

Beispielprüfung/*Mock-Exam*

Datenstrukturen und Algorithmen (CSE/CBB)

Felix Friedrich, Departement Informatik

ETH Zürich, 11.6.2019.

Name, Vorname:

Legi-Nummer:

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:

General guidelines:

1. Dauer der Prüfung: 75 Minuten. *Duration: 75 minutes.*
2. Erlaubte Unterlagen: Wörterbuch (für von Ihnen gesprochene Sprachen). 4 A4 Seiten handgeschrieben oder ≥ 11 pt Schriftgrösse. *Allowed aids: dictionary (for languages spoken by you). 4 A4 pages hand written or ≥ 11 pt font size.*
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) oder einen Bleistift. Bitte schreiben Sie leserlich. Nur korrekte Antworten werden bewertet. *Use a ballpoint pen (blue or black) or a pencil. Please write legibly. Only correct solutions that we can read will be evaluated.*
4. Lösungen sind direkt auf das Aufgabenblatt in den vorgesehenen Boxen zu schreiben (und dort, wenn mehr Platz benötigt wird). Ungültige Lösungen sind nicht durchzustreichen! Korrekturen sind nicht zulässig. Antworten geben bitte unmissverständlich an. *Solutions are to be written directly onto the exam sheet in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out. Corrections are not allowed. Please provide corrections to answers of questions without any ambiguity!*
5. Es gibt keine Negativpunkte. *There are no negative points for wrong answers.*
6. Störungen durch irgendjemanden sind nicht zulässig. Wenn Sie durch Störungen gestört werden, lassen Sie bitte sofort den Aufsichtsperson wissen. *Do not be disturbed by anyone or anything, let the supervisor of the exam know immediately.*
7. Wir sammeln die Examen am Ende der Prüfung. Bitte lassen Sie Ihre Examen von den Assistenten einsammeln und lassen Sie Ihre Examen zurück. Dasselbe gilt, wenn Sie vorzeitig fertig sind. Bitte lassen Sie Ihre Examen zurück. Vorzeitige Abgaben sind nur bis 15 Minuten vor Ende der Prüfung möglich. *We will collect the exams at the end. Important: You must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: Please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.*
8. Wenn Sie zur Toilette müssen, melden Sie sich bitte bei einer Aufsichtsperson durch Handzeichen. *If you need to go to the toilet, raise your hand and wait for a supervisor.*
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt. *We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.*

Beispielprüfung (Teil Parallele Programmierung) als Richtwert für den Stil der Prüfung im August. Nicht zählend. / Example exam (part parallel programming) as a rough guideline for the style of the exam in August. Does not count.

Question:	1	2	Total
Points:	16	18	34
Score:			

Anmerkung zu dieser Probeprüfung / Remark to this mock-exam

Am Anfang der August-Prüfung stehen noch Teile zum Kernthema Datenstrukturen und Algorithmen. In dieser Bespielrührung geht es nur um das Training des, zu etwa einem Viertel zählenden, Teiles über parallele Programmierung.

At the beginning of the August exam there will be parts on data structures and algorithms. This mock-exam is only about training the part on parallel programming that will contribute with a relative weight of about 1/4.

Aufgabe 1: Parallele Programmierung (16P)

- /4P (a) Niklas hat sein sequentielles Programm parallelisiert und den sequentiellen Anteil seines neuen Programmes mit 0.2 spezifiziert. Er misst mit 3 Prozessoren eine Laufzeitverbesserung (Speedup) von 2.5. Mehr Details hat er leider nicht preisgegeben. Welches Gesetz (Amdahl oder Gustavson oder beide?) kann diese Laufzeitverbesserung besser erklären?

Nilas has parallelized his sequential program and specifies the sequential part of his new program with 0.2. Using 3 processors, he measures a runtime speedup of 2.5. More details he does not provide. Which law (Amdahl or Gustavson or both) can explain this runtime improvement in a better way?

- /2P (b) Erklären Sie kurz, warum in manchen Fällen eher das Gesetz von Gustavson angebracht ist und in anderen Fällen das Gesetz von Amdahl.

