Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (3 Punkte):
Aufgabe 2 (4 Punkte):
Aufgabe 3 (2 Punkte):
Aufgabe 4 (2 Punkte):
Aufgabe 5 (4 Punkte):
Aufgabe 6 (1 Punkt):
Aufgabe 7 (1 Punkt):
Aufgabe 8 (2 Punkte):
Aufgabe 9 (4 Punkte):
Aufgabe 10 (2 Punkte):
Aufgabe 11 (5 Punkte):
Aufgabe 12 (3 Punkte):
Aufgabe 13 (5 Punkte):
Aufgabe 14 (2 Punkte):

Gesamtpunkte (40 Punkte):

Schriftlicher Test (120 Minuten)
VU Einführung ins Programmieren für TM

29. September 2017

Aufgabe 1 (3 Punkte). Was sind die Bestandteile M, $e_{\min}$, $e_{\max}$ des Gleitkommazahlsystems $\mathbb{F}(2, M, e_{\min}, e_{\max})$? Wie lässt sich jede Gleitkommazahl $x \in \mathbb{F}(2, M, e_{\min}, e_{\max})$ darstellen? Welchen Wert haben die größte und die kleinste positive normalisierte Gleitkommazahl im `double`-Gleitkommazahlsystem $\mathbb{F}(2, 53, -1021, 1024)$?

Lösung zu Aufgabe 1.

Aufgabe 2 (4 Punkte). Was ist der Shell-Output des folgenden Programms?

```
#include <iostream>
using std::cout;

class dp {
private:
  int x;
  int y;
public:
  dp(int x, int y) {
    this->x = x;
    this->y = y;
    cout << "new: x=" << x << ", y=" << y <<"\n";
  }
  ~dp() {
    cout << "old: x=" << x << ", y=" << y <<"\n";
  }
  dp(const dp& input) {
    int a = input.x;
    int b = input.y;
    int c = 0;
    while ( a != b ) {
      if ( a < b ) {
        c = a;
        a = b;
        b = c;
      }
      a = a - b;
      cout << "a=" << a << ", b=" << b << ", c=" << c << "\n";
    }
    x = input.x/a;
    y = input.y/b;
  }
};


int main() {
  dp q(20,45);
  dp r = q;
  return 0;
}
```

Lösung zu Aufgabe 2.

Aufgabe 3 (2 Punkte). Was ist eine rekursive Funktion und was darf dabei nicht fehlen? Erläutern Sie das Konzept anhand eines selbstgewählten Beispiels und geben Sie einen entsprechenden C/C++ Code an!

Lösung zu Aufgabe 3.

Aufgabe 4 (2 Punkte). Eine obere Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ ist eine Matrix mit der Eigenschaft $A_{jk} = 0$ für $j > k$, d.h.

$$A = \begin{pmatrix} A_{00} & A_{01} & A_{02} & \dots & A_{0,n-1} \\ 0 & A_{11} & A_{12} & \dots & A_{1,n-1} \\ 0 & 0 & A_{22} & \dots & A_{2,n-1} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & A_{n-1,n-1} \end{pmatrix}$$

Zur effizienten Speicherung wird $A \in \mathbb{R}^{n\times n}$ in Form eines Vektors $a \in \mathbb{R}^N$ mit $N = \sum_{j=1}^n j = \frac{n(n+1)}{2}$ abgelegt, d.h. $A_{jk} = a_\ell$ für einen geeigneten Index ℓ, der eindeutig von j und k abhängen muss. Leiten Sie eine Formel für ℓ her (in Abhängigkeit von j und k). Begründen Sie Ihre Formel.

Hinweis. Am einfachsten ist spaltenweise Speicherung der Einträge A_{jk} für $j \geq k$, d.h.

$$a = (A_{00}, A_{01}, A_{11}, A_{02}, A_{12}, A_{22}, \dots, A_{n-1,n-1}) \in \mathbb{R}^N.$$

Lösung zu Aufgabe 4.

Hinweis. In den folgenden Aufgaben seien die oberen Dreiecksmatrizen $A \in \mathbb{R}^{n\times n}$ in Objekten der C++ Klasse `TriMatrix` gespeichert, die unten definiert ist. Neben Konstruktor (mit optionalem Initialisierungswert), Kopierkonstruktor, Destruktor und Zuweisungsoperator gibt es eine Methode, um die Dimension n auszulesen (`size`). Der Koeffizientenvektor `coeff` speichert nur die $\frac{n(n+1)}{2}$ nicht-trivialen Einträge A_{jk} mit $j \leq k$, auf die mittels `A(j,k)` lesend und schreibend zugegriffen wird:

```
class TriMatrix {
private:
  int n;
  double* coeff;
public:
  TriMatrix(int n=0, double init=0);
  TriMatrix(const TriMatrix&);
  ~TriMatrix();
  TriMatrix& operator=(const TriMatrix&);
  int size() const;
  const double& operator()(int j, int k) const;
  double& operator()(int j, int k);
};
```

Aufgabe 5 (4 Punkte). Schreiben Sie den Konstruktor der Klasse `TriMatrix`. Stellen Sie mittels `assert` sicher, dass $n \geq 0$ ist, wobei für $n = 0$ eine leere Matrix angelegt werde.
Hinweis. Beachten Sie, dass `coeff` ein Vektor der Länge $\frac{n(n+1)}{2}$ ist.

