Name, Vorname (Druckbuchstaben): _____

Legi Nummer: _____

252-0024-00L
Parallele Programmierung
ETH/CS: FS 2016
Basisprüfung
Montag, 15.8.2016
120 Minuten

Diese Prüfung enthält 24 Seiten (inklusive dieses Deckblatts) und 9 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, fordern Sie bei den Assistenten zusätzliche Blätter an. Versehen Sie die Aufgabe gegebenfalls mit einem klaren Hinweis. 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	10	
2	6	
3	8	
4	10	
5	8	
6	11	
7	12	
8	7	
9	8	
Total:	80	

Sequential Java(10 points)

Programmierung

Programming

1. (a) Gegeben sei die rekursive Methode `fRec`. Vervollständigen Sie die iterative Methode `f` so, dass sie das selbe Ergebnis berechnet wie Methode `fRec`, aber keine Rekursion verwendet wird. Hinweis: zeichnen Sie den Rekursionsbaum, falls hilfreich.

Given the recursive method `fRec`, complete the iterative method `f` below such that it computes the same result as `fRec` but does not use recursion. Hint: draw the recursion tree, if helpful. (5)

```
static int fRec(int n){
    if (n<=0)
        return 10;
    else if (n==1)
        return 20;
    else
        return 2*(fRec(n-1)+fRec(n-2));
}

static int f(int n){
    if (n<=0)
        return 10;
    else{
        int res=20;
        Solution: (below)
        return res;
    }
}
```

Solution:

```
int fnm1 = res;
int fnm2 = 10;
while (n > 1){
    res = 2*(fnm1+fnm2);
    fnm2 = fnm1;
    fnm1 = res;
    --n;
}
```

Code verstehen *Code interpretation*

(b) Gegeben sei folgende Methode: *Consider the following method:* (3)

```
public static void increment(int a, Integer b, int[] c,
   IntWrapper d){
    a++;
    b++;
    c[0]++;
    d.v++;
}
```

Betrachten Sie die Klasse `IntWrapper`: *Consider the class* `IntWrapper`*:*

```
class IntWrapper {
    public int v;
    IntWrapper(int value){
        this.v = value;
    }
}
```

Nun betrachten Sie den folgenden Code: *Now consider the following code:*

```
int A = 1;
Integer B = 1;
int[] C = {1};
IntWrapper D = new IntWrapper(1);
increment(A, B, C, D);
System.out.println(A+" "+B+" "+C[0]+" "+D.v);
```

Geben sie die Werte von A, B, C[0] und D.v nach Ausführung des Codes an und begründen sie ihre Antwort für jede der Variablen kurz.

Determine the values of A, B, C[0] and D.v after executing the given code and justify your answer shortly for each variable.

A:

Solution: 1 – call by value

B:

Solution: 1 – call by value

C[0]:

Solution: 2 – ein array ist eine Referenz auf die Daten

D.v:

Solution: 2 – d wird als Referenz übergeben

Multiple-choice

Multiple-choice

(c) Welche der untenstehenden Deklarationsanweisungen ist **inkorrekt** in Java?

*Which is **not** a correct variable declaration statement in Java?* (1)

- ☐ `int _a`
- ☐ `Integer[ ] b`
- √ `float 3var`
- ☐ `static final byte d = 0`

(d) Carefully read the following code:

Consider the following Code: (1)

```
int[] a = new int[5];
int i = 0;
do {
    System.out.print(a[++i]);
} while(i < 5);
```

Was gibt der Code aus?

What is the output of this code?

- ☐ 0000 — *0000*
- √ **nach der Ausgabe 0000 wird eine Exception geworfen** — *after printing 0000 an exception i thrown*
- ☐ 00000 — *00000*
- ☐ nach der Ausgabe 00000 wird eine Exception geworfen — *after printing 00000 an exception i thrown*

Speedup, Amdahl, Gustafson (6 points)

2. Beantworten Sie die folgenden Fragen zu Speedup sowie zum Gesetz von Amdahl und Gustafson:

Answer the following questions about speedup, 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:

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

S: Speedup
p: Number of processors
f: sequential fraction of any parallel process
Grading: 1/2 points for correct formula, 1/2 for each correct parameter description.

(b) Gehen Sie davon aus, dass 20% der Implementierung eines Algorithmus nicht parallelisierbar sind. Was ist der maximal erreichbare Speedup nach dem **Amdahlschen** Gesetz? (2)

Assume that 20% of the code of an algorithm cannot be parallelized. What is the maximally achieveable speedup according to ***Amdahl's*** *law?*

Solution:

$$f = 0.2;$$
$$S_p = \frac{1}{f + \frac{1-f}{p}}$$

with $p \to \infty$ we get $S_\infty = 5$.
Grading: 1p for formula only, 1p for result.

