Vor und Nachname (Druckbuchstaben): _____

Legi Nummer: _____

252-0024-00L
Parallele Programmierung
ETH/CS: FS 2015
Basisprüfung
Montag, 1.2.2016
120 Minuten

Diese Prüfung enthält 21 Seiten (inklusive diesem Deckblatt) und 11 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	2	
3	6	
4	7	
5	4	
6	6	
7	10	
8	13	
9	17	
10	13	
11	12	
Total:	94	

Java Sequential Programming (12 points)

1. Die n-te Fibonacci Zahl liesse sich zum Beispiel mit folgender Implementierung berechnen.

   The following code shows one possible implementation to compute the n^{th} Fibonacci number

```
// pre: n >= 0
// post: return the n-th fibonacci number
public static int fibonacci(int n){
    if (n==0)
        return 0;
    else if (n==1)
        return 1;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}
```

   (a) Vervollständigen Sie untenstehende Methode `fibonacciCalls`, analog zu `fibonacci(n)`, so dass Ihr Code die Anzahl der rekursiven Aufrufe zurück gibt. Für diese Teilaufgabe muss ihre Implementierung die n-te Fibonacci-Zahl nicht zwingend berechnen. (2)

   Analogously to `fibonacci(n)`, complete the following method such that it computes the number of recursive calls when called `fibonacciCalls(n)`. Your implementation does not necessarily need to compute the actual Fibonacci number.

```
// post: returns the number of calls
public static int fibonacciCalls(int n)
{
    if (n<2)

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

    else

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

   (b) Vervollständigen Sie folgende Methode so, dass sie die grösste Rekursionstiefe zurückgibt, wenn diese mit `fibonacciSpan(n)` aufgerufen wird. Für diese Teilaufgabe muss ihre Implementierung die n-te Fibonacci-Zahl nicht zwingend berechnen. (2)

   Complete the following function such that it returns the largest recursion depth when called as `fibonacciSpan(n)`. Your implementation does not necessarily need to compute the actual Fibonacci number.

```
// post: returns the max recursion depth
public static int fibonacciSpan(int n)
{
    if (n<2)

        ..........................................
```

```
        else

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

2. Lesen Sie den folgenden Quellcode gründlich durch *Carefully read the following code* (2)

```
class A {
}

class B extends A {
}

class C extends B {
}

class D extends B {
}

class E extends A {
}

public class Main {
    public static void main(String[] args) {

        // are the following allowed ?
        D w = new C(); //  Ja/Yes  Nein/No
        B x = new D(); //  Ja/Yes  Nein/No
        E y = new A(); //  Ja/Yes  Nein/No
        A z = new D(); //  Ja/Yes  Nein/No

    }
}
```

Geben Sie für den obigen Programmcode an, ob die Zuweisungen in der main Funktion korrekt (Ja/Yes) oder ungültig sind (Nein/No).

Given the above code, decide if the assignment calls in the main method are correct (Ja/Yes) or invalid (Nein/No).

3. Gegeben sei eine Folge $a = \{a_0, \cdots, a_{N-1}\}$ von ganzen Zahlen. Vervollständigen Sie folgende Methode so, dass sie die Länge $n \geq 0$ der längsten aufsteigenden Teilfolge $\{a_k, \ldots, a_{k+n-1}\}$ mit $a_k \leq a_{k+1} \leq \cdots \leq a_{k+n-1}$ zurückgibt. (Teil-)Folgen der Länge 1 gelten als aufsteigend. Die leere Folge hat Länge 0.
Beispiel: Die längste aufsteigende Teilfolge von $\{1, 2, 3, \underline{1, 2, 3, 4}, 1, 2, -10\}$ hat die Länge 4.

