Vor und Nachname (Druckbuchstaben): _____

Legi Nummer: _____

252-0024-00L
Parallele Programmierung
ETH/CS: FS 2015
Basisprüfung
Montag, 10.08.2015
120 Minuten

Diese Prüfung enthält 28 Seiten (inklusive diesem Deckblatt) und 8 Aufgaben. Überprüfen Sie, dass keine Seiten fehlen. Füllen Sie alle oben verlangten Informationen aus. Schreiben Sie die Legi-Nummer oben auf jede einzelne Seite, für den Fall, dass Seiten verlorengehen oder abgetrennt werden.
Nehmen Sie sich am Anfang 5 Minuten Zeit, um alle Aufgaben durchzulesen. Während dieser Zeit ist es nicht erlaubt, Prüfungsfragen zu beantworten. Danach haben Sie 120 Minuten Zeit für die Lösung der Aufgaben.
Falls Sie sich durch irgendjemanden oder irgendetwas gestört fühlen, melden Sie dies sofort einer Aufsichtsperson. Wir sammeln die Prüfung zum Schluss ein. Wichtig: stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: bitte melden Sie sich lautlos und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.
Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen. Es darf zur gleichen Zeit immer nur eine Studentin oder ein Student zur Toilette.
Es gelten die folgenden Regeln:

- **Lösungen müssen lesbar sein**. Verwenden Sie für Ihre Lösungen den verfügbaren Platz. Lösungen mit unklarer Reihenfolge oder anderweitig unverständlicher Präsentation können zu Punktabzügen führen.
- **Lösungen ohne Lösungsweg erhalten nicht die volle Punktzahl**. Eine korrekte Antwort ohne Lösungsweg, Erklärungen oder algebraischen Umformungen erhält keine Punkte; inkorrekte Antworten mit teilweise richtigen Formeln, Berechnungen und Umformungen können Teilpunkte erhalten.
- Falls mehr Platz benötigt wird, schreiben Sie auf die leeren Seiten der Prüfung oder fordern Sie bei den Assistenten zusätzliche Blätter an. Versehen Sie die Aufgabe mit einem klaren Hinweis, falls das der Fall ist. Als Richtlinie: **Die Aufgaben sollten sich alle in dem vorgegebenen Platz beantworten lassen.**
- Die Aufgaben können auf **Englisch oder Deutsch** beantwortet werden. Benutzen Sie keinen roten Stift!

Problem	Points	Score
1	4	
2	4	
3	18	
4	4	
5	26	
6	12	
7	12	
8	12	
Total:	92	

Java Sequential Programming (8 points)

1. Gegeben ist der folgende Java-Code. *Consider the following Java code.*

```
interface Shape {
    public int area();
    public int perimeter();
}

class Square implements Shape {
    protected int length;
    public Square(int length) {
        this.length = length;
    }
    public int area() {
        return length * length;
    }
    public int perimeter () {
        return 4 * length;
    }
}

class Rectangle extends Square {
    int width;
    public Rectangle(int length, int width) {
        super(length);
        this.width = width;
    }
    public int area() {
        return length * width;
    }
}

public class Shapes {
    static void show(Shape x) {
        System.out.println(x.area() + ", " + x.perimeter());
    }

    public static void main(String[] args) {
      ...
    }
}
```

Für die Checkbox-Fragen gibt nur die korrekte Antwort 1 Punkt. Falsch markierte Checkboxen ergeben -1 Punkt. Die minimale Punktzahl einer Aufgabe ist 0.

For checkbox questions only the correct answer gives 1pt. Wrongly marked checkboxes result in -1 pts. The minimum number of points for one task is 0.

(a) Sie fügen den folgenden Code in der `main`-Methode ein und führen dann das Programm aus. Was ist die erwartete Ausgabe? *You are inserting the following code in your `main` method and then run the program. What is the expected result?* (1)

```
Shape a = new Rectangle(10, 20);
show(a);
```

- ☐ Das Programm kompiliert nicht. *The program does not compile.*
- ☐ Das Programm wirft eine Exception. *The program throws an exception.*
- ☐ Das Programm gibt "200, 60" aus. *The program prints "200, 60".*
- √ **Das Programm gibt "200, 40" aus.** *The program prints "200, 40".*

(b) Sie fügen den folgenden Code in der `main`-Methode ein und führen dann das Programm aus. Was ist die erwartete Ausgabe? *You are inserting the following code in your `main` method and then run the program. What is the expected result?* (1)

```
Shape b = new Square(30);
show(b);
```

- ☐ Das Programm kompiliert nicht. *The program does not compile.*
- ☐ Das Programm wirft eine Exception. *The program throws an exception.*
- √ **Das Programm gibt "900, 120" aus.** *The program prints "900, 120".*

(c) Sie fügen den folgenden Code in der `main`-Methode ein und führen dann das Programm aus. Was ist die erwartete Ausgabe? *You are inserting the following code in your `main` method and then run the program. What is the expected result?* (1)

```
Square c = new Rectangle(10, 20);
show(c);
```