(c) Ihnen steht ein Prozessor mit 47 Kernen zur Verfügung, um ein Programm parallel auszuführen. Insgesamt sind 94% des Programmes parallelisierbar. Alle parallelisierbaren Teile können ohne weitere Overheads mit der selben Laufzeit auf allen Cores ausgeführt werden. Welcher Speedup lässt sich gemäss **Amdahlschen** Gesetz erziehlen? (2)

Imagine you have a processor which has 47 cores and a program you want to parallelize. 94% of this program can be parallelized. Moreover, the program can be executed at the same speed on all those cores without additional overheads. By using ***Amdahl****'s law, what is the parallel speedup achievable?*

Solution:

$$S_p = \frac{1}{f + \frac{1-f}{p}} = \frac{1}{0.06 + 0.94/47} = 1/0.08 = 100/8 = 12.5$$

Grading: -1 if math is wrong. -1 if result is wrong. Formula follows from previous part.

Pipelining (8 points)

3. "Wash it yourself" bietet einen Raum zum Waschen, Trocknen und Bügeln von Kleidern. Hierfür steht eine Waschmaschine, ein Trockner und ein Bügeleisen zur Verfügung. Um die Maschinen stärker aus zu nutzen, beschloss die Firma mehrere Kunden gleichzeitig in den Waschraum zu lassen. Diese können die Maschinen nun wie im Pipeline Modell in Figure 1 dargestellt nutzen. Einen kompletten Zyklus von Waschen, Trocknen und Bügeln bezeichnen wir als 'Runde'

"Wash it yourself" is a service where customers can wash, dry and iron their clothes. There is one washing machine, one tumbler and one ironing board. To fully utilize the laundry room the company decided to allow more than one client to access the laundry room at the same time. They can use the machines according to the pipeline model shown in Figure 1. A complete cycle of washing, drying and ironing is called a 'round'.

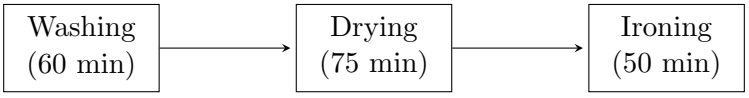


Abbildung 1: Pipeline

(a) Berechnen Sie den Durchsatz der Pipeline in Runden/Stunde. *Calculate the throughput of the pipeline. The unit is rounds/hour.* (2)

Solution: $60/75 = 0.8$ [Rounds/Hour]

(b) Ist die Pipeline balanciert oder unbalanciert? Was ist die Latenz der ersten und was die Latenz der zweiten Runde? Bitte begründen Sie Ihre Antwort mit einer Zeichnung. *Is the pipeline balanced or unbalanced? What is the latency of the first and second round? Please justify your answer with a drawing.* (3)

Solution: The pipeline is unbalanced. The first round takes: $60 + 75 + 50 = 185$ min. The second round takes: $75 + 75 + 50 = 200$ min. In the second round we have to wait until the tumbler becomes available.

(c) Wenige Jahre später entdeckt die Firma eine neuartige Maschine die sich "WashDryIron" nennt. Diese Maschine kann die Kleider in 240 Minuten waschen, trocknen und bügeln. Die Firma überlegt nun, ob Sie die drei alten Maschinen (Waschmaschine, Trockner und Bügeleisen) durch drei identische "WashDryIron" Maschinen ersetzen soll. Die folgenden Szenarien werden betrachtet: *After some years of service, the company discovers a novel 3-in-1 machine called "WashDryIron". This machine can wash, dry and iron the clothes in 240 minutes. The company wants to decide if they should replace the three current machines (washing machine,tumbler and iron), with three identical "WashDryIron" machines. The following scenarios are considered:* (3)

1.) Die Maschinen sind zuerst ungenutzt und es kommen gleichzeitig drei Kunden zum Waschen, Trocknen und Bügeln. Welche Konfiguration ist hier besser, also wie lange benötigt die alte (Waschmaschine, Trockner und Bügler) im Vergleich zur neuen (drei WashDryIron Maschinen) Konfiguration?

1.) The machines are initially unused and three customers at once come to wash, dry and iron. Which configuration is better, i.e. how long does the old (washing machine, dryer and iron) compared to the new setup (three WahsDryIron machines) take?