Let $a = \{a_0, \ldots, a_{N-1}\}$ be a sequence of integers. Complete below method to return the length $n \geq 0$ of the longest ascending subsequence $\{a_k, \ldots, a_{k+n-1}\}$ with $a_k \leq a_{k+1} \leq \cdots \leq a_{k+n-1}$. A (sub-)sequence of length 1 is ascending. The empty sequence is of length 0.
Example: The longest increasing subsequence of $\{1, 2, 3, \underline{1, 2, 3, 4}, 1, 2, -10\}$ has a length of 4.

(6)

```
// pre: array a, potentially null or of zero length
// post: return length of longest increasing subsequence of a
//       return 0, if a == null or a has no elements
public static int longestIncreasingSequence (int[] a){

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

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

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

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

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

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

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

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

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

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

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

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

Speedup, Amdahl, Gustafson (7 points)

4. Beantworten Sie die folgenden Fragen zum Thema Speedup sowie Amdahlsches und Gustafsonsches Gesetz:
Answer the following questions about speedup, Amdahl's and Gustafson's laws:

   (a) Schreiben Sie die Formel für den Speedup nach dem **Amdahlschen** Gesetz auf und erläutern Sie kurz alle Terme. (3)
*Write down the formula for speedup according to **Amdahl**'s law and briefly describe its terms.*

   .....

   .....

   .....

   .....

   (b) Die Analyse eines Programms hat ergeben, dass es einen Speedup von 7 (skaliert) hat, wenn es auf 16 Prozessoren läuft. (2)
The analysis of a program has shown a speedup of 7 (scaled) when running on 16 cores
Was ist der serielle Anteil nach dem **Gustafsonschen** Gesetz?
*What is the serial fraction according to **Gustafson**'s law?*

   .....

   .....

   .....

   .....

   .....

   (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?*

   .....

   .....

   .....

   .....

   .....

Task Graphs und Pipelining (10 points)

5. Abbildung 1 zeigt den Task Graphen eines Algorithmus. Die Zahl an jedem Knoten bezeichnet die benötigte Ausführungszeit für den jeweiligen Berechnungsschritt.

Figure 1 shows the task graph for an algorithm. The number in each node denotes the execution time per task. (4)

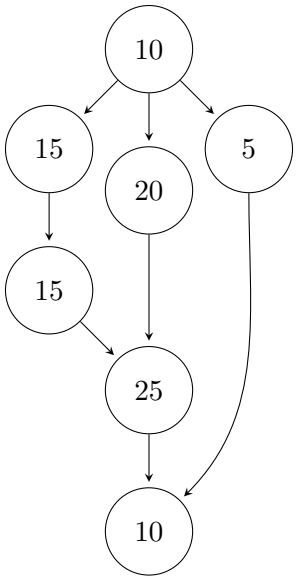

Abbildung 1: Task graph

Wie gross ist der maximale Speedup der mit Parallelisierung erreicht werden kann, verglichen mit sequentieller Durchführung bei einer einzigen Ausführung des Algorithmus?

What is the maximum overall speedup that can be achieved by parallelism when the algorithm runs once compared to sequential execution?

.....

.....

.....

.....

.....

6. Für jedes Element x eines sehr grossen Arrays A soll eine Funktion $F(x) = f_3(f_2(f_1(f_0(x))))$ berechnet werden. Die Ausführungszeiten $T(f_i)$ sind: $T(f_3) = 50\mu s$, $T(f_2) = 30\mu s$, $T(f_1) = 40\mu s$ and $T(f_0) = 50\mu s$. Gehen Sie im weiteren davon aus, dass für jede Stufe der Pipeline stets eine separate Ausführungseinheit zur Verfügung steht Ein System bestehend aus einer Pipeline zur Berechnung von F ist dargestellt in Abbildung 2.

For each element x of a very large array A, a function $F(x) = f_3(f_2(f_1(f_0(x))))$ shall be computed. The execution times $T(f_i)$ are given as: $T(f_3) = 50\mu s$, $T(f_2) = 30\mu s$, $T(f_1) = 40\mu s$ and $T(f_0) = 50\mu s$. In the following you can assume that for each stage of the pipeline a separate execution unit is available. A system consisting of a pipeline to compute F is depicted in Figure 2.

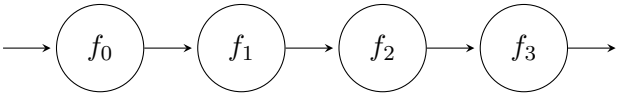

Abbildung 2: Pipeline

(a) Geben Sie die Latenz der Pipeline an. *Provide the latency of the pipeline.* (2)

. . . . . .

. . . . . .

(b) Wie gross ist der Durchsatz dieser Pipeline (in Elementen / Sekunde)? *What is the throughput of this pipeline (in elements / second)?* (2)

. . . . . .

. . . . . .

(c) Wie gross ist der Speedup, wenn Sie den Durchsatz mit dem einer sequentiellen Ausführung ohne Pipeline vergleichen? *What speedup does this yield, when you compare the throughput with that of a non-pipelined, sequential execution?* (2)

. . . . . .

. . . . . .

Fork/Join Framework (10 points)

7. Die Klasse `ForkJoinSearch` benutzt das Java Fork/Join Framework um die Suche von Integer-Elementen in Arrays zu beschleunigen: Das Array wird aufgeteilt, und parallel von unterschiedlichen Tasks durchsucht. Ihre Aufgabe ist es die `compute` Methode zu implementieren, welche `true` zurückgibt, falls der `searchKey` in dem `arrayToSearch` gefunden wird und `false` falls nicht. Der `SEQUENTIAL_THRESHOLD` Wert definiert die maximale Anzahl an Array-Elementen, die von einem Task sequentiell überprüft werden.

The class `ForkJoinSearch` uses the Java Fork/Join framework to speed-up the search of integer elements in a large array by dividing the array and then search it in parallel by different tasks. Your objective is to implement the `compute` method to return `true` if the `searchKey` was found in `arrayToSearch` or `false` otherwise. The `SEQUENTIAL_THRESHOLD` defines the maximum amount of elements that are checked by a single Fork/Join task sequentially. (10)

Das Code-Template befindet sich auf der nächsten Seite!

The Code Template can be found on the next page!