- ☐ Das Programm kompiliert nicht. *The program does not compile.*
- ☐ Das Programm wirft eine Exception. *The program throws an exception.*
- ☐ Das Programm gibt "200, 60" aus. *The program prints "200, 60".*
- √ **Das Programm gibt "200, 40" aus.** *The program prints "200, 40".*
- ☐ Das Programm gibt "100, 40" aus. *The program prints "100, 40".*

(d) Sie fügen den folgenden Code in der `main`-Methode ein und führen dann das Programm aus. Was ist die erwartete Ausgabe? *You are inserting the following code in your `main` method and then run the program. What is the expected result?* (1)

```
Rectangle d = new Square(30);
show(d);
```

- √ **Das Programm kompiliert nicht.** *The program does not compile.*
- ☐ Das Programm wirft eine Exception *The program throws an exception.*
- ☐ Das Programm gibt "900, 120" aus. *The program prints "900, 120".*

2. Schreiben Sie eine Java-Funktion sum mit drei Argumenten: einem Integer-Array a, und zwei Integer-Werten first und last. Die Funktion soll die Summe aller Array-Werte mit gültigem Index von (inklusive) first bis (exklusive) last berechnen. Sie können first $\leq$ last annehmen. Die Summe ist 0 falls kein solches Element existiert. Die Funktion soll die berechnete Summe auf der Standard-Ausgabe ausgeben. Sie können die Methoden min und max verwenden, die jeweils zwei Argumente erhalten und das Minimum beziehungsweise Maximum der beiden Argumente zurück geben.

Write a Java function sum that takes three arguments: an array a of integers and two integers first and last. The function computes the sum of all array elements with valid indices between (and including) first and (excluding) last. You can assume first $\leq$ last. The sum is 0 if no such element exists. The function prints the sum to the standard output. You may assume the existence of the methods min and max that take as arguments two integers and return the minimum and respectively the maximum of their arguments. (4)

```
public static void sum(int[] a, int first, int last) {

    ..................................................

    ..................................................

    ..................................................

    for (int i=........; i<..............; .....) {

    ..................................................

    ..................................................

    ..................................................

    ..................................................

    }

    ..................................................

    ..................................................
}
```

Solution:

```
import static java.lang.Math.*;
...
  public static void sum(int[] a, int first, int last) {
    int s = 0;
    for (int i = Math.max(0, first); i < Math.min(last,
       a.length); i++) {
      s = s + a[i];
    }
    System.out.println("s = " + s);
  }
```

(A very generous) grading:

- +2pts for selecting the right indexes, +1pt for computing the sum (i.e. declaring the accumulator variable, filling some loop conditions, and doing the update), +1pt for printing the sum
- rounding is done to the lowest higher integer (e.g. 1.5 pts leads to a 2)
- syntactic mistakes are ignored, including using `this`, `public`, `private` and other non-relevant nonsense
- using wrong bounds (e.g. `i < last - 1`): -0.5pts
- using `a[i] >= first && a[i] < last`: +1pt instead of +2pts
- swapping `min` and `max`: +1pt instead of +2pts
- correct checks (i.e. `first >= 0`, `last <= a.length`), but wrongly setting the sum to 0: +1pt instead of +2pts

Speedup, Amdahl, Gustafson (18 points)

3. Beantworten Sie die folgenden Fragen zum Thema Speedup sowie Amdahlsches und Gustafsonsches Gesetz:

Answer the following questions about speedup and both Amdahl's and Gustafson's laws:

(a) Schreiben Sie die Formel für den Speedup nach dem **Gustafsonschen** Gesetz auf und erläutern Sie kurz die Parameter. (2)

*Write down the formula for speedup according to **Gustafson**'s law and briefly describe its parameters.*

Solution:

$$S_p = p - b(p-1)$$

S Speedup

p Number of processors

b Non-parallalizable fraction of any parallel process

Grading: ½ points for correct formula, ½ for each correct parameter description.

(b) Die Auswertung eines Programms hat ergeben, dass ein Programm einen Speedup von 10 (skaliert) hat, wenn es auf 16 Prozessoren läuft. (2)

An evaluation of a program has shown a (scaled) speedup of 10 when running on 16 cores

Was ist der serielle Anteil nach dem **Gustafsonschen** Gesetz?

*What is the serial fraction according to **Gustafson**'s law?*

Solution:

$$b = \frac{p - S_p}{p - 1}$$

$$b = \frac{2}{5}$$

Grading: 1p for formula only, 2p for result. No rounding required, and variables can already be substituted in formula.

(c) Was ist der serielle Anteil von (b) nach dem **Amdahlschen** Gesetz? (2)

*What is the serial fraction of (b) according to **Amdahl**'s law?*

Solution:

$$f = \frac{\frac{1}{S_p} - \frac{1}{p}}{1 - \frac{1}{p}}$$

$$f = \frac{1}{25} \qquad \text{also:} \quad \frac{6}{150}$$

Grading: 1p for formula only, 2p for result. No rounding required, and variables can already be substituted in formula.