Solution: alt: 185 + 75 + 75 = 335 Minuten
neu: 240 Minuten
neu ist besser.

2.) Es kommen so viele Kunden, dass die Maschinen praktisch ständig laufen. Welche Konfiguration hat den höheren Durchsatz? Geben Sie die Zahlen an.

2.) There are so many customers that the machines are practically in permanent use. Which setup has the higher throughput? Provide the numbers.

Solution: old: 1 / 75 Minutes, new: 3/240 Minutes = 1/80 =¿ old is better

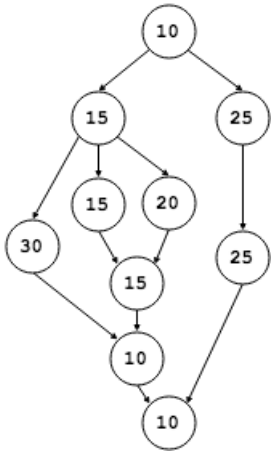
Task Parallelism (10 points)

4. Task Graphen

Betrachten Sie folgenden Task-Graphen. Knoten enthalten die jeweilige Laufzeit:

Task Graphs.

Consider the following task graph. Each node contains its execution time:



(a) Geben Sie die untere und obere Schranke für die Ausführungszeiten an. Was ist der maximale Speedup (ausgehend von unendlich vielen Prozessoren)? Wieviele Prozessoren benötigen Sie tatsächlich, um diesen Speedup auf obigen Task Graphen zu erreichen? Erklären Sie Ihre Antwort kurz. (3)

Give lower and upper bounds on execution times. What is the maximum possible speed up (assuming infinite processors)? What is the number of processors required to achieve the maximum speedup on the given Task Graph? Briefly explain your answer.

Solution: $T_1 = 25*2+10*3+15*3+20+30 = 175$, $T_\infty = 10+15+20+15+10+10 = 80$, Lower bounds: $T_p \geq (T_1/p) \geq T_\infty$ upper bound: $T_p \leq (T_1/p) + T_\infty$ Maximum possible speedup $= T_1/T_\infty = 2.187$, 4 processors are sufficient to achieve maximum speed up. A maximum of 4 nodes can be available for processing at any given time. 0.5 points each for upper and lower bound. 1 point for correct number of processors and 1 point for correct rationale.

(b) Geben Sie das beste Scheduling Szenario für eine 2-Prozessor-Maschine an (schematisch wenn nötig). Welchen Speedup erhalten Sie? (3)

Provide the best scheduling scenario for a 2 processor machine (schematically if necessary). What speedup do you get?

Solution:

```
P1: n1(10), n2(15), n6(20), n5(15), n7(15), n9(10)
P2: ------, n3(25)  , n4(30)       , n8(25)      , n10(10)
T2 = 10+25+30+25+10 = 100, speedup = 175 / 100 = 1.75.
```

(c) Ausgehend von einer N-core Maschine, sollen Sie nie mehr als N tasks gleichzeitig zur Ausführung absetzen, um Overheads zu vermeiden. Untenstehendes Java Programm verwendet das ForkJoin Framework zur Implementation einer Reduce Operation. Füllen Sie die Lücken aus. Hinweis: denken Sie über den sequentiellen Cutoff nach.

Given a multi-core machine you are advised not to issue more parallel tasks than there are cores; otherwise your program will slow down. The following Java program uses the ForkJoin Java framework to parallelize a Reduce function. Fill in the missing lines of code. Hint: think about the sequential cutoff. (4)

```
public class ReduceClass extends RecursiveTask<Integer> {

 private int[] input;
 private int start;
 private int finish;
 private int processors;

 private ReduceClass (int[] input, int start, int finish,
    int processors) {
    this.input  = input;
    this.start  = start;
    this.finish = finish;
    this.processors = processors;
 }

protected Integer reduce(){

 if ( ..........................  ) {
    return computeReduce( array, start, finish );
 }
  Solution: (below)
}
```

Solution:

```
protected Integer reduce(){

if ( processors == 1 ) {
   return computeReduce( array, start, finish );
}
int mid = (finish - start) / 2;
int p1  = processors / 2;
ReduceClass t1 = new ReduceClass(array, start,
   start+mid, p1);
ReduceClass t2 = new ReduceClass(array, start+mid,
   finish, processors - p1);
```

```
        t1.fork();
        Integer r2 = t2.reduce();

        return r2 + t1.join();
      }
```

Synchronization (8 points)

5. Bei den folgenden Multiple-Choice Fragen gibt es einen Punkt, wenn alle korrekten Möglichkeiten angekreuzt wurden. Andernfalls gibt es 0 Punkte.