```
public class ForkJoinSearch extends RecursiveTask<Boolean> {

    private static int SEQUENTIAL_THRESHOLD = 10000;
    int from;
    int to;
    int searchKey;
    int[] arrayToSearch;

    public ForkJoinSearch(int[] arrayToSearch, int searchKey,
       int from, int to) {
        this.from = from;
        this.to = to;
        this.searchKey = searchKey;
        this.arrayToSearch = arrayToSearch;
    }

    protected Boolean compute() {
        if ((to - from) <= SEQUENTIAL_THRESHOLD) {

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

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

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

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

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

            ..................................................
        }
        else {

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

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

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

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

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

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

Synchronization (13 points)

8. Ein java.util.concurrent.locks.Condition Object exportiert die folgenden Methoden:

   A java.util.concurrent.locks.Condition object exports the following methods:

   - await()
   - signal()
   - signalAll()

   - *await()*
   - *signal()*
   - *signalAll()*

   (a) Erklären Sie den Effekt den diese drei Methoden haben.

   Explain the effect of these three methods. (3)

   .....

   .....

   .....

   .....

   .....

   (b) Was passiert mit Locks im Besitz eines Callers wenn die await() Methode einer Condition aufgerufen wird?

   What happens with Locks owned by the caller upon a call to await() on a condition? (2)

   .....

   .....

   .....

   .....

   .....

   (c) Ein Sparbuch Objekt hat immer ein positives Guthaben (balance) und stellt die Methoden `deposit(k)` und `withdraw(k)` bereit. `deposit(k)` addiert k zum Guthaben und `withdraw(k)` substrahiert k wenn das Guthaben mindestens k beträgt und blockt sonst bis das Guthaben wieder grösser oder gleich gross wie k ist.
   Vervollständigen Sie die SavingsAccount Klasse auf der nächsten Seite entsprechend.

