14 Punkte Aufgabe 1 (O-Notation)

5 Punkte (a) Vereinfachen Sie die folgenden Funktionen mit Hilfe der O-Notation. Geben Sie dabei eine möglichst geringe obere Schranke an. Ein Beweis für die Gültigkeit Ihrer Aussage ist nicht erforderlich.

$T_1(n) = 4n + 23$

$T_2(n) = 23n^2 - 6n + 1024$

$T_3(n) = 18n^3 + n^2\log(n)$

$T_4(n) = (18n^3 + n^2)\log(n)$

$T_5(n, m) = nm + 63m\log(n)$

$T_6(n, m) = n\log(m) + m\log(n)$

$T_7(n, m, o) = 5n^3 + 8m^2 + 10m\log(o)$

$T_8(n, m) = 3m^3n^2 + 5m^2n^2 + m^3\log(n)$

$T_9(n, m) = \max(3^m + 5n^2, 16m^3 + 9n + 9867)$

$T_{10}(n, m) = 2^{nm} + (nm)^{12} + n^m$

Hinweis: Beachten Sie, daß eine falsche Antwort zu Punktabzug führt. Haben Sie also 5 richtige und 5 falsche Antworten, so ergibt sich eine Gesamtpunktzahl von 0. Insgesamt werden in dieser Teilaufgabe jedoch mindestens 0 Punkte vergeben.

9 Punkte (b) Geben Sie Laufzeitkomplexitäten für die folgenden Algorithmen an. Begründen Sie Ihre Entscheidung kurz.

```
algorithm alg1(n)
var prod;
begin
  prod := 1;
  for i := 1 to n do
    prod := prod * i;
  end for
  return prod;
end alg1;
```

```
algorithm alg2(n)
var p;
begin
  p := 1;
  while n > 1 do
    p := p * n;
    n := n / 3;
  end while
  return p;
end alg2;
```

```
algorithm alg3(n)
var p,r;
begin
  p := 2;
  r := 0;
  for i:=1 to n do
    for j := 1 to p do
      r := r + 1;
    end for
    p := 2*p;
  end for
  return r;
end alg3;
```

```
algorithm alg4(n)
var p, t;
begin
  p := 1;
  for i := 0 to n do
    for j := i to n do
      t := n;
      while t > 0 do
        p := p + 1;
        t := t / 2;
      end while
    end for
  end for
  return p;
end alg4;
```

Aufgabe 2 (Hashing) *16 Punkte*

Gegeben sei eine Folge von Elementen:

107, 20, 37, 46, 1, 2, 4, 50, 0

Fügen Sie die Elemente in eine Hashtabelle ein.

(a) Benutzen Sie geschlossenes Hashing mit $b = 2$, $m = 5$ und verallgemeinertes, lineares Sondieren als Kollisionsstrategie mit $c = 1$. Geben Sie die Zwischenergebnisse der $h_i(x)$ an. *6 Punkte*

(b) Benutzen Sie geschlossenes Hashing mit $b = 1$, $m = 11$ und Doppelhashing mit $h(x)$ als Divisionsmethode und $h'(x) = 2h(x)$. Geben Sie die Zwischenergebnisse der $h_i(x)$ an. *6 Punkte*

(c) Welche Eigenschaften sollte eine Hashfunktion haben und was muß zusätzlich für $h(x)$, $h'(x)$ beim Doppelhashing gelten? Beurteilen Sie die grundsätzliche Eignung von $h'(x)$ aus Aufgabenteil (b) als Hashfunktion sowie im Zusammenhang mit Doppelhashing. *4 Punkte*

14 Punkte

Aufgabe 3 (Radixsort)

Es sei *keytype* die Menge aller Wörter der Länge 4, die nur aus Kleinbuchstaben und dem Leerzeichen bestehen. Das Leerzeichen sei kleiner als 'a' und die Buchstaben seien gemäß der gewöhnlichen alphabetischen Ordnung sortiert. Ordnen Sie die folgenden Wörter mittels Radixsort und geben Sie das Ergebnis nach jeder Sortierphase an. Dabei genügt die Angabe nichtleerer Behälter. Wörter mit weniger als 4 Buchstaben werden am Ende mit Leerzeichen aufgefüllt. Die zu sortierende Folge lautet:

es sei die mit vier nur aus und dem das als der sie an

18 Punkte

Aufgabe 4 (Minimale Spannbäume)

8 Punkte (a) Geben Sie eine Definition für einen *minimalen Spannbaum* und beweisen oder widerlegen Sie folgende Eigenschaften:

(i) Zu einem gegebenen Graphen G gibt es genau einen minimalen Spannbaum.

(ii) Zu einem gegebenen Graphen G gibt es mindestens 2 minimale Spannbäume.

10 Punkte (b) Geben Sie die Idee für einen Algorithmus an, der zu zwei gegebenen Graphen G und T prüft, ob T ein minimaler Spannbaum von G ist.

22 Punkte

Aufgabe 5 (Rangebaum)

8 Punkte (a) Tragen Sie die Koordinaten $a = 1$, $b = 3$, $c = 5$, $d = 6$, $e = 3$, $f = 6$, $g = 128$ in einen Rangebaum ein.

Bauen Sie die Struktur so auf, daß der Platzbedarf lediglich $O(n \log N)$ beträgt, wobei n die Anzahl der einzutragenden Koordinaten und N die Anzahl der Rasterpunkte bezeichnet. In den Baum sollen später auch noch weitere Koordinaten aus dem Bereich 1-128 eingetragen werden können.

14 Punkte (b) Gegeben sei eine Java-Klasse *Rangebaum*, die folgende Funktionalität bietet:

```
public class Rangebaum {
  public Rangebaum(int[] raster) {...}
  public Object[] query(int links, int rechts) {...}
  public void insert(int koordinate, Object o) {...}
}
```

Der Konstruktor baut einen Rangebaum über die angegebenen Rasterkoordinaten auf. Mittels der *insert*-Methode können beliebige Objekte im Baum verwaltet werden. Die Methode *query* liefert dann alle Objekte, die in einem Anfrageintervall enthalten sind.

Formulieren und erläutern Sie Codebeispiele, die diese Klasse benutzen, um eine Menge von 2-dimensionalen Punkten der Klasse *Point* zu speichern und für ein gegebenes achsenparalleles Query-Rechteck R die darin enthaltenen Punkte effizient zu berechnen, wenn die Ergebnismenge klein im Verhältnis zur Punktmenge ist?

Die Laufzeit für eine Query darf höchstens $O((\log N_x + T_x)(\log N_y + t))$ betragen, wobei N_x (N_y) die Anzahl verschiedener x- (y-) Koordinaten, T_x die Anzahl unterschiedlicher x-Koordinaten im Anfragerechteck und t die Anzahl der gefundenen Punkte beschreibt.

Aufgabe 6 (B-Bäume)

16 Punkte

(a) Fügen Sie in einen initial leeren B-Baum der Ordnung 2 die folgenden Schlüssel ein. Geben Sie den Baum immer dann an, wenn eine Overflow-Operation nötig ist und markieren Sie den entsprechenden Knoten. *8 Punkte*

1, 2, 7, 9, 10, 8, 12, 20, 3, 4, 5, 6, 21, 22, 23, 13, 17, 19

(b) Löschen Sie nun aus dem erzeugten Baum aus Aufgabenteil (a) der Reihe nach die folgenden Elemente: *8 Punkte*

8, 12, 23, 1

Zeichnen Sie den Baum nach jeder Löschoperation und geben Sie an, welche Art von Underflow-Behandlung Sie verwenden.