RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe A

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für Binärbäume (Sie dürfen annehmen, daß die hier betrachteten Bäume keine Zyklen enthalten; d. h. irgendwann sind die *left* und *right* Pointer *null*, um leere Teilbäume zu repräsentieren):

```
public class Node {
  public int value;
  public Node left;
  public Node right;
}
```

Implementieren Sie die folgende Methode *product* in der Klasse *Node*, welche das Produkt der *value* Attribute aller Elemente des Binärbaums zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

```
public int product() {



}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe B

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für Binärbäume (Sie dürfen annehmen, daß die hier betrachteten Bäume keine Zyklen enthalten; d. h. irgendwann sind die *left* und *right* Pointer *null*, um leere Teilbäume zu repräsentieren):

```
public class Node {
  public int value;
  public Node left;
  public Node right;
}
```

Implementieren Sie die folgende Methode *sum* in der Klasse *Node*, welche die Summe der *value* Attribute aller Elemente des Binärbaums zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

```
public int sum() {




}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe C

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für einfach verkettete zyklische Listen (Sie dürfen also annehmen, daß die hier betrachteten Listen immer zyklisch sind; d. h. der *next* Pointer ist niemals *null*):

```
public class Node {
  public int value;
  public Node next;
}
```

Implementieren Sie die folgende Methode *sum* in der Klasse *Node*, welche die Summe der *value* Attribute aller Elemente der zyklischen Liste zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

```
public int sum() {




}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe D

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für einfach verkettete zyklische Listen (Sie dürfen also annehmen, daß die hier betrachteten Listen immer zyklisch sind; d. h. der *next* Pointer ist niemals *null*):

```
public class Node {
  public int value;
  public Node next;
}
```

Implementieren Sie die folgende Methode *contains* in der Klasse *Node*, welche *true* genau dann zurückgeben soll, wenn es in der zyklischen Liste ein Element mit dem übergebenen Wert als *value* Attribut gibt. Ansonsten soll die Methode *false* zurückgeben. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

```
public boolean contains(int v) {




}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe E

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für einfach verkettete zyklische Listen (Sie dürfen also annehmen, daß die hier betrachteten Listen immer zyklisch sind; d. h. der *next* Pointer ist niemals *null*):

```
public class Node {
  public int value;
  public Node next;
}
```

Implementieren Sie die folgende Methode *product* in der Klasse *Node*, welche das Produkt der *value* Attribute aller Elemente der zyklischen Liste zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

```
public int product(){




}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe A

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für Binärbäume (Sie dürfen annehmen, daß die hier betrachteten Bäume keine Zyklen enthalten; d. h. irgendwann sind die *left* und *right* Pointer *null*, um leere Teilbäume zu repräsentieren):

```
public class Node {
  public int value;
  public Node left;
  public Node right;
}
```

Implementieren Sie die folgende Methode *product* in der Klasse *Node*, welche das Produkt der *value* Attribute aller Elemente des Binärbaums zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

Lösungsvorschlag:

```
public int product() {
  int res = this.value;
  if (this.left != null) {
    res *= this.left.product();
  }
  if (this.right != null) {
    res *= this.right.product();
  }
  return res;
}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe B

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für Binärbäume (Sie dürfen annehmen, daß die hier betrachteten Bäume keine Zyklen enthalten; d. h. irgendwann sind die *left* und *right* Pointer *null*, um leere Teilbäume zu repräsentieren):

```
public class Node {
  public int value;
  public Node left;
  public Node right;
}
```

Implementieren Sie die folgende Methode *sum* in der Klasse *Node*, welche die Summe der *value* Attribute aller Elemente des Binärbaums zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

Lösungsvorschlag:

```
public int sum() {
  int res = this.value;
  if (this.left != null) {
    res += this.left.sum();
  }
  if (this.right != null) {
    res += this.right.sum();
  }
  return res;
}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe C

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für einfach verkettete zyklische Listen (Sie dürfen also annehmen, daß die hier betrachteten Listen immer zyklisch sind; d. h. der *next* Pointer ist niemals *null*):

```
public class Node {
  public int value;
  public Node next;
}
```

Implementieren Sie die folgende Methode *sum* in der Klasse *Node*, welche die Summe der *value* Attribute aller Elemente der zyklischen Liste zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

Lösungsvorschlag:

```
public int sum() {
  return this.next.sum(this);
}

public int sum(Node origin) {
  if (this == origin) {
    return this.value;
  } else {
    return this.value + this.next.sum(origin);
  }
}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe D

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für einfach verkettete zyklische Listen (Sie dürfen also annehmen, daß die hier betrachteten Listen immer zyklisch sind; d. h. der *next* Pointer ist niemals *null*):

```
public class Node {
  public int value;
  public Node next;
}
```

Implementieren Sie die folgende Methode *contains* in der Klasse *Node*, welche *true* genau dann zurückgeben soll, wenn es in der zyklischen Liste ein Element mit dem übergebenen Wert als *value* Attribut gibt. Ansonsten soll die Methode *false* zurückgeben. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

Lösungsvorschlag:

```
public boolean contains(int v) {
  return this.next.contains(this, v);
}

public boolean contains(Node origin, int v) {
  if (this == origin) {
    return this.value == v;
  } else if (this.value == v) {
    return true;
  } else {
    return this.next.contains(origin, v);
  }
}
```

RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Blatt 6 Gruppe E

Minipräsenzübung zur Vorlesung Programmierung

Aufgabe 6 (10 Punkte)

Betrachten Sie die folgende Datenstruktur für einfach verkettete zyklische Listen (Sie dürfen also annehmen, daß die hier betrachteten Listen immer zyklisch sind; d. h. der *next* Pointer ist niemals *null*):

```
public class Node {
  public int value;
  public Node next;
}
```

Implementieren Sie die folgende Methode *product* in der Klasse *Node*, welche das Produkt der *value* Attribute aller Elemente der zyklischen Liste zurückgeben soll. Sie dürfen in Ihrer Implementierung keine Schleifen verwenden, aber sie dürfen Hilfsmethoden schreiben und Rekursion verwenden.

Lösungsvorschlag:

```
public int product() {
  return this.next.product(this);
}

public int product(Node origin) {
  if (this == origin) {
    return this.value;
  } else {
    return this.value * this.next.product(origin);
  }
}
```