For the following multiple choice questions, 1 point is provided if and only if all correct answers are marked, otherwise 0 points are given.

(a) Welche der folgenden Situationen führen zu einer Race Condition, wenn keine Synchronisationsmechanismen eingesetzt werden? (1)

Which of the following situations leads to a race condition provided that no synchronisation mechanism is in place?

- √ **Ein Thread A liest eine Variable x zur gleichen Zeit wie ein Thread B in diese Variable x schreibt.** — *A thread A reads a variable x at the same time with a thread B that writes this variable x.*
- □ Ein Thread A liest eine Variable x zur gleichen Zeit wie ein Thread B diese Variable x liest. — *A thread A reads a variable x at the same time with a thread B that reads this variable x.*
- √ **Ein Thread A schreibt eine Variable x zur gleichen Zeit wie ein Thread B diese Variable x schreibt.** — *A thread A writes a variable x at the same time with a thread B that writes this variable x.*
- □ Ein Thread A schreibt eine Variable x zur gleichen Zeit wie ein Thread B eine Variable y schreibt. — *A thread A writes a variable x at the same time with a thread B that writes a variable y.*
- □ Ein Thread A liest eine Variable x zur gleichen Zeit wie ein Thread B eine Variable y schreibt. — *A thread A reads a variable x at the same time with a thread B that writes a variable y.*
- □ Ein Thread A liest eine Variable x zur gleichen Zeit wie ein Thread B eine Variable y liest. — *A thread A reads a variable x at the same time with a thread B that reads a variable y.*

(b) Welche der folgenden Sätze sind wahr? (1)

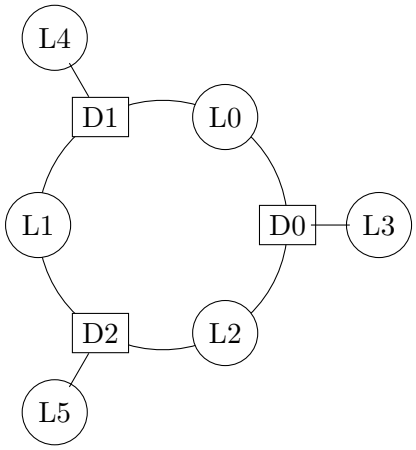
Which of the following sentences are true?

- √ **Es ist möglich, Nebenläufigkeit auf einem einzelnen Core zu haben.** — *It is possible to have concurrency on a single core.*
- √ **Es braucht mindestens zwei Cores, um eine parallele Ausführung zu erreichen.** — *At least two cores are required to achieve parallel execution*
- □ Eine Art, Parallelismus zu erreichen, ist der Gebrauch von mehreren Threads. Andererseits kann Nebenläufigkeit durch einen einzelnen Thread erreicht werden. — *One way to achieve parallelism is the use of multiple threads. On the other hand, concurrency may be achieved with a single thread.*
- √ **Parallelismus befasst sich mit dem schnelleren Lösen von Problemen.** — *Parallelism aims at solving a problem faster.*
- √ **Nebenläufigkeit befasst sich mit dem korrekten und effizienten Zugriffsmanagement auf geteilte Resourcen.** — *Concurrency aims at correct and efficient management of access to the shared resources.*

(c) Folgende Abbildung zeigt ein rundes Gebäude mit drei Türen und sechs Lichtern. Jede Türe hat jeweils ein Licht auf beiden Seiten und eins oberhalb, also insgesamt drei benachbarte Lichter. Die Lichter werden durch die Klasse `Light` repräsentiert, die Türen durch die Klasse `Door`. Das Öffnen der Türen ist trickreich: jede Tür hat einen Schalter, welcher die erreichbaren drei Lichter umschaltet (an wird zu aus, aus wird zu an). Nur wenn alle drei Lichter einer Tür angeschaltet sind, öffnet sich die Tür. Die Türschalter aller Türen können gleichzeitig benutzt werden.
Vervollständige die Methode `tryOpen()` in der Klasse `Door` so, dass zuerst die drei erreichbaren Lichter umgeschaltet werden (toggle) und `true` zurückgegeben wird genau dann, wenn dann alle drei Lichter eingeschaltet sind.
Eine beliebige Anzahl Threads darf auf den Türen operieren. Erklären Sie Ihr Synchronisationsschema kurz und bündig.