Lösung zu Aufgabe 5.

Aufgabe 6 (1 Punkt). Schreiben Sie den Destruktor der Klasse `TriMatrix`.

Lösung zu Aufgabe 6.

Aufgabe 7 (1 Punkt). Schreiben Sie die Methode `size` der Klasse `TriMatrix`.

Lösung zu Aufgabe 7.

Aufgabe 8 (2 Punkte). Schreiben Sie den Koeffizientenzugriff der Klasse `TriMatrix` für `const`-Objekte. Stellen Sie mittels `assert` sicher, dass für $A \in \mathbb{R}^{n \times n}$ die Indizes $0 \leq j \leq k \leq n-1$ erfüllen.
Hinweis. Verwenden Sie Ihre Formel aus Aufgabe 4.

Lösung zu Aufgabe 8.

Aufgabe 9 (4 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `TriMatrix`.

Lösung zu Aufgabe 9.

Aufgabe 10 (2 Punkte). Beweisen Sie mathematisch, dass das Produkt $C = AB \in \mathbb{R}^{n\times n}$ zweier oberer Dreiecksmatrizen $A, B \in \mathbb{R}^{n\times n}$ wieder eine obere Dreiecksmatrix ist, indem Sie die Laufindizes der Summe des allgemeinen Matrizenprodukts

$$C_{j\ell} = \sum_{k=0}^{n-1} A_{jk} B_{k\ell} \quad \text{für } j, \ell = 0, \ldots, n-1$$

mithilfe der Dreiecksstruktur von A und B vereinfachen.
Hinweis: Eine obere Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ ist durch $A_{jk} = 0$ für $j > k$ charakterisiert.

Lösung zu Aufgabe 10.

Aufgabe 11 (5 Punkte). Überladen Sie den `*` Operator so, dass er das Produkt $C = AB \in \mathbb{R}^{n\times n}$ zweier oberer Dreiecksmatrizen $A, B \in \mathbb{R}^{n\times n}$ berechnet. Stellen Sie mittels `assert` sicher, dass A und B dieselbe Dimension haben.

Hinweis. Beachten Sie, dass die Funktion nur Koeffizienten C_{jk} für $0 \le j \le k \le n-1$ berechnen soll und auch nur auf entsprechende Koeffizienten von A und B zugreifen darf. Verwenden Sie dazu Ihre Erkenntnisse aus Aufgabe 10.

Lösung zu Aufgabe 11.

Hinweis. In den folgenden Aufgaben seien Vektoren $x \in \mathbb{R}^n$ in Objekten der C++ Klasse `Vector` gespeichert, die unten definiert ist. Neben Konstruktor, Kopierkonstruktor, Destruktor und Zuweisungsoperator gibt es eine Methode, um die Dimension n auszulesen (`size`). Auf die Koeffizienten x_j des Vektors kann mittels `x(j)` für $0 \leq j \leq n-1$ zugegriffen werden. Sie müssen keine der genannten Methoden implementieren!

```
class Vector {
private:
  int n;
  double* coeff;
public:
  Vector(int n=0, double init=0);
  Vector(const Vector&);
  ~Vector();
  Vector& operator=(const Vector&);
  int size() const;
  const double& operator()(int j) const;
  double& operator()(int j);
};
```

Aufgabe 12 (3 Punkte). Leiten Sie für gegebenes $b \in \mathbb{R}^n$ eine Formel her, um für eine obere Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ mit $A_{jj} \neq 0$ für alle $j = 0, \dots, n-1$ die Lösung $x \in \mathbb{R}^n$ von $Ax = b$ zu berechnen, indem Sie die Formel des Matrix-Vektor-Produkts

$$b_j = (Ax)_j = \sum_{k=0}^{n-1} A_{jk} x_k$$

mithilfe der Dreiecksstruktur von A vereinfachen. In welcher Reihenfolge müssen die Koeffizienten b_j berechnet werden, damit der Algorithmus wohldefiniert ist (d.h. alle in der Formel benötigten x_k sind bereits bekannt)?

Hinweis. Eine obere Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ ist durch $A_{jk} = 0$ für $j > k$ charakterisiert.

Lösung zu Aufgabe 12.

Aufgabe 13 (5 Punkte). Überladen Sie den `|` Operator so, dass `x = A|b` für eine obere Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ (vom Typ `TriMatrix`) und einen Vektor $b \in \mathbb{R}^n$ (vom Typ `Vector`) die Lösung $x \in \mathbb{R}^n$ von $Ax = b$ (als Objekt vom Typ `Vector`) berechnet. Stellen Sie mittels `assert` sicher, dass A und b passende Dimension haben und dass $A_{jj} \neq 0$ für alle $j = 0, \dots, n-1$.

Lösung zu Aufgabe 13.

Aufgabe 14 (2 Punkte). Bestimmen Sie den Aufwand Ihrer Funktion aus Aufgabe 13. Falls die Funktion für $n = 10^3$ eine Laufzeit von 3 Sekunden hat, welche Laufzