt.

Nicht erlaubt:

- Unterlagen (Slides, Bücher, ...).
- Computer (Laptop, Smartphone, ...).
- Kommunikation mit anderen Personen.

Bewertung:

- Multiple Response: ☐ *Ja* oder ☐ *Nein* ankreuzen, +1/-1 Punkt pro richtige/falsche Antwort, beide nicht ankreuzen ergibt +0 Punkte; Total pro Frage gibt es nie weniger als 0 Punkte.
- Offene Fragen: Bewertet wird Korrektheit, Vollständigkeit und Kürze der Antwort.
- Programme: Bewertet wird die Idee/Skizze und Umsetzung des Programms.

Fragen zur Prüfung:

- Während der Prüfung werden vom Dozent keine Fragen zur Prüfung beantwortet.
- Ist etwas unklar, machen Sie eine Annahme und notieren Sie diese auf der Prüfung.

Erste Schritte in C

1) Welche dieser Ausdrücke kompilieren fehlerfrei und ohne Warnung in C? Punkte: ___ / 6

Zutreffendes ankreuzen; Annahmen: Keine #includes; Compiliert mit gcc -std=c99

- ☐ Ja | ☐ Nein `int b = !0;`
- ☐ Ja | ☐ Nein `float f = 0.2 * 210;`
- ☐ Ja | ☐ Nein `double d = sizeof(int);`
- ☐ Ja | ☐ Nein `string s = "Hello, World!";`
- ☐ Ja | ☐ Nein `char msg[3] = { 'h', 'o', 'i' };`
- ☐ Ja | ☐ Nein `char data[] = { 0x68, 0x6f, 0x69, 0x00 };`

2) Nennen Sie einen Vor- und einen möglichen Nachteil von *unsigned* Typen? Punkte: ___ / 4

Vorteil:

Nachteil:

3) Welche der folgenden Aussagen sind immer korrekt?

Punkte: ___ / 4

Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein Ein *int* Wert ist im Speicher höchstens 4 Byte gross.
- ☐ Ja | ☐ Nein Der *sizeof* Operator liefert für den Typ *char* den Wert 1.
- ☐ Ja | ☐ Nein Ein *long* Wert belegt im Speicher mehr Bytes als ein *int* Wert.
- ☐ Ja | ☐ Nein Der Typ *double* nutzt doppelt so viele Bytes wie der Typ *float*.

4) Welche Abfolge von Statements führt zu folgender Situation im Speicher? Punkte: ___ / 4



Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein `int i = 3; int *p = &i; int **q = &p;`
- ☐ Ja | ☐ Nein `int i = 3; int *p = &i; int **q = p;`
- ☐ Ja | ☐ Nein `int i = 2; int *p = &i; *p += 1; int **q = &p;`
- ☐ Ja | ☐ Nein `int i = 2; int *p = &i; p += 1; int **q = p;`

Funktionen in C

5) Implementieren Sie eine Funktion *append()*, die einen String *src* hinten an den String *dest* anhängt und *dest* zurück gibt, für beliebige Strings*, ohne Hilfsfunktionen. Punkte: ___ / 12

**Annahme: Das übergebene Argument dest ist ein char Array mit genügend Speicherplatz.*

Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:

// Idee:

6) Gegeben den folgenden Code, beschreiben Sie (in der Tabelle unten) vier wesentliche

Unterschiede der Funktionen `create_point1()` und `create_point2()`.

Punkte: ___ / 8

```
#include ... // ignore

struct point {
    int x;
    int y;
};

struct point create_point1(int x, int y) {
    struct point result = { x, y };
    return result;
}

struct point *create_point2(int x, int y) {
    struct point *result = malloc(sizeof(struct point));
    result->x = x;
    result->y = y;
    return result;
}

int main() {
    struct point p = create_point1(0, 0);
    struct point *q = create_point2(0, 0);
}
```

Unterschiede hier eintragen, jeweils beide Seiten des Unterschieds ausformulieren:

create_point1()	create_point2()

7) Schreiben Sie ein Programm *count*, das die Häufigkeit jedes Kleinbuchstabens von a - z in den übergebenen Argumenten zählt, und wie in diesem Beispiel ausgibt. Punkte: ____ / 12

```
$ ./count not bad meaning bad but bad meaning good
a: 5
b: 4
...
z: 0
```

Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
int printf(const char *format, ...); // format string %s, char %c, int %d
size_t strlen(const char *s); // calculate the length of a string
```

Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:

```
// Idee:
```

8) Gegeben den folgenden Code, implementieren Sie die Funktion *add()*, welche ein neues Item vorne in die unsortierte Liste *list* hängt und diese zurück gibt. Sowie die Funktion *log()*, welche für alle Items in der Liste *list* den Key und Value mit *printf()* ausgibt. Punkte: ___ / 12

```
#include ... // ignore

struct item {
    int key;
    char value[32];
    struct item* next;
};

struct item *list = NULL;

struct item *add(struct item *list, int key, char *value);
void log(struct item *list);

int main() {
    list = add(list, 3000, "Bern");
    list = add(list, 4000, "Basel");
    list = add(list, 8000, "Zurich");
    log(list);
}
```

Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
void *malloc(size_t size); // Allocates size bytes, returns pointer to
the allocated memory.
```

```
int printf(const char *format, ...); // format string %s, char %c, int %d
```

```
char *strcpy(char *dest, char *src); // Copies src to dest, incl. '\0'.
```

Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:

```
// Idee:
```

```
// Fortsetzung auf Folgeseite
```

(8) Fortsetzung:

File In-/Output

9) Schreiben Sie ein Programm *merge*, welches ein leeres File *file1* erstellt, und alle weiteren per Command Line übergebenen Files *file2, file3, ... fileN*, dort reinkopiert, z.B.: P.kte: _ / 12

```
$ echo "Hello, " > hello
$ echo "World!" > world
$ ./merge result hello world
$ cat result
Hello, World!
$ ./merge result hello hello hello world
$ cat result
Hello, Hello, Hello, World!
```

Verwenden Sie dazu die folgenden System Calls, Fehlerbehandlung können Sie weglassen:

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode); // Opens the file
specified by pathname. Or creates it if O_CREAT is used. Returns the file
descriptor. Flags include O_APPEND, O_CREAT, O_TRUNC, O_RDONLY, O_WRONLY.
Modes, which are used together with O_CREAT include S_IRUSR and S_IWUSR.
```

```
ssize_t read(int fd, void *buf, size_t n); // Attempts to read up to n
bytes from file descriptor fd into buf. Returns number of bytes read ≤ n.
```

```
int ftruncate(int fd, off_t length); // Truncate a file to length bytes.
```

```
ssize_t write(int fd, const void *buf, size_t n); // Writes up to n bytes
from buf to the file referred to by fd. Returns nr. of bytes written ≤ n.
```

(9) Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen & Fragen-Nr.:

// Idee :

Prozesse und Signale

10) Gegeben den folgenden Code, erklären Sie wieso printf() den Wert 23 ausgibt. P.kte: __ / 8

```
01: #include <stdio.h>
02:
03: void f(int i) {}
04:
05: void g() {
06:     int j;
07:     printf("%d\n", j);
08: }
09:
10: int main() {
11:     f(23);
12:     g();
13: }
```

Erklärung hier eintragen, Schritt für Schritt ausformulieren:

```
// Erklärung
```

11) Schreiben Sie ein Programm *raise* welches sich selbst das Signal SIGUSR1 genau dreimal sendet, dazwischen jeweils dessen Empfang abwartet, und dann terminiert. Punkte: ___ / 8

```
int pause(void); // Pause causes the calling process to sleep until a
signal terminates the process or causes invocation of a handler function.
```

```
int raise(int sig); // Send the signal sig to the calling process.
```

```
typedef void (*sighandler_t)(int);
sighandler_t signal(int sig, sighandler_t handler); // set SIG_IGN,
SIG_DFL, or a programmer-defined function to handle the signal sig.
```

Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:

```
// Idee:
```

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____