(d) Ein Programm hat eine parallele Effizienz von $e = 50\%$ und einen seriellen Anteil von 10%. Berechnen Sie die Anzahl Prozessoren p, auf denen das Programm läuft. Benutzen Sie dazu den Speedup S_p nach dem **Amdahlschen** Gesetz sowie die folgende Definition der Effizienz. (5)

*A program is determined to have a parallel efficiency of $e = 50\%$. We know it has a serial fraction of 10%. Calculate the number of CPUs p it is using. Use the speedup S_p according to **Amdahl**'s law and the following definition of efficiency to solve the task.*

$$e = \frac{S_p}{p}$$

Solution:
The above formula defines the parallel efficiency. Combining it with Amdahl, we get:

$$\begin{aligned} e &= \frac{S_p}{p} \\ &= \frac{1}{p} \cdot \frac{1}{f + \frac{1}{p}(1-f)} \end{aligned}$$

Simplify to

$$e = \frac{1}{1 + f(p-1)}$$

Solve for p

$$p = 1 + \frac{\frac{1}{e} - 1}{f}$$

Substituting $e = \frac{1}{2}$ and $f = \frac{1}{10}$ leads to

$$\begin{aligned} p &= 1 + \frac{\frac{1}{\frac{1}{2}} - 1}{\frac{1}{10}} \\ p &= 11 \end{aligned}$$

Grading: Correct result 5p, only formula 3p.

(e) Das **Amdahlsche** Gesetz berücksichtigt keinen Overhead, der durchs Parallelisieren auftritt. Lösen Sie dieses Problem, indem Sie die Formel des Amdahlensch Gestetzes um einen additiven Term a_p erweitern, der den Overhead bei p Prozessoren beschreibt. (7)

***Amdahl**'s law does not account for the overhead created by parallelizing an application. Address this shortcoming by changing the formula of Amdahl's law to include an additive term a_p, which describes the parallel overhead when using p CPUs.*

Solution:

$$S_p = \frac{1}{f + \frac{1-f}{p} + a_p}$$

Grading: 2p for a_p at the correct place and formula correct itself.

Ein Programm hat einen seriellen Anteil von 4% und wir wissen, dass der Overhead $a_p = \frac{p}{150}$ ist. Was ist die optimale Anzahl Prozessoren, um das Programm möglichst schnell ablaufen zu lassen? Hinweis: Sie können die unten gegebene Quotientenregel benutzen.

A program has a serial fraction of 4% and we know the overhead is $a_p = \frac{p}{150}$. What is the optimal number of CPUs to allocate to the program in order to minimize its runtime? Hint: you can use the quotient rule given below.

$$\left(\frac{u}{v}\right)' = \frac{u'v - uv'}{v^2}$$

Solution:

Given $f = 0.04$ and $a_p = \frac{p}{150}$, we maximize the above formula:

$$\begin{aligned} \max S_p &= \max \frac{1}{f + \frac{1-f}{p} + a_p} \\ &= \max \frac{1}{0.04 + \frac{1-0.04}{p} + \frac{p}{150}} \end{aligned}$$

Derive and set to zero:

$$S_p' \stackrel{!}{=} 0 \qquad (n > 0)$$

Let

$$h(p) = 0.04 + \frac{1 - 0.04}{p} + \frac{p}{150}$$

Using the quotient rule, we can simplify to:

$$S_p' = \frac{-h'(p)}{[h(p)]^2}$$

$h(p)$ is positive for $p > 0$, thus we only have to find $h'(p) = 0$:

$$0 = -\frac{0.96}{p^2} + \frac{1}{150}$$
$$p = \sqrt{0.96 \cdot 150} = 12$$

Grading: 3p for formula only, 5p for result. 1p for realizing that it is based on a maximization problem that can be solved with a derivative.

Synchronization (30 points)

4. Beschreiben Sie den Unterschied zwischen Parallelismus und Nebenläufigkeit so wie in der Vorlesung besprochen.

   Describe the difference between parallelism and concurrency as it was discussed in the lectures. (4)

Solution:

Parallelismus: Arbeit wird auf Ressourcen aufgeteilt. Erfordert Koordination. (2 pt) Concurrency: Mehrere Anfrage auf beschränkte Ressourcen. Erfordert Synchronisierung. (2 pt)

Grading: 1pt for several / shared ressources each, 1pt for coordination / synchronization each

5. Ein Zoo-Simulationsprogramm hat das folgende Problem: Es gibt zwei Thread-Klassen genannt Rabbit und Lion. Es gibt eine einzige Wasserquelle genannt WaterSource, welche folgendermassen verwendet werden muss:

   In a Zoo simulation program, there exists the following problem: There are two classes of threads called Rabbit and Lion. There is a single water source called WaterSource that must be used in the following way:

   - Tiere (Threads) unterschiedlicher Klasse dürfen nicht gleichtzeitig von der Wasserquelle trinken.

     Animals (threads) of different type may not drink from the water source simultaneously.
   - Jedes Tier (Thread), das von der Wasserquelle trinken will, kommt irgendwann zum trinken.

     Every animal (thread) who needs to drink from the water source eventually succeeds.