The following figure depicts a building with three doors and six lights. Each door has one light on each side and one above it, i.e., three adjacent lights. The lights are represented with the class `Light`*, the doors are represented with the class* `Door`*. Opening a door is tricky: there is a switch that toggles (on becomes off, off becomes on) the adjacent lights and if and only if then all three adjacent lights are finally on, the door will open.*
Complete the method `tryOpen()` *of the* `Door` *class such that it first toggles its left, right and top light and returns true if and only if all lights are on.*
An arbitrary number of threads is allowed to act on the doors. Explain in brief your synchronization scheme.

(6)




```
public class Light {
   private final int id; // unique
   private boolean on = false;

   public Light() {
    // set unique id omitted for brevity
   }
   public int getID() { return this.id };
   public void toggle() {on = !on};
   public boolean on() {return on;};
}
```

```
public class Door {
   private final Light left;
   private final Light right;
   private final Light top;

   public Door(Light l, Light r) {
    left = l;
    right = r;
    top = new Light;
   }

   public synchronized boolean tryOpen() {
       Light l = left;
       Light t = top;
       Light r = right;
       Solution: (below)
             l.toggle();
             r.toggle();
             t.toggle();
             return l.on() && r.on() && t.on();
       Solution: (below)
   }
}
```

Kurze Erklärung *Short explanation*

Solution:

```
public synchronized boolean tryOpen() {
   Light l = left;
   Light t = top;
   Light r = right;

   if (r.getID() > l.getID()) {
      Light tmp = l;
      l = r;
      r = tmp;
   }

   synchronized(l) {
      synchronized(r) {
         synchronized(t) {
             l.toggle();
             r.toggle();
             t.toggle();
             return l.on() && r.on() && t.on();
```

```
                }
            }
        }
}
```

We need to sort the left and the right light to avoid a deadlock. However, the order that we lock the top light is not important, since the top light is owned by a single door.

Mutex Implementation (11 points)

6. Bei den folgenden Multiple-Choice Fragen gibt es einen Punkt, wenn alle korrekten Möglichkeiten angekreuzt wurden. Andernfalls gibt es 0 Punkte.

For the following multiple choice questions, 1 point is provided if and only if all correct answers are marked, otherwise 0 points are given.

(a) Welches dieser Statements ist korrekt im Bezug auf Javas volatile Schlüsselwort. (1)

Which of these statements are correct regarding Java's volatile key word?

- √ **Caching und Reordering ist für die mit volatile markierten Variablen deaktiviert.** — *Caching and reordering is disabled for variables marked volatile.*
- □ Der Wert einer mit volatile gekennzeichneten Variablen wird immer vom Cache gelesen. — *The value of a variable marked volatile is always read from cache.*
- □ Verwendung von mit volatile gekennzeichneten Variablen beschleunigt die Programmausführung grundsätzlich. — *The use of volatile variables speeds up code generally.*
- √ **Volatile kann zum Verhindern von Data Races verwendet werden.** — *Volatile can be used to avoid data races.*
- □ Volatile kann zum Verhindern von Bad Interleavings verwendet werden. — *Volatile can be used to avoid bad interleavings.*

(b) Welche der folgenden Aussagen zu den Mutex Algorithmen ist wahr? (1)

Which of the following statements about different mutex algorithms is true?

- □ Der Filter Lock Algorithmus ist fair. — *The Filter Lock Algorithm is fair.*
- √ **Der Filter Lock basiert auf der Verwendung mehrere Instanzen des Peterson Algorithmus.** — *The Filter Lock is based on using multiple instances of the Peterson algorithm.*
- □ Der Peterson Lock Algorithmus basiert auf der Verwendung mehrerer Instanzen des Filter Locks. — *The Peterson Lock is based on using multiple instances of the Filter Lock algorithm.*
- □ Der Peterson Lock unterliegt Starvation. — *The Peterson Lock suffers from starvation.*

(c) Erklären Sie knapp (Stichworte oder 1-2 Sätze), warum Lock contention problematisch ist. Nennen und erklären Sie kurz eine in der Vorlesung diskutierte Strategie, um damit umzugehen. (2)

Explain in keywords or short sentences (1-2) why lock contention is problematic. Name and briefly explain one strategy, discussed in the lecture, to alleviate the problem(s).

Solution: If the same memory location is used in threads running on multiple cores, the value can not be read from/written to local caches. (Or rather the cache coherency protocol has a lot of work, but cache coherency was not in the lecture.) This results in very poor performance. (1/2 point naming, 1/2 point explanation)
Usual strategy are to implement a back off if the resource can not be acquired. (1 point; 1 point also given if another valid strategy, such as restructuring the program so that less threads access the same resource at once, again 1/2 point for naming and 1/2 point for explanation)

Betrachten Sie folgende (korrekte) Implementation eines Peterson-Locks:

Consider the following (correct) implementation of a Peterson lock:

```
class PetersonLock
{
    AtomicBoolean flag[] = new AtomicBoolean[2];
    volatile int victim;

    // pre: id = 0 or 1
    public void Acquire(int id) {
        flag[id].set(true);
        victim = id;
        while (flag[1-id].get() && victim == id);
    }

    // pre: id = 0 or 1
    public void Release(int id) {
        flag[id].set(false);
    }
}
```

Jemand schlägt vor, folgendermassen drei Peterson Locks hierarchisch für die Implementation eines Locks für Systeme mit vier Prozessen zu verwenden (Prozess ID 0,1,2,3):

Someone suggests to use three Peterson locks hierarchically in order to implement a lock for a system with four processes (having id 0,1,2 and 3) in the following way.

```
class Lock4{
    PetersonLock[] lock = new PetersonLock[3];

    // pre: id = 0, 1, 2 or 3
    public void Acquire(int id){
        lock[0].Acquire(id / 2);
        lock[1+id/2].Acquire(id % 2);
    }

    // pre: id = 0, 1, 2 or 3
    public void Release(int id){
        lock[1+id/2].Release(id % 2);
        lock[0].Release(id / 2);
    }
}
```

(d) Zeigen sie in einem Diagram die Struktur des (inkorrekt) implementierten Locks `Lock4` auf. *Draw a figure sketching the structure of the (incorrectly) implemented lock* `Lock4`. (2)

Solution: A tree with $lock_0$ at the top and two children $lock_1$ and $lock_2$. The top most node is intended to filter the four processes.

(e) Erklären Sie warum das Lock `Lock4` nicht korrekt funktioniert. *Explain why the lock* `Lock4` *cannot not work correctly.* (2)

Solution: The lock cannot work because lock[0] can be driven by more than two processes at a time, the Peterson lock does not allow this.

(f) Zeigen Sie einen Lösungsweg auf, der trotzdem lediglich drei PetersonLocks verwendet. *Explain what to change in order to make it work. Your solution must use the three Peterson locks.* (3)

Solution: We simply turn around the tree in order to first sycnrhonize between processes 0 and 1 or between processes 2 and 3 and then synchronize between process 0/1 and 2/3:

```
class Lock4{
    PetersonLock[] lock = new PetersonLock[3];

    public void Acquire(int id){
        lock[1+id/2].Acquire(id % 2);
        lock[0].Acquire(id / 2);
    }

    public void Release(int id){
        lock[0].Release(id / 2);
        lock[1+id/2].Release(id % 2);
    }
}
```

Lock-free programming (12 points)

7. Diese Frage behandelt die lock-freie Programmierung. *This question covers lock-free programming.*

(a) Beschreiben Sie das ABA Problem und wann es auftritt. *Describe what the ABA problem is and when it occurs.* (3)

Solution:
Lecture?: "The ABA problem occurs when one activity fails to recognise that a single memory location was modified temporarily by another activity and therefore erroneously assumes that the overall state has not been changed."
Wikipedia: In multithreaded computing, the ABA problem occurs during synchronization, when a location is read twice, has the same value for both reads, and value is the same is used to indicate nothing has changed. However, another thread can execute between the two reads and change the value, do other work, then change the value back, thus fooling the first thread into thinking nothing has changed even though the second thread did work that violates that assumption.

(b) Nennen Sie drei Ansätze zur Lösung des ABA Problems. *Name three methods that can solve ABA problem.* (3)

Solution:

- DCAS (double compare and swap)
- Garbage Collection
- Pointer Tagging
- Hazard Pointers
- Transactional memory

Grading: Any three of above sufficient form max three points.

(c) Geben Sie in der Tabelle an, welche Voraussetzung an einen nebenläufigen Algorithmus zu den dargestellten Eigenschaften führt. *Fill in the requirement of a concurrent algorithm such that it provides the properties given in the table below.* (2)

	Non-Blocking	Blocking
Everyone makes progress	**Solution:** Wait-free	**Solution:** Starvation-free
Someone makes progress	**Solution:** Lock-Free	**Solution:** Deadlock-Free

(d) Der folgende Code verwendet Locks für die Implementation eines Algorithmus zur Akkumulation von Integer Einträgen. Der Code darunter soll die gleiche Funktionalität in lock-freier Weise implementieren. Füllen Sie die fehlenden Codestücke entsprechend ein. (4)

The following piece of code uses locks to implement the accumulation of integer array entries. The code below is supposed to implement the same functionality using a lock-free algorithm. Fill in the missing code lines accordingly.

Lock-based Implementation:

```
public class Accumulator{
    private int sum;

    public synchronized int getVal(){  return sum;  }

    public synchronized void accumulate(int [] data){
        for (int index = 0; index<data.length; ++index)
            sum = sum + data[index];
    }
};
```

Lock-Free Implementation:

```
public class Accumulator{
    private  Solution: AtomicInteger  sum;

    public int getVal(){ return  Solution: sum.get()  ;  }

    public void accumulate(int [] data){
         Solution: (below)
    }
};
```

Solution:

```
for (int i = 0; i< data.length; ++i){
    int v = sum.get();
    while (!sum.compareAndSet(v, v+data[i]))
        v = sum.get();
}

// faster and better:
int s = 0;
for (int i = 0; i< data.length; ++i){
    s  += sum.get();
}
v = sum.get();
```

```
    while (!sum.compareAndSet(v, s))
        v = sum.get();
    }
```

Linearizability (7 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();
}
```

Welches der folgenden Szenarien ist linear konsistent? Markiere entweder den Linearisierungspunkt oder erkläre, warum es nicht linear konsistent ist.

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

(a) (1)

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

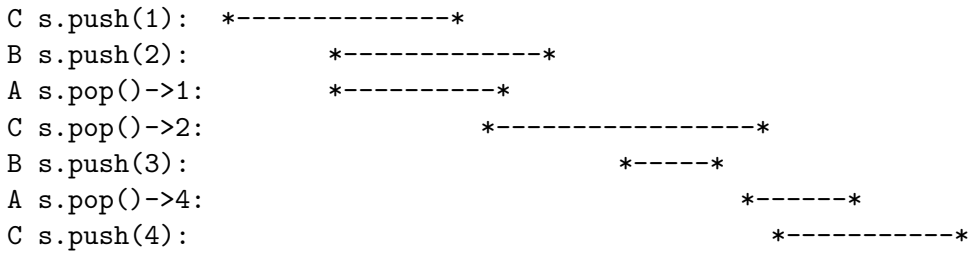
.....

.....

Solution:

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

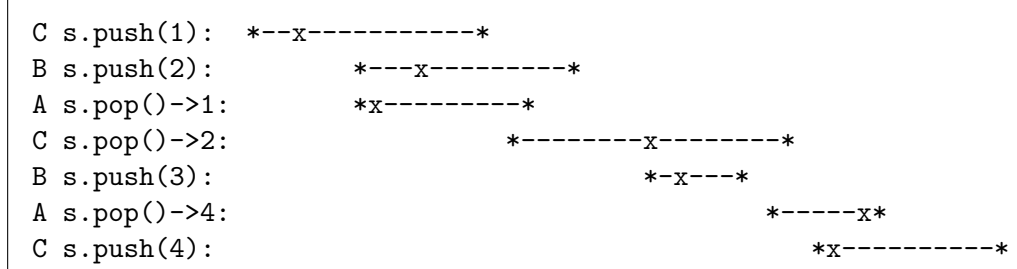
(b) (1)



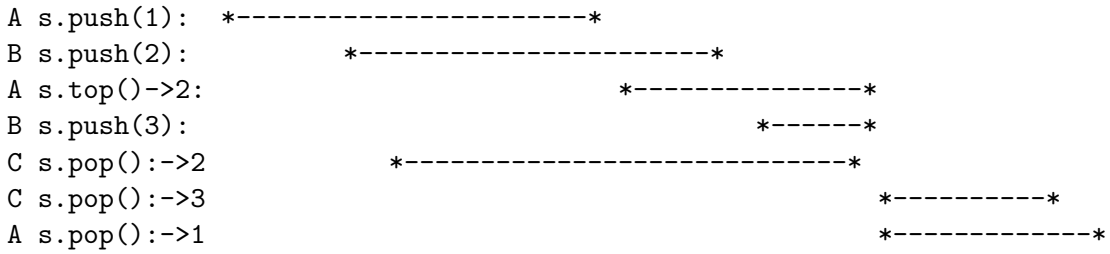
.....

.....

Solution:



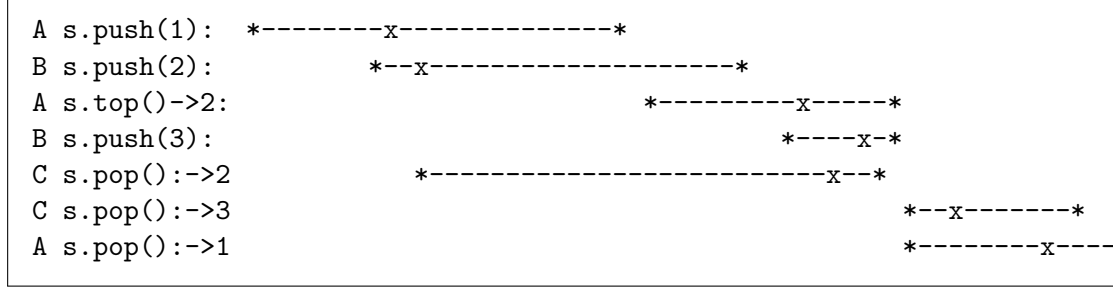
(c) (1)



.....

.....

Solution:



(d) (1)

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

. . . . . .

. . . . . .

Solution:

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

The operation with # cannot be satisfied.

(e) Was ist der Unterschied zwischen Linearisierbarkeit und sequenzieller Konsistenz? Konstruiere ein Beispiel welches sequenziell konsistens aber nicht linearisierbar ist. (3)

What are the differences between linearizability and sequentical consistency? Construct an example that is sequential consistent but not linearizable.

Solution:

Linearizability requires both the order between event across threads and the order between thread local events to be preserved. Sequential consistency only requires thread local event order to be preserved.

(f)

