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

Probeklausur zur Vorlesung Programmierung

Lösungsvorschlag zu Aufgabe 1

```
int viertKleinstesElement(int a[ ]) throws BadInputException {
  if (a.length < 4) throw new BadInputException();

  for(int i = 0; i < 4; i++) {
    for (int j = i; j < a.length; j++) {
      if(a[j] < a[i]) {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
      }
    }
  }

  return a[3];
}
```

Lösungsvorschlag zu Aufgabe 2

a) Die folgende Tabelle ist die vollständige Ausgabe. Jede fehlerfrei ausgefüllte Zeile ist zwei Punkte wert, Folgefehler werden berücksichtigt.

i	a
-	$[6, 5, 4, 3, 1, 7, 2]$
0	$[5, 4, 3, 1, 6, 2, 7]$
1	$[4, 3, 1, 5, 2, 6, 7]$
2	$[3, 1, 4, 2, 5, 6, 7]$
3	$[1, 3, 2, 4, 5, 6, 7]$
4	$[1, 2, 3, 4, 5, 6, 7]$
5	$[1, 2, 3, 4, 5, 6, 7]$
6	$[1, 2, 3, 4, 5, 6, 7]$

b) Um das Array absteigend zu sortieren, muss schlicht der Vergleich $\textbf{if}(a[j-1] > a[j])$ in der vierten Zeile zu $\textbf{if}(a[j-1] < a[j])$ geändert werden.

Lösungsvorschlag zu Aufgabe 3

a) Die im Baum gespeicherten Wörter sind “progra”, “prolog”, “prüfung”, “prüfungen” und “prüfer”.

b)
```
public void printAllWords() {
  this.printAllWords("");
}

public void printAllWords(String prefix) {
  prefix = prefix + this.content;
  if (this.isEndOfWord) {
    System.out.println(prefix);
  }

  for (int i = 0; i < this.children.length; i++) {
    this.children[i].printAllWords(prefix);
  }
}
```

Lösungsvorschlag zu Aufgabe 4

```
public class Element⟨T⟩ {
  ...
  Element⟨T⟩ insert(T v, Comparator⟨T⟩ comp) {
    int c = comp.compare(v, this.val);
    if (c == 0) return this;
    else if (c < 0) {
      Element⟨T⟩ newEl = new Element⟨T⟩(v, this.prev, this);
      if (this.prev != null) this.prev.next = newEl;
      this.prev = newEl;
      return newEl;
    } else {
      if (this.next == null) {
        Element⟨T⟩ newEl = new Element⟨T⟩(v, this, null);
        this.next = newEl;
      } else this.next.insert(v, comp);
      return this;
    }
  }

  void accept(Visitor⟨T⟩ vis) {
    vis.visit(this);
    if (this.next != null) this.next.accept(vis);
  }
}

public class ReorderingListSetVisitor⟨T⟩ implements Visitor⟨T⟩ {
```

```
  private ListSet⟨T⟩ res;

  public ReorderingListSetVisitor(Comparator⟨T⟩ comp) {
    this.res = new ListSet⟨T⟩(comp);
  }

  public void visit(Element⟨T⟩ e) {
    this.res.insert(e.val);
  }

  public ListSet⟨T⟩ getRes() {
    return this.res;
  }
}
```

Lösungsvorschlag zu Aufgabe 5

```
public static double niederschlag(Wetter[ ] phaenomene) {
    double res = 0.0;
    for (int i = 0; i < phaenomene.length; i++) {
        if (phaenomene[i] instanceof Niederschlag) {
            res += ((Niederschlag)phaenomene[i]).getMenge();
        }
    }
    return res;
}
```

Lösungsvorschlag zu Aufgabe 6

a)

$\langle \mathtt{n} \geq 0 \rangle$

$\langle \mathtt{n} \geq 0 \wedge 1 = 1 \wedge 3 = 3 \wedge 1 = 1 \rangle$

```
i = 1;
```

$\langle \mathtt{n} \geq 0 \wedge \mathtt{i} = 1 \wedge 3 = 3 \wedge 1 = 1 \rangle$

```
k = 3;
```

$\langle \mathtt{n} \geq 0 \wedge \mathtt{i} = 1 \wedge \mathtt{k} = 3 \wedge 1 = 1 \rangle$

```
res = 1;
```

$\langle \mathtt{n} \geq 0 \wedge \mathtt{i} = 1 \wedge \mathtt{k} = 3 \wedge \mathtt{res} = 1 \rangle$

$\langle \mathtt{res} = \frac{3^{\mathtt{i}}-1}{2} \wedge \mathtt{k} = 3^{\mathtt{i}} \wedge \mathtt{i} \leq \mathtt{n}+1 \rangle$

```
while (i <= n){
```

$\langle \mathtt{res} = \frac{3^{\mathtt{i}}-1}{2} \wedge \mathtt{k} = 3^{\mathtt{i}} \wedge \mathtt{i} \leq \mathtt{n}+1 \wedge \mathtt{i} \leq \mathtt{n} \rangle$