   Das Protokoll soll mit den Funktionen `enterLion()`, `leaveLion()`, `enterRabbit()`, `leaveRabbit()` implementiert werden. `enterLion()` blockiert den Caller bis es okay ist für einen Löwen zu trinken. `leaveLion()` wird aufgerufen wann immer ein Löwe fertig ist mit trinken. `enterRabbit()` und `leaveRabbit()` machen dasselbe für Rabbits. Als Beispiel,

   The protocol is implemented via the following four functions `enterLion()`, `leaveLion()`, `enterRabbit()`, `leaveRabbit()`. `enterLion()` delays the caller until it is okay for a lion to drink. `leaveLion()` is called whenever a lion is finished drinking, while `enterRabbit()` and `leaveRabbit()` do the same for rabbits. For example,

```
waterSource.enterLion();
lion.drink(amount);
waterSource.leaveLion();
```

(a) Nennen Sie zwei wichtige Konzepte, die ein korrektes paralleles Programm erfüllen sollte wenn Locks verwendet werden. *Name two important properties for the correct execution of parallel programs in the presence of locks?* (4)

Solution:
Students can name any two of them:

- Mutual exclusion (2pt)
- Starvation freedom / No starvation (2pt)
- Deadlock freedom (2pt) Grading: 2pts for any of them (livelock, data race...)

(b) Implementieren Sie die Klasse WaterSource entsprechend der Spezifikation ohne dabei `synchronized (this){...}` zu benutzen. *Implement the class WaterSource according to the specification without using `synchronized (this){...}`.* (16)

```
public class WaterSource {
    // Lock to protect water source

    Lock waterLock = ..............................;

    // Monitor for notifications

    ............... monitor = ..............................;

    // Number of lions drinking at water source

    ............................... int lionCount = 0;

    // Number of rabbits drinking at water source

    ............................... int rabbitCount = 0;

    void enterLion() throws InterruptedException {

        ................................................;

        while (rabbitCount > 0) {

            ..............................;

            synchronized (monitor) {

                ....................................;
            }

            ..............................;
```

```
        }
        lionCount++;

        ..................................................;
    }


    void leaveLion() {

        ..................................................;

        lionCount--;
        if (lionCount == 0) {
            synchronized (monitor) {

                ....................................;
            }

        }

        ..................................................;
    }


    void enterRabbit() throws InterruptedException {

        ..................................................;

        while (lionCount > 0) {

            ............................;

            synchronized (monitor) {

                ....................................;
            }

            ............................;
        }
        rabbitCount++;

        ..................................................;

    }


    void leaveRabbit() {
```

```
            ..................................................;


            rabbitCount--;
            if (rabbitCount == 0) {
                synchronized (monitor) {

                    ..................................;
                }
            }

            ..................................................;

    }
}
```

Solution:

```
public class WaterSource {

    // 2pts   -1 for use of Interface, e.g. new Lock()
    Lock waterLock = new ReentrantLock();
    // 1pt
    Object monitor = new Object();

    // 1pt  -1/2 for atomic
    volatile int lionCount = 0;
    // 1pt  -1/2 for atomic
    volatile int rabbitCount = 0;

    void enterLion() throws InterruptedException {
        waterLock.lock(); // 1pt
        while (rabbitCount > 0) {
            waterLock.unlock(); // 1pt
            synchronized (monitor) {
                monitor.wait(); // 2pts - 1 if outside of
                   synchronized
            }
            waterLock.lock(); / 1pt
        }
        lionCount++;
        waterLock.unlock(); // 1pt
    }

    void leaveLion() {
        waterLock.lock(); // 1/2 pt
        lionCount--;
```

```
        if (lionCount == 0) {
            synchronized (monitor) {
                monitor.notifyAll(); // 2pts - 1 if
                    outside of
                     // synchronized or just notify()
            }
        }
        waterLock.unlock(); // 1/2 pt
    }


    // 1 pt for rest of locks being correct, i.e. global
       correctness.

    // 1 pt flexibly distributed, e.g. if everything looks
       about
    // right. Not sure if that was a good call, but for a
       question
    // like this, it seemed good to have some flexibility
       to scale up and
    // down a bit. For example, I would not give this
       point if the use
    // of wait/notify was not clean, or missing, or if
       there can be a
    // deadlock. This is kind of a mechanism to subtract
       points
    // independent of the rest of the grades.

    void enterRabbit() throws InterruptedException {
        waterLock.lock();
        while (lionCount > 0) {
            waterLock.unlock();
            synchronized (monitor) {
                monitor.wait();
            }
            waterLock.lock();
        }
        rabbitCount++;
        waterLock.unlock();
    }

    void leaveRabbit() {
        waterLock.lock();
        rabbitCount--;
        if (rabbitCount == 0) {
            synchronized (monitor) {
```

```
                monitor.notifyAll();
            }
        }
        waterLock.unlock();
    }
}
```

(c) Erklären Sie, ob und weshalb Ihre Implementierung die beiden in Teilaufgabe a genannten Eigenschaften erfüllt.

Explain why your water source implementation does or does not satisfy the two properties given in part a of this question. (6)

Solution: Grading: 3 Points one correct explanation of a property. 6 if both explanations are present and correct. -1 point for an incorrect statement i.e. there is no starvation. however there is!

OpenMP & Data Parallelism (12 points)

6. Theoriefragen. Für die Checkbox-Fragen gibt nur die korrekte Antwort 1pt. Falsch markierte Checkboxen resultiert in -1pt. Die Minimale Punktzahl ist 0.

Theory questions. For the checkbox questions only the correct answer gives 1pt. Wrongly marked checkboxes result in -1pts. The minimum number of points is 0.

   (a) Was beschreibt das Programm im Data-Parallelism (was ist das Programmier-Modell)? (1)

What does the program describe in data parallelism (what is the programming model)?

   - √ **Was getan werden muss (deklarativ).** *What needs to be done (declarative).*
   - □ Wie die Arbeit gemacht werden soll (imperativ). *How the work should be done (imperative).*
   - □ Wie Daten-Objekte interagieren (objekt-orientiert). *The interaction between data objects (object oriented).*

   (b) Welche der folgenden Klassen von Patterns sind geeignet, um mit Data-Parallelism implementiert zu werden? (3)

Which of these patterns are well suited to be implemented using data parallelism?

   - √ **Map: Führe eine Funktion auf jedem Element einer Liste/Sammlung aus** *map: Execute a function on each element of a list/collection*
   - □ Komplexe Synchronisierung von Berechnungen *Complex synchronization of computations*
   - √ **Reduktionen: Berechne die Summe, das Maximum/Minimum etc. einer Liste/Sammlung** *reductions: Calculate the sum/maximum/minimum/... of a list/collection.*
   - √ **Filter: Wähle einige Element einer Liste/Sammlung basierend auf einer Filtrierungs-Funktion, welche auf jedes Element angewendet wird, aus** *filter: select some elements of a list/-collection based on a filtering function which is applied to each element.*
   - □ Berechnungen mit komplexen Nachrichten-Patterns zwischen Tasks *Computations with complex communcation patterns between tasks*

(c) OpenMP Programmier-Frage: In dieser Frage sollen Sie einen parallelen Sortieralgorithmus implementieren, welcher wie folgt funktioniert: Zuerst sollen Sie mit dem Quicksort-Algorithmus n Teile eines Arrays parallel, und in-place sortieren (n ist die Anzahl Threads, welche von der OpenMP Laufzeitumgebung zur Verfügung gestellt wird), und dann alle sortierten Array-Teile mergen, um das sortierte Array zu erstellen. Sie können die Funktionen `merge` und `quicksort`, wie unten gezeigt, verwenden. Weiterhin können Sie annehmen, dass die OpenMP Laufzeitumgebung Ihnen eine Zweierpotenz von Threads zur Verfügung stellt und dass die Länge des Input-Arrays eine Zweierpotenz ist.

OpenMP programming question: In this question, you should implement a parallel sort algorithm that works as follows: First use quicksort to sort n chunks of an array in parallel, and in-place (n being the number of threads provided by the OpenMP runtime) and then merge all the sorted chunks to produce the sorted array. You are given the functions `merge` and `quicksort` as shown below. You can assume that the OpenMP runtime will provide a power-of-two amount of threads and that the input array is power-of-two sized. (8)

```
// from the OpenMP API:
// return the number of threads provided by the OpenMP
   runtime
int omp_get_num_threads();

// quicksort: sort elements [start, end) in the given array
   in-place.
void quicksort(int *array, int start, int end);

// merge: merge sorted elements [start, mid) and [mid, end)
   in the given array in-place
void merge(int *array, int start, int mid, int end);
```

Vervollständigen Sie das gegebene Programm auf der nächsten Seite.

Complete the program on the following page.

```
void parallel_sort(int *array, int length)
{
  // Number of parallel executions
  int runs = 0;
  // Number of elements to be sorted by each parallel thread
  int run_size = 0;

  #pragma omp parallel
  {

    runs = ..............................;

    run_size = ..............................;
    // Get index for current thread from 0 to
       omp_get_num_threads()-1
    int thread_id = omp_get_thread_num();

    int start = ..............................;

    int end   = ..............................;
    // sort interval [start, end)
    quicksort(array, start, end);
  }

  // recursively merge already sorted parts of the array
     until whole array sorted
  while (runs > 1) {
    for (int i = 0; i < runs; i += 2) {

      merge(array, .....................................);
    }

    runs = ..............................;

    run_size = ..............................;
  }
}
```

Solution: Notes:

- See code for distribution of points

(1) 0pts when they try to use `sizeof(array)` instead of `length`

(2) 0pts if `end = start + run_size - 1` or equivalent.

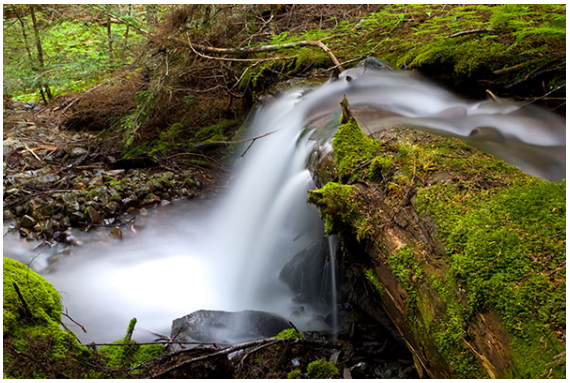
(3) 1pt if `expr - 1` for mid or end.

```
void parallel_sort(int *array, int length)
{
  // Number of parallel executions
  int runs = 0;
  // Number of elements to be sorted by each parallel
     thread
  int run_size = 0;

  #pragma omp parallel
  {
    runs = omp_get_num_threads(); // [1p]
    run_size = length / runs; // [1p (1)]
    // Get index for current thread from 0 to
       omp_get_num_threads()-1
    int thread_id = omp_get_thread_num();
    int start = thread_id * run_size; // [1p]
    int end   = (thread_id+1) * run_size; // [1p (2)]
    // sort interval [start, end)
    quicksort(array, start, end);
  }

  // recursively merge already sorted parts of the array
     until whole array sorted
  while (runs > 1) {
    for (int i = 0; i < runs; i += 2) {
      merge(array, i*run_size, (i+1)*run_size,
         (i+2)*run_size); // [2p (3)]
    }
    runs /= 2; // [1p]
    run_size *= 2; // [1p]
  }
}
```

