Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Self Assessment II

Informatik I (D-BAUG)

Felix Friedrich, Hermann Lehner

ETH Zürich, 11.2017.

Name, Vorname:	. . . . .
Legi-Nummer:	. . . . .

Diese **Selbsteinschätzung** dient Ihrer und unserer Orientierung. Sie wird eingesammelt, korrigiert und vertraulich behandelt. Sie hat aber keinen Einfluss auf eine spätere Leistungsbewertung. **Sie haben 20 Minuten Zeit.**

Das folgende Kleingedruckte finden Sie auch auf einer "scharfen" Prüfung.

Allgemeine Richtlinien:

1. Dauer der Prüfung: 20 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für gesprochene Sprachen). 4 A4 Seiten handgeschrieben oder $\geq$ 11pt Schriftgrösse.
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen sind deutlich durchzustreichen! Korrekturen bei Multiple-Choice Aufgaben bitte unmissverständlich anbringen!
5. Es gibt keine Negativpunkte für falsche Antworten.
6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.
7. 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.
8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

1. *Exam duration: 20 minutes.*
2. *Permitted examination aids: dictionary (for spoken languages). 4 A4 pages hand written or $\geq$ 11pt font size*
3. *Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.*
4. *Solutions must be written directly onto the exam sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Provide corrections to answers of multiple choice questions without any ambiguity!*
5. *There are no negative points for false answers.*
6. *If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.*
7. *We collect the exams at the end. Important: you must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.*
8. *If you need to go to the toilet, raise your hand and wait for a supervisor.*
9. *We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.*

Question:	1	2	3	4	Total
Points:	6	4	6	4	20
Score:					

Aufgabe 1: Code Lesen (6P)

Beschreiben Sie in den folgenden Boxen jeweils in Worten, was die darunter stehende Methode macht. Beschreiben Sie nicht die einzelnen Schritte sondern die Bedeutung.

Provide in the following boxes what the methodes are doing. Do not describe the single steps but rather the meaning.

/2P (a)

```
// pre: non-null array a
static boolean p(int[] a, int b){
  int r = a.length;
  while (r > 0){
      if (a[--r] == b) {return true;}
  }
  return false;
}
```

/2P (b)

```
  // pre: n >= 0
  static int q(int n){
      int r = 0;
      for (int i = 1; i<=n; ++i){
          if (n % i == 0){ r += i; }
      }
      return r;
  }
```

/2P (c)

```
// pre: a,b > 0
static int r(int a, int b){
  if (b == 0){
      return 1;
  }
  return a * r(a,b-1);
}
```

Aufgabe 2: Code Schreiben (4P)

/4P

Vervollständigen Sie folgende Methode `minDist` so, dass Sie die minimale absolute Differenz $|x - y|$ zweier unterschiedlicher Werte x und y aus dem Array a zurückgibt.

Complement the following method `minDist` *such that it returns the minimal absolute difference* $|x - y|$ *of two different points* x *and* y *from the array* a.

```
// pre: a != null, a.length >= 2
// post: return minimum distance Math.abs(a[i]-a[j]) of all pairs i != j
public static int minDist(int a[]){
    int dist = Math.abs(a[0]-a[1]);
    for (int i = 0; i<a.length; ++i){
        for (                                                    ){
            int d = Math.abs(a[i]-a[j]);
                                                                 
                                                                 
                                                                 
                                                                 
            }
        }
    }
    return dist;
}
```

Aufgabe 3: Listen (6P)

Sei für diese Aufgabe folgende Listknoten-Klasse gegeben:

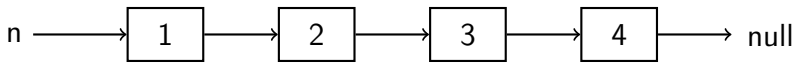
Let for this task be the following list-node class be given:

```
class ListNode{
    ListNode next;
    int value;

    ListNode (int v, ListNode n){
        value = v; next = n;
    }
}
```

Es sei eine verkettete Liste von Knoten gegeben. Das Ende der Liste ist durch einen Null-Knoten gegeben. Beispiel:

Assume a linked list of node elements. The end of the linked ist is provided by a null-node. Example:



Die folgende rekursive Funktion gibt die Werte der Knoten aus, startend beim gegebenen Knoten n bis zum Ende der verketteten Liste.

The following recursive function outputs the node's values, starting from the given node element n and traversing to the end of the linked list.

```
void output(ListNode n){
  if (n != null){
    Out.println(n.value);
    output(n.next);
  }
}
```

Ausgabe *Example Output:*
Beispiel
1
2
3
4

/2P (a) Vervollständigen Sie den folgenden Code so, dass die Werte in umgekehrter Reihenfolge ausgegeben werden. Verwenden Sie Rekursion!