$\langle \mathtt{res} + \mathtt{k} = \frac{3^{\mathtt{i}+1}-1}{2} \wedge \mathtt{k} * 3 = 3^{\mathtt{i}+1} \wedge \mathtt{i}+1 \leq \mathtt{n}+1 \rangle$

```
    res = res + k;
```

$\langle \mathtt{res} = \frac{3^{\mathtt{i}+1}-1}{2} \wedge \mathtt{k} * 3 = 3^{\mathtt{i}+1} \wedge \mathtt{i}+1 \leq \mathtt{n}+1 \rangle$

```
    k = k * 3;
```

$\langle \mathtt{res} = \frac{3^{\mathtt{i}+1}-1}{2} \wedge \mathtt{k} = 3^{\mathtt{i}+1} \wedge \mathtt{i}+1 \leq \mathtt{n}+1 \rangle$

```
    i = i + 1;
```

$\langle \texttt{res} = \frac{3^{\texttt{i}}-1}{2} \wedge \texttt{k} = 3^{\texttt{i}} \wedge \texttt{i} \leq \texttt{n}+1 \rangle$

}

$\langle \texttt{res} = \frac{3^{\texttt{i}}-1}{2} \wedge \texttt{k} = 3^{\texttt{i}} \wedge \texttt{i} \leq \texttt{n}+1 \wedge \texttt{i} \not\leq \texttt{n} \rangle$

$\langle \texttt{res} = \frac{3^{\texttt{n}+1}-1}{2} \rangle$

b) Eine gültige Variante für die Terminierung ist $V = \texttt{n} - \texttt{i}$, denn die Schleifenbedingung $B = \texttt{i} \leq \texttt{n}$ impliziert $\texttt{n} - \texttt{i} \geq 0$ und es gilt:

$\langle \texttt{n} - \texttt{i} = m \wedge \texttt{i} \leq \texttt{n} \rangle$

$\langle \texttt{n} - (\texttt{i}+1) < m \rangle$

```
res = res + k;
```

$\langle \texttt{n} - (\texttt{i}+1) < m \rangle$

```
k = k * 3;
```

$\langle \texttt{n} - (\texttt{i}+1) < m \rangle$

```
i = i + 1;
```

$\langle \texttt{n} - \texttt{i} < m \rangle$

Damit ist die Terminierung der einzigen Schleife in P gezeigt.

Lösungsvorschlag zu Aufgabe 7

```
findSum::[Int]→Int→Bool
findSum [ ] res = res == 0
findSum (x : xs) res = if findSum xs (res − x) then True
                       else if findSum xs (res + x) then True
                            else False
```

Lösungsvorschlag zu Aufgabe 8

a)
```
zipMult::MultTree a→MultTree b→MultTree (a, b)
zipMult (MultNode x xs) (MultNode y ys) =
    MultNode (x, y) (map (\(u, v)→zipMult u v) (zip xs ys))
```

b)
```
bfs::MultTree a→[a]
bfs t = bfsH [t]
    where
      bfsH [ ] = [ ]
      bfsH xs = (map getElem xs)++bfsH (concat (map getList xs))
      getElem (MultNode x _) = x
      getList (MultNode _ xs) = xs
```

Lösungsvorschlag zu Aufgabe 9

a) i) $f(a, Y, X), f(X, b, Y)$: clash failure a/b

ii) $f(X,Y,Z), f(h(c), g(X,X), g(Y,Y))$:
mgu $X/h(c), Y/g(h(c),h(c)), Z/g(g(h(c),h(c)), g(h(c),h(c)))$

iii) $f(X,Y,Z), f(Z,a,h(X))$: occur failure X in $h(X)$

iv) $f(X,Y,Z), f(g(Y,Y), g(Z,Z), h(X))$: occur failure Z in
$g(g(h(Z),h(Z)), g(h(Z),h(Z)))$

b) *removed*$(X, [X|Y], Y)$.
removed$(X, [X1|Y], [X1|Y\_ohne\_X]) :-$ *removed*(X, Y, Y_ohne_X).
permutation$([\,], [\,])$.
permutation$([X|Y], Z) :-$ *removed*$(X, Z, Z1)$, *permutation*$(Y, Z1)$.

c) i) `append([1],[],A).`
Antwortsubstitutionen: $\{A/[1]\}, \{A/[1]\}, \{A/[1]\}$

ii) `append(A,[],A).`
Antwortsubstitutionen: $\{A/[]\}, \{\}, \{A/[X]\}, \{A/[X|Y]\}$, terminiert nicht; Reihenfolge 1,2,3,1,2,3,1,2,3,...

iii) `append(A,B,[1,2,3]).`
Antwortsubstitutionen:
$\{A/[], B/[1,2,3]\}$, $\{A/[1,2,3], B/[]\}$, $\{A/[1], B/[2,3]\}$, $\{A/[1,2,3], B/[]\}$,
$\{A/[1,2], B/[3]\}$, $\{A/[1,2,3], B/[]\}$, $\{A/[1,2,3], B/[]\}$, $\{A/[1,2,3], B/[]\}$