   A savings account object holds a non-negative balance, and provides methods `deposit(k)` *and* `withdraw(k)`*, where* `deposit(k)` *adds k to the balance and* `withdraw(k)` *subtracts k, if the balance is at least k, and otherwise blocks until the balance becomes k or greater.*

   Complement the SavingsAccount class on the next page accordingly. (8)

```
public class SavingsAccount {
    protected int balance;
    final Lock lock = new ReentrantLock();

    final Condition ......... = ........................

    public void deposit(int k) {

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

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

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

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

            balance += k;

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

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

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

        ..............................
    }

    public void withdraw(int k) {

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

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

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

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

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

            balance -= k;

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

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

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

} }
```

Lock-free programming (17 points)

9. Diese Frage behandelt die lock-freie Programmierung.

This question covers lock-free programming.

(a) Nennen Sie drei Probleme die in parallelen Programmen die locks verwenden auftreten können. *Name three problems that may occur in parallel programs using locks.* (3)

.....

.....

.....

.....

(b) Was ist wait-freedom und lock-freedom und was ist der Zusammenhang zwischen ihnen. *What is wait-freedom and lock-freedom? How are they related?* (3)

.....

.....

.....

.....

(c) Welche Vorraussetzungen muss ein Program erfüllen, um die aufgelisteten Eigenschaften zu erzielen? *What requirements have to be fulfilled to satisfy the properties in below table?* (2)

	Non-Blocking	Blocking
Everyone makes progress	.....	.....
Someone makes progress	.....	.....

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

.....

.....

.....

.....

(e) Der folgende Code verwendet Locks. *The following code uses locks.* (6)

```
public static class Node {
    public final Long value;
    public Node next;

    public Node(long val){
        this.value = val;
    }
}

public class Stack {
    Node top = NULL;
    public synchronized void push(long value) {
        Node n = new Node(value);
        node.next = top;
        top=node;
    }
    public synchronized long pop() {
        if (top == NULL) {
            throw new StackEmtpyException();
        }
        Node n = top;
        top = top.next;
        return n.value;
    }
}
```

Implementieren Sie die `Stack`-Klasse mit Hilfe der Methode

Implement the `Stack` class using the method

```
boolean compareAndSet(V expect, V update)
```

der Klasse `AtomicReference`

of class `AtomicReference`

```
public class ConcurrentStack {

  AtomicReference<.......> ........ = new
     AtomicReference<..................>();

  public void push(Long item) {

      Node head, newNode;
      Node newi = new Node(item);

      do {

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

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

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

      } while (.....................................);
  }

  public Long pop() {
      Node head, next;
      do {

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

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

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

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

       } while (.....................................);

       return head.item;
  }
}
```

OpenMP & Data Parallelism (13 points)

10. Theoriefragen.
Für Multiple-choice Aufgaben: Eine korrekte Antwort gibt 1 Punkt, eine inkorrekte -1 Punkt. Minimale Punktzahl sind 0 Punkte.

Theory questions.
For multiple-choice questions: For a correct answer you get 1 point, for a wrong one -1 point. Minimum points are 0.

(a) Welche der folgenden Aussagen über OpenMP sind wahr? *Which of the following statements about OpenMP are true?* (3)

- ☐ OpenMP is eine Paralelle Laufzeitumgebung — *OpenMP is a parallel programming runtime.*
- ☐ OpenMP ist eine Programmiersprache — *OpenMP is a programming language*
- ☐ OpenMP ist eine Spracherweiterung für mehrere Programmiersprachen — *OpenMP is a language extension for multiple programming languages*

(b) In OpenMP kann die `schedule()`-Klausel verwendet werden, um zu definieren wie und wann die Arbeit aufgeteilt werden soll. *In OpenMP you can use `schedule()` clauses to specify how and when work is split up.*

i. Welche der folgenden Aussagen zu `static` Scheduling sind wahr? *Which of the following statements about `static` scheduling are true?* (3)

- ☐ Die Arbeitsaufteilung kann vom Programmierer vorhergesagt werden — *The schedule is predictable by programmer*
- ☐ Die Aufteilung wird während dem Kompilieren vorgenommen — *Scheduling is done at compile time*
- ☐ `static` Scheduling benötigt viel Arbeit zur Laufzeit — *`static` scheduling needs a lot of work at runtime*

ii. Geben Sie für die zwei folgenden Aussagen an welche zu `static` Scheduling und welche zu `dynamic` Scheduling gehört *Indicate for the two statements below which belongs to `static` and which to `dynamic` scheduling* (1)

1. Jeder Thread erhält Iterationen zugeteilt. — *Deal out iterations to each thread.*
2. Jeder Thread nimmt Iterationen von einer Queue bis alle Iterationen abgearbeitet sind. — *Each thread grabs iterations of a queue until all iterations are handled.*

_____ `dynamic` _____ *`static`*

(c) Finde und beschreibe kurz eine Lösung zum **Korrigieren der Fehler** in den folgenden OpenMP Programm-Stücken

Find and give a short explanation on how to **correct the bugs** *in the following OpenMP program snippets*

i. Das folgende Code-Stück sollte $\sum_{i=0}^{N} i$ berechnen

The following snippet should compute $\sum_{i=0}^{N} i$ (2)

```
int sum = 0;
#pragma omp parallel for
for (int i = 0; i < N; i++) {
  sum += i;
}
// check that we correctly summed up all elements
assert(sum == (N+1)*N / 2);
```

.....

.....

ii. Im folgenden Code-Stück, verwendet `computation_B()` alle Teilresultate von `computation_A()`. Deshalb müssen alle Instanzen von `computation_A()` fertig sein, bevor `computation_B()` ausgeführt werden kann.

In the following snippet, `computation_B()` *depends on all the partial results of* `computation_A()`*. Therefore all instances of* `computation_A()` *need to be finished before any instance of* `computation_B()` *can be executed.* (2)

```
// Assume we have declared sufficiently large arrays A
   and B
#pragma omp parallel
{
  int id = omp_get_thread_num();
  A[id] = computation_A(id);
  // computation_B() depends on the whole array A.
  B[id] = computation_B(id);
}
```

.....

.....

iii. Der folgende Code soll `something(i)` für $0 \leq i < N$ auf den verfügbaren OpenMP threads ausführen. Sie können annehmen, dass N gegeben ist und dass die Tasks `something(i)` hinreichend unabhängig sind.

The following code should provide an execution of `something(i)` *for* $0 \leq i < N$ *on the available OpenMP threads. We assume that* N *is given and that the tasks for each* i *are sufficiently independent.* (2)

```
int Nthreads = omp_get_num_threads();
#pragma omp parallel
{
        int id = omp_get_thread_num();
        int il = id * N / Nthreads;
        int ir = (id+1) * N / Nthreads;
        for (int i = il; i < ir; ++i)
                something(i);
}
```

.....

.....

OpenCL (12 points)

11. Sie wollen ein verrauschtes Sprachsignal entzerren. Dies kann man mithilfe eines Gauss-Filters erreichen, indem man das usrprüngliche Signal mit dem Filter faltet (als Beispiel siehe Abb. 3). Die mathematische Operation eines Gauss-Filter ist wie folgt definiert:

