

# Assessment I 4ibb1

Vorname: \_\_\_\_\_

Punkte: \_\_\_\_ / 90, Note: \_\_\_\_

Name: \_\_\_\_\_

*Frei lassen für Korrektur.*

Klasse: ☒ Klasse 4ibb1 ☐ Klasse 4ibb2

Hilfsmittel:

- Die vom Dozent ausgeteilte C Referenzkarte.
- Lösen Sie die Aufgaben 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.
- Multiple Choice: Eine ☒ *Antwort* pro Frage ankreuzen, 4 Punkte pro richtige Antwort.
- Offene Fragen: Bewertet wird Korrektheit, Vollständigkeit und Kürze der Antwort.
- Programme: Bewertet wird die Skizze/Idee 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) Für welche dieser Typen ist *sizeof(type)* nie kleiner als *sizeof(int)*?

Punkte: \_\_\_\_ / 4

*Zutreffendes ankreuzen:*

- ☐ Ja | ☐ Nein      float
- ☐ Ja | ☐ Nein      char
- ☐ Ja | ☐ Nein      long
- ☐ Ja | ☐ Nein      unsigned int

2) Welche dieser Ausdrücke kompilieren fehlerfrei in C?

Punkte: \_\_\_\_ / 4

*Zutreffendes ankreuzen; C = C99:*

- ☐ Ja | ☐ Nein      int i = 3.141;
- ☐ Ja | ☐ Nein      byte b = 0x23;
- ☐ Ja | ☐ Nein      char s[] = {0x23, 0x00};
- ☐ Ja | ☐ Nein      int j = !3;

3) Was sind zwei Gründe System-Datentypen wie *size\_t* zu nutzen, statt *int*? Punkte: \_\_\_\_ / 4

## Funktionen in C

4) Schreiben Sie ein Programm, das seine Argumente in Uppercase ausgibt: Punkte: \_\_\_\_ / 12

```
$ ./my_program just a test  
JUST A TEST
```

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

```
int printf(const char *format, ...); // format string %s, int %d  
int toupper(int ch); // convert a character to uppercase, e.g. 'a' => 'A'
```

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

5) Gegeben den folgenden Code, welchen Wert hat  $k$  nach Aufruf von  $f()$ ? Punkte: \_\_\_\_ / 4

```
int f(int *m, int n) {  
    return *(m + 1) * n;  
}  
  
int main() {  
    int i[] = {3, 5};  
    int j = 2;  
    int k = f(i, j);  
}
```

Resultat und Begründung hier eintragen:

6) Gegeben den folgenden Code, welche Aufrufe von  $map()$  sind erlaubt? Punkte: \_\_\_\_ / 4

```
int sum(int *a, int n) { int s = 0; while (n > 0) { s += a[--n]; } return s; }  
int avg(int a[], int n) { return sum(a, n) / n; }  
void map(int (*op)(int[], int), int a[], int n) { printf("%d", op(a, n)); }
```

Zutreffendes ankreuzen:

- ☐ Ja | ☐ Nein      `map((int[]) {1, 2}, 2, sum);`
- ☐ Ja | ☐ Nein      `map(avg, (int []) {1, 2}, 2);`
- ☐ Ja | ☐ Nein      `map((int []) {1, 2, 3, 4}, 4, avg);`
- ☐ Ja | ☐ Nein      `map(sum, 2, (int[]) {1, 2});`

## File In-/Output

7) Schreiben Sie ein Programm, das *n* per Command-line angegebene, existierenden ASCII

Dateien, z.B. *a*, *b* und *c* (ohne *.txt*) zu einer neuen Datei *abc* zusammenhängt. Punkte: \_\_\_ / 16

*Nutzen Sie dazu diese System-Calls, #includes und 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_RDONLY, O_WRONLY, O_RDWR.  
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.
```

```
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.
```

```
int close(int fd); // Closes the file descriptor fd.
```

```
void sprintf(char *str, const char *format, ...); // Prints to a string,  
with the same format as printf(), e.g. %d for int, %s for string.
```

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

## Prozesse und Signale

8) Was passiert mit einem Prozess, der endet bevor der Parent *wait()* aufruft? Punkte: \_\_\_ / 4

Eine Antwort (von 4) ankreuzen:

- |   |  |
|---|--|
| <input type="checkbox"/> Er wird zu einem Zombie-Prozess.     | <input type="checkbox"/> Er wird vom <i>init</i> -Prozess adoptiert. |
| <input type="checkbox"/> Der Child-Prozess bekommt die PID 0. | <input type="checkbox"/> Der Prozess wartet auf ein Signal.          |

9) Schreiben Sie ein Programm, welches das SIGINT Signal (CTRL-C) genau einmal ignoriert, und dann das Default-Verhalten wieder herstellt, d.h. es terminiert bei CTL-C. Punkte: \_\_\_ / 8

```
typedef void (*sighandler_t)(int);  
sighandler_t signal(int signum, sighandler_t handler); // set SIG_IGN,  
SIG_DFL, or a programmer-defined function to handle the signal signum.  
  
int pause(void); // Pause causes the calling process to sleep until a  
signal terminates the process or causes invocation of a handler function.
```

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

10) Gegeben die Queue Datenstruktur (unten), implementieren Sie *enqueue()* um einen Node zu allozieren und hinten in die Queue *queue* einzufügen, und die Funktion *dequeue()* um den vordersten Wert abzuholen (FIFO) und Speicher des Nodes freizugeben. Punkte: \_\_ / 9

```
typedef struct node {  
    struct node *next;  
    int item;  
} Node;  
  
static Node *queue_head = NULL;  
static Node *queue_tail = NULL;  
  
void enqueue(int item); // TODO  
int dequeue(); // TODO
```

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

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

```
void free(void *ptr); // Frees the memory space pointed to by ptr, which  
must have been returned by a previous call to malloc().
```

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

## Prozess-Lebenszyklus

11) Was ist der Output dieses Programms, und wieso?

Punkte: \_\_\_\_ / 9

```
#include ...

int main(void) {
    for (int i = 0; i < 2; i++) {
        pid_t pid = fork();
        if (pid == 0) {
            printf("child pid = %d\n", getpid());
        } else {
            printf("parent pid = %d\n", getpid());
            wait(NULL);
        }
    }
    return 0;
}
```

*Output und Begründung hier eintragen; Annahme: #includes sind da, Parent läuft zuerst:*



## Threads und Synchronisation

12) Was sind drei wichtige Vorteile von Threads gegenüber Prozessen?

Punkte: \_\_\_\_ / 6

13) Was sind drei wichtige Eigenschaften einer Critical Section?

Punkte: \_\_\_\_ / 6

*Zusatzblatt zu Aufgabe Nr. \_\_\_\_ von (Name) \_\_\_\_\_*