```
A s.push(x):  *------*
B s.push(y):           *------*
A s.pop():->y                  *----*
```

Parallel Algorithms (8 points)

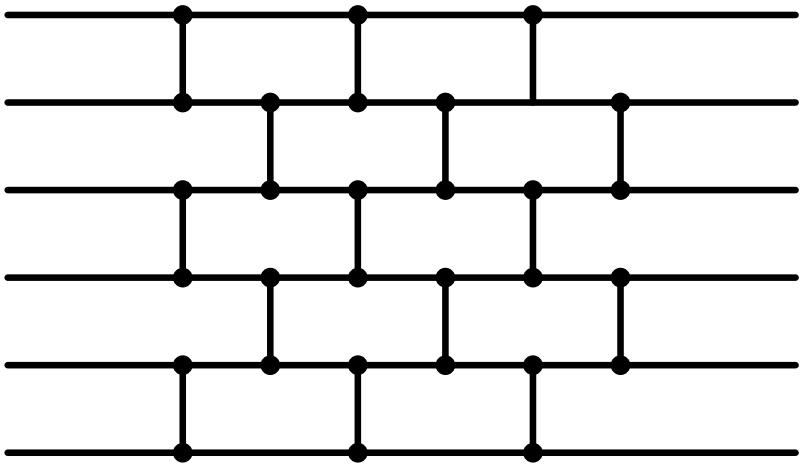
9. Paralleles Sortieren und Sortiernetzwerke *Parallel Sorting and Sorting Networks*

(a) Was sagt das Null-Eins-Prinzip für Komparatornetzwerke aus? *What is the statement of the zero-one-principle for comparator networks?* (4)

Solution: The zero-one-principle says that a comparator network sorts any input sequence consisting of 0s and 1s, then it sorts all input sequences.

(b) Geben Sie eine Eingabefolge bestehend aus 0 und 1 für das folgende Odd-Even-Sortiernetzwerk der Tiefe 6 an, sodass zwei Werte bei einem Komparator der Tiefe 6 vertauscht werden. Schreiben Sie auf alle Leitungen die entsprechenden Werte.
Hinweis: Denken Sie daran, wie wir in den Übungen gezeigt haben, dass ein Odd-Even-Sortiernetzwerk für eine Input-Sequenz der Länge n Tiefe n haben muss, und konstruieren sie eine Input-Sequenz, bei der der "schlimmste Fall" auftritt.

Consider the following odd-even sorting network of depth 6. Provide an input sequence of 0 and 1 such that there is a swap of two values at a comparator of depth 6. Write all values carried by the wires next to them.
Hint: Recall how we showed in the exercises that an odd-even sorting network needs depth n for inputs of size n and provide an input triggering the "worst case". (4)



Solution: Basically any sequence that consists of an even number of 1s followed by 0s only.