OpenCL (12 points)



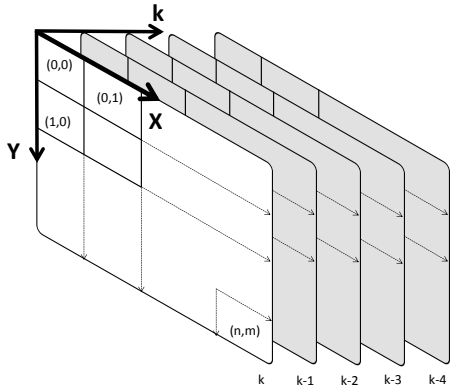


Abbildung 1

7. (a) Neben dem Studium arbeiten Sie an einem Bildbearbeitungssprogramm. Sie wollen einen Filter einbauen, der den Langzeitbelichtungseffekt (siehe Abb. 1) approximiert. Solch ein Effekt lässt sich erziehlen, indem man 5 Bilder ohne Kamerabewegung aufnimmt und den Durchschnitt aus den 5 Bildern berechnet.
Implementieren Sie einen OpenCL-Kernel für den Zeit-Smooting-Effekt. Der Kernel erhält als Input ein Schwarzweiss-Bild, repräsentiert durch einen Vektor von Integerwerten zwischen 0 und 255 (0 für Schwarz und 255 für Weiss). Der Kernel wird für jedes Pixel (x, y) des Output-Bildes aufgerufen und verarbeitet die 5 Eingabepixel (x, y, k), $(x, y, k-1)$, $(x, y, k-2)$, $(x, y, k-3)$ und $(x, y, k-4)$. (8)

Beside your studies you work on an image editing software. You want to approximate the long exposure effect with a filter in your program. You know that you can implement this effect by averaging 5 images taken from the exact same position.
Implement the time-smoothing effect in an OpenCL-Kernel. The kernel takes a black and white image as input represented by a vector of integer values between 0 and 255 (0 for black and 255 for white). The kernel will be called for each pixel (x, y) of the output image and processes 5 input pixels (x, y, k), $(x, y, k-1)$, $(x, y, k-2)$, $(x, y, k-3)$ and $(x, y, k-4)$.

- Die 5 Bilder werden als **ein** Vektor mit folgender Grösse gespeichert:

  *The 5 input images are stored as **one** vector with size:*

  $$width \cdot height \cdot 5$$

- Den Index i für ein Pixel im Bildvektor mit den Koordinaten $[x, y, k]$ können sie mit der Formel berechnen:

  The index i for a pixel in the image vector with the coordinates $[x, y, k]$ can be computed with the formula:

  $$i = x + y \cdot width + k \cdot (width \cdot height)$$

- Der gefilterte Pixelwert von Pixel (x, y) ist der Durchschnitt aus den 5 Eingabepixeln:

  The filtered pixel value at location (x, y) is the average of the 5 input-pixels:

  $$p(x, y) = \frac{1}{5} \sum_{i=1}^{5} I_i(x, y)$$

```
__kernel void TimeFilter(__global uchar* input,
  __global uchar* output)
{
  // find current coordinates in the cube
  int x = get_global_id(0);
  int y = get_global_id(1);


  // find frame size
  int frameSize = get_global_size(0)*get_global_size(1);
  // find line size
  int width = get_global_size(0);
```

Solution:

```
  float ik =   (float)input[x+y*width+0*width*height];
  float ik_1 = (float)input[x+y*width+1*width*height];
  float ik_2 = (float)input[x+y*width+2*width*height];
  float ik_3 = (float)input[x+y*width+3*width*height];
  float ik_4 = (float)input[x+y*width+4*width*height];
  //  (5pt) (1 for right type conversion, 2 for right
     indexing, 2 for indexing all 5 pixels)

  // There is also a solution with a for loop which

  float res = (float)(1/5.0*(ik+ik_1+ik_2+ik_3+ik_4));
  //  (1pt)

  // The math operation here is applied to each element of
  // the unsigned char vector and the final result is
     applied
  // back to the output image
  output[x+y*width] = (uchar)res;
  // (2pt) (1 for right output pixel index, 1 for typecast)
```

```
}
```

(b) Hinweis: Diese Teilaufgabe lässt sich unabhängig von der Lösung der ersten Teilaufgabe bearbeiten. Ein Histogramm lässt sich zur Beurteilung der Bildqualität ihres Filters aus Aufgabe (6a) verwenden.
Ein Histogramm bildet die Helligheitsverteilung des Bilds ***A*** *in einem Array* ***B*** *ab, wie in Bild 2 zu sehen ist. Haben im ganzen Bild* ***A*** *z.B. 100 Pixel den Wert 0 (schwarz) und 134 Pixel den Wert 1, dann setzen wir* ***B****[0]=100 und* ***B****[1]=134. Diesem Schema folgend werden alle Pixel gemäss ihrem Helligkeitswert katalogisiert.*

Hint: This question can be solved independently from the previous question. A histogram can be used to measure the quality of the output of your filter from question (6a).
A histogram ***B*** *is a count (frequency) of all intensity values in an image* ***A****. For example, if the number of pixels with the value 0 is 100, and 134 pixel have the value 1 then we set* ***B****[0]=100 and* ***B****[1]=134. According to this rule, the algorithm should count all intensity values and fill in the bins of the histogram.* (4)

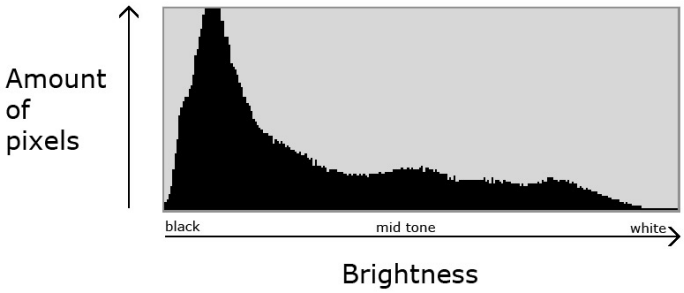


Abbildung 2

Der folgende Code stellt zwei alternative Implementierungen der Histogrammberechnung dar. N ist die Anzahl Pixel (Konstante).

The code below provides two alternative implementations of a histogram computation in Open CL. N is a constant holding the number of pixels.

Listing 1: Implementation A

```
kernel void hist(local int* A,
                 local int* B)
{
  int id = get_global_id(0);
  int t = A[id];
  atomic_inc(&B[t]);
}
```

Listing 2: Implementation B

```
kernel void hist(local int* A,
                 local int* B)
{
  int id = get_global_id(0);
  int t = A[id];
  for(int j=0; j<N; j++)
  {
     if(id == j)
        B[t]++;
     barrier();
  }
}
```

Analysieren Sie die beiden Kernel bezüglich ihrer Korrektheit (1pt) und bzgl. des Parallelismus (3pt) (und der daraus resultierenden Effizienz).

Read the code carefully and provide an analysis of the kernels in terms of correctness (1pt) and parallelism (3pt) (and the resulting efficiency).

Solution:

Correctness: Both Implementations are correct (this is the only valid answer) (1pt)
Parallelism:

- Kernel A has more parallelism as Kernel B (1pt).
- Impl A only needs synchronization if two threads write to the same bin; otherwise threads are fully independent (1pt).
- Impl B is a fully sequential implementation as all work-items need to encounter barrier (hence loop over all pixels) but this means all work-items synchronize on write and have to wait until all other work-items have finished looping over the image (1pt).

Linearizability and Sequential Consistency(12 points)

8. In den folgenden Aufgaben seien A, B und C Threads, welche mit einem gemeinsam benutzten Stack-Objekt arbeiten. Die Spezifikation ist in der folgenden Interfacedefinition ersichtlich.

In the following questions let A, B and C denote threads operating on a shared stack object as specified in the listing below.

```
public interface Stack {
    /* pushes element v onto the stack */
    public void push(int v);

    /* removes the top element and returns it */
    public int pop();

    /* returns the top element */
    public int top();
}
```

(a) Welches der folgenden Szenarien ist linear konsistent? Markiere entweder den Linearisationspunkt oder erkläre, warum es nicht linear konsistent ist.

Which of the following scenarios are lineraly consistent? Either mark the point of linearization or explain why it is not linearly consistent. (4)

1.
```
A s.push(1):  *--------------*
B s.push(2):         *-------------*
A s.pop()->2:                                *--------------------*
B s.pop()->1:                          *----------------------*
```

2.
```
A s.push(1):  *----------*              *----------*
A s.pop()->2:                  *-------*
B s.push(2):       *----------*                        *------------*
B s.pop()->1:                             *-------*
C s.push(2):                                 *-----------*
C s.pop()->1:                    *--------*
```

3.
```
A s.push(1):  *------------------------*
A s.pop()->2:                           *--------------*
B s.push(2):      *------------*
B s.pop()->1:                   *------*
C s.top()->2:           *--------------*
C s.top()->1:                *----------------------*
```

.....

.....

4.
```
C s.push(2):                              *-------------------*
A s.push(2): *----------*
B s.top()->1:          *--------------------*
A s.pop()->1:                               *---------------*
C s.push(1):               *----*
B s.pop()->2:                    *----------*
```

.....

.....

Solution: 1, 2, 3 are linearizable. 4 is not. (1 pt each)

1.
```
A s.push(1):  *---x---------*
B s.push(2):         *-----x------*
A s.pop(2):                              *---x------------*
B s.pop(1):                          *---------------x---*
```

2.
```
A s.push(1):  *---x------*              *-x--------*
A s.pop(2):                   *-x-----*
B s.push(2):       *---x------*                        *-------x-*
B s.pop(1):                               *-----x-*
C s.push(2):                                 *-------x---*
C s.pop(1):                    *----x---*
```

3.
```
A s.push(1):  *--------------x-----------*
A s.pop(2):                              *----------x----*
B s.push(2):      *-------x----*
B s.pop(1):                    *--x---*
C s.top(2):             *-------------x*
C s.top(1):                  *--x-------------------*
```

4. C s.push(2): *------------------*
   A s.push(2): *-----x----*
   B s.top(1) *----------x--------*
   A s.pop(1) *--------------*
   C s.push(1): *--x-*
   B s.pop(2): *-------#--*

   The operation with # cannot be satisfied.

(b) Gegeben sind folgende Historien H und G. Beantworten sie folgende Fragen für beide Historien. Begründen sie ihre Antworten. Richtig angekreuzte Antworten geben einen Punkt. Richtige Begründung liefert einen weiteren Punkt. Falsch angekreuzte Antworten geben einen Minuspunkt. Die minimale Punktzahl ist 0.

Consider the following two histories H and G. Answer the following questions for both histories. Justify your answer. Correct checkbox answers receive 1 point. Correct justification delivers another point. Wrong checkbox answers receive a negative point. The minimum number of points is 0. (6)

```
H= A s.push(x)      G= A s.push(x)
   B s.pop()           A s:void
   B s:x               A s.push(y)
   A s:void            A s:void
   A s.push(y)         B s.pop()
   B s.push(y)         B s:x
   A s:void            A s.pop
   A s.pop             A s:y
   A s:y               B s.push(y)
   B s:void            B s:void
```

Sind die Historien equivalent? *Are these histories equivalent?*
`[ ] Yes [ ] No` *`[ ] Yes [ ] No`*

(c)

. . . . . .

. . . . . .

Sind sie sequentiell? *Are they sequential?*
`H: [ ] Yes [ ] No` *`H: [ ] Yes [ ] No`*
`G: [ ] Yes [ ] No` *`G: [ ] Yes [ ] No`*

(d)

. . . . . .

. . . . . .

Sind sie legal? *Are they legal?*
`H: [ ] Yes [ ] No` *`H: [ ] Yes [ ] No`*
`G: [ ] Yes [ ] No` *`G: [ ] Yes [ ] No`*

(e)

. . . . . .

. . . . . .

Solution: Just the ticks: negative points for wrong ticks.

2pt: *Two histories are equivalent if:* $H|A == G|A$ *and* $H|B == G|B$. H and G are equivalent. 1pt for the tick, 1pt for the explanation.

```
H|A=  A s.push(x)      G|A= A s.push(x)
      A s:void                A s:void
      A s.push(y)             A s.push(y)
      A s:void                A s:void
      A s.pop                 A s.pop
      A s:y                   A s:y

H|B=  B s.pop()        G|B=   B s.pop()
      B s:x                   B s:x
      B s.push(y)             B s.push(y)
      B s:void                B s:void
```

2pt: *Sequential history: Method calls of different threads do not interleave.* History G is sequential, H is not (calls 1, 3, 4). 1/2 pts for the ticks 1/2 pts for the explanation. (for G and H)

```
H= A s.push(x) 1    G= A s.push(x) 1
   B s.pop()   2       A s:void    1
   B s:x       2       A s.push(y) 2
   A s:void    1       A s:void    2
   A s.push(y) 3       B s.pop()   3
   B s.push(y) 4       B s:x       3
   A s:void    3       A s.pop     4
   A s.pop     5       A s:y       4
   A s:y       5       B s.push(y) 5
   B s:void    4       B s:void    5
```

2pt: *A sequential history H is legal, if for every object x H|x adheres to the sequential specification of x* H is not sequential, thus not legal. G is sequential, but: push(x), push(y) pop(x) violates the sequential specification of the stack. Not legal. 1/2 pts for the ticks 1/2 pts for the explanation. (for G and H

(f) Ist die Komposition mehrerer sequentiell konsistenter Objekte selbst immer sequentiell konsistent? Belegen Sie Ihre Antwort mit einem (Gegen-)Beispiel.

Is the result of composing multiple sequentially consistent objects itself always sequentially consistent? Justify your answer with a (counter-)example. (2)

Solution: NO. 1/2 pts. Proof by counter example:

```
A:  *-- p.enq(x) --*  *-- q.enq(x) --*      *-- p.deq(y) --*
B:          *-- q.enq(y) --*    *-- p.enq(y) --*    *-- q.deq(x) --*
|
<--   can be moved forward  -----
```

Both objects, p and q are sequentially consistent. But we cannot reorder operations within the same thread!

The execution as a whole is not. We have a cycle:
$A : q.enq(X) \rightarrow B : q.enq(y)$ and $B : p.enq(Y) \rightarrow A : p.enq(X)$

program order gives us:
$A : p.enq(X) \rightarrow A : q.enq(X)$ and $B : q.enq(y) \rightarrow B : p.enq(y)$

Give only little points for the correct answer (this is easy to guess for them way we ask the question: älways sequentially consistent", and a lot of points for the justification.