You want to reduce the noise in a voice signal. A Gauss filter applied to the original signal does this (see Abb. 3 for an example). The mathematical operation of a Gauss filter is defined as:

$$G = \left[\tfrac{1}{16} \quad \tfrac{1}{4} \quad \tfrac{1}{2} \quad \tfrac{1}{1} \quad \tfrac{1}{2} \quad \tfrac{1}{4} \quad \tfrac{1}{16}\right]$$

$$\text{FilteredSignal}[x] = \frac{1}{Z}\sum_{i=0}^{6} \text{InputSignal}[x+i-3] \cdot G[i]$$

$$Z = \sum_{i=0}^{6} G[i]$$

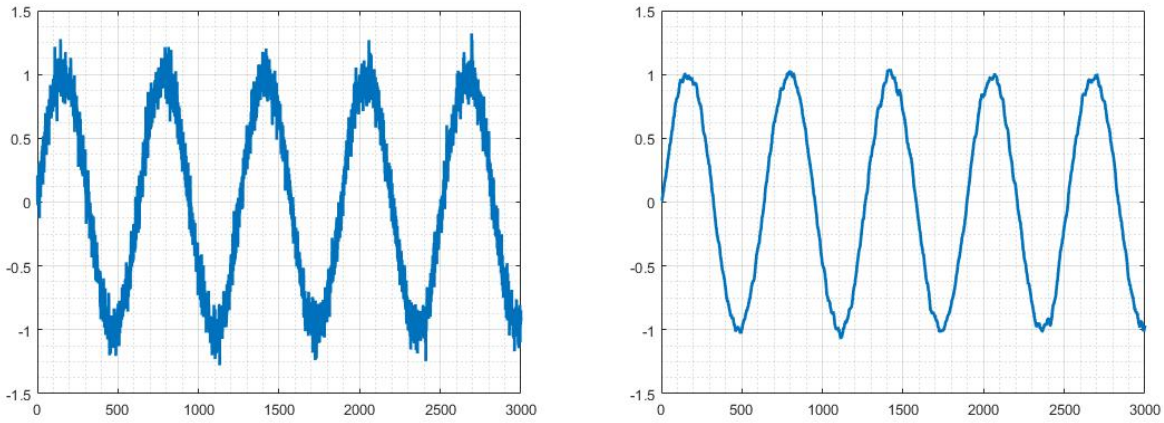

Abbildung 3: Input signal on the left and the filtered signal on the right.

(a) Berechnen sie die Normalisierungskonstante Z. *Compute the normalization constant Z.* (1)

. . . . . .

. . . . . .

(b) Sie wollen nun genau 10 Werte des Input Signals verarbeiten. Wieviele Operationen (Multiplikation, Divison, Addition, Subtraktion) braucht eine sequentielle Implementierung (Sie können Probleme am Rand des Intervals ignorieren)? *You want to filter exactly 10 values of the input signal. How many operations (multiplication, division, addition, subtraction) do you need with a sequential filter implementation (you can ignore boundary and windowing issues)?* (1)

.....

.....

(c) Geben Sie den Wertebereich des Indexes x mit welchem das input signal gefiltert werden kann. Benützen Sie dazu die Variable `SignalLength`, welche die Gesamtlänge des input signals angibt. *Compute the values for the index x where the input signal can be filtered. Use the variable `SignalLength` which gives the total length of the input signal.* (2)

.....

.....

(d) Implementieren Sie den Filter als OpenCL Kernel in dem folgendem Template. Nehmen Sie an, dass der aktuelle Index x als work-item ID in Dimension 0 gespeichert wird (i.e., `int x = get_global_id(0);`).

Implement the filter in OpenCL using the provided code skeleton. You can assume that the current index x is stored as a work-item ID in dimension 0 (i.e., `int x = get_global_id(0);`).

(8)

```
// pre: InputSignal: An array containing the original signal
// pre: SignalLength: The length of the InputSignal
// post: FilteredSignal: An array that stores the result
// after applying the Gauss filter or 0 if the filter
// could not be applied at this index.
__kernel void filter (
    __global float *InputSignal ,
    __global float *FilteredSignal,
    __constant int SignalLength)
{
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....