Shortly explain, why in some cases Gustavson's law is more appropriate and in other cases the law of Amdahl.

Wir verwenden im folgenden Teil der Aufgabe C++ Code und nehmen an, dass Sie die jeweilige Syntax und Semantik verstehen. Es geht aber um das theoretische Verständnis der nebenläufigen Vorgänge. Treffen Sie insbesondere keine aussergewöhnliche Annahmen über die verwendete Architektur, das Speichermodell oder das Verhalten des Compilers.

Es werden Threads ausgeführt. Gehen Sie jeweils davon aus, dass die Funktion `print` jeweils einen einzigen Buchstaben ausgibt (Thread-sicher). Geben Sie jeweils alle möglichen Ausgaben der Programme unter dem Code an (allenfalls getrennt durch Semikolon). Bestimmen Sie, ob das Programm terminiert. Wenn das Programm nicht terminiert, geben Sie alle Ausgaben immer so weit an, wie sie auftreten können.

For the following part of the task we use C++ code and assume that you understand the syntax and semantics. But this task is more about the theoretical understanding of what can happen in a concurrent setup. Do particularly not make any exceptional assumptions about the architecture used, the memory model or the behavior of the compiler.

Threads are executed. We assume for each part that the function `print` outputs a single character (threads-safe). Provide for each part all (theoretically) possible outputs (separated by semicolons) of the following programs below the code. Specify if the program terminates. If the program does not terminate, provide all possible outputs as far as they can occur.

(c)

```
void print(char c); // output character c
```

```
void A(){
    print('A');
    print('B');
}
```

```
void B(){
    print('C');
    print('D');
}
```

```
int main(){
    std::thread t1(A);
    t1.join();
    std::thread t2(B);
    t2.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*

Das Programm terminiert/*the program terminates*

immer/*always* nie/*never* weder immer noch nie/*neither always nor never*

/2P

/2P (d)

```
void print(char c); // output character c

void A(){
    print('A');
    print('B');
}

void B(){
    print('C');
    std::thread t(A);
    print('D');
    t.join();
}

int main(){
    std::thread t(B);
    t.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*Das Programm terminiert/*the program terminates*

- immer/*always* nie/*never* weder immer noch nie/*neither always nor never*

(e)

/2P

```
void print(char c); // output character c
std::mutex m; // global variable useable by all threads

void A(){
    m.lock();
    print('A');
    print('B');
    m.unlock();
}

void B(){
    print('C');
    std::thread t(A);
    print('D');
    t.join();
}

int main(){
    std::thread t(B);
    t.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*Das Programm terminiert/*the program terminates*

immer/*always* nie/*never* weder immer noch nie/*neither always nor never*

/2P

 (f)

```
void print(char c); // output character c
std::mutex m; // global variable useable by all threads

void A(){
    m.lock();
    print('A');
    print('B');
    m.unlock();
}

void B(){
    m.lock();
    print('C');
    std::thread t(A);
    print('D');
    t.join();
    m.unlock();
}

int main(){
    std::thread t(B);
    t.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*

Das Programm terminiert/*the program terminates*

- immer/*always* nie/*never* weder immer noch nie/*neither always nor never*

(g)

/2P

```
void print(char c); // output character c
std::mutex m1; // global variable useable by all threads
std::mutex m2; // global variable useable by all threads

void A(){
    m2.lock();
    m1.lock();
    print('A');
    print('B');
    m1.unlock();
    m2.unlock();
}

void B(){
    m1.lock();
    print('C');
    std::thread t(A);
    m2.lock();
    print('D');
    m2.unlock();
    m1.unlock();
    t.join();
}

int main(){
    std::thread t(B);
    t.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*Das Programm terminiert/*the program terminates* immer/*always* nie/*never* weder immer noch nie/*neither always nor never*

Aufgabe 2: Programmieraufgabe: Readers-Writers Lock (18P)

/18P

Lösen Sie diese Aufgabe in der Online-Umgebung (CodeExpert via Moodle).

Solve this task in the online environment (CodeExpert via Moodle).