Complement the following Code such that the values in the list are output in reversed order. Use recursion!

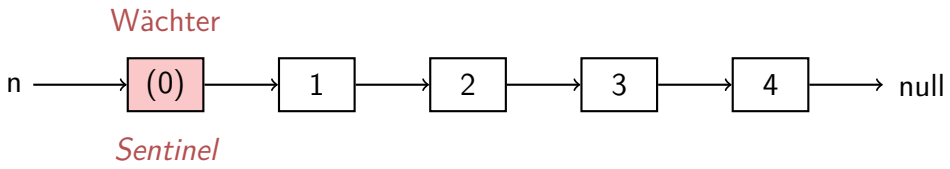
```
void outputR(ListNode n){
    
}
```

Ausgabe *Example*
Beispiel *Output:*
4
3
2
1

(b) Auf der nächsten Seite sehen Sie eine (korrekte) Implementation einer Listenklasse, mit Operationen out, empty und append. Jemand behauptet, dass die Implementation mancher Operationen einfacher wird, wenn die Liste am Anfang immer mit einem nicht-leeren (Wächter-) Element beginnt und schlägt daher die danach folgende SentinelListe vor. Vervollständigen Sie also die Methoden der SentinelListe so, dass die beiden Klassen List und SentinelList dieselbe Semantik aufweisen.
Die Liste, welche auf der linken Seite dargestellt ist, sähe mit Wächterelement wie folgt aus.

On the following page you see a (correct) implementation of a list class with operations out, empty and append. Someone claims that the implementation of some operations becomes simpler when the list always starts with a non-empty (sentinel) element. Therefore he proposes the following SentinelList. Complement the methods of the SentinelList such that both classes List and SentinelList provide the same semantics.
The list displayed on the left hand side would look like the following with a sentinel.

/4P



Vervollständigen Sie die Methoden der SentinelListe auf der rechten Seite so, dass die beiden Klassen List und SentinelList dieselbe Semantik aufweisen. Das heist, der von Aussen beobachtbare Effekt des ausgeführten Codes ist identisch.

Complement the methods of the SentinelList on the right page such that both classes List and SentinelList provide the same semantics. That is, the effect observable from outside is identical.

```
class List{
  ListNode head;

  List(){ head = null; }

  // post: outputs the list content (first in, first out)
  public void out(){
    ListNode n = head;
    while (n != null){
        Out.print(n.value + " ");
        n = n.next;
    }
  }

  // post: returns if the list is empty
  public boolean empty(){
    return head == null;
  }

  // post: appends a new node with value at the end of the list
  public void append(int value){
    if (head == null){
      head = new ListNode(value, null);
      return;
    }
    ListNode p = head;
    while (p.next != null){
      p = p.next;
    }
    p.next = new ListNode(value,null);
  }
}
```

```
class SentinelList{
  ListNode head;

  SentinelList(){
    head = new ListNode(0, null); // head = sentinel
  }

  // post: outputs the list content (first in, first out)
  public void out(){
     ListNode n = head.next;
     while (n != null){
         Out.print(n.value + " ");
         n = n.next;
     }
  }

  // post: returns if the list is empty
  // (a list with sentinel is called empty when it contains no element but the sent
  public boolean empty(){
     
  }

  // post: appends a new node with value at the end of the list
  public void append(int value){
     
  }
}
```

/4P

Aufgabe 4: Fehlersuche (4P)

Im nachfolgenden Codebeispiel steckt ein Fehler. Das Beispiel kompiliert vollständig. Die vorhandenen Fehler werden vom Compiler also nicht bemerkt. Erklären jeweils kurz den Fehler und geben Sie ganz kurz eine mögliche Behebung in den Boxen unter den Codestücken an. In dieser Aufgabe betrachten wir Code dann als fehlerfrei, wenn er korrekt terminiert und unter der Vorbedingung die Nachbedingung erfüllt.

The following code example has a bug. It compiles successfully. That means the bug is not identified by the compiler. Please explain the bug and a possible resolution in the boxes below the code pieces. Within this task we consider code bug-free if it terminates correctly and if it fulfills the post-condition under the pre-condition.

```
// swap integers a and b
static void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
}

// pre: unsorted array a != null
// post: a sorted
static void bubbleSort(int a[]){
    for (int i = 0; i<a.length; ++i)
        for (int j = i; j<a.length; ++j)
            if (a[i] > a[j]) swap(a[i],a[j]);
}
```

Problem / *Problem*:

Behebung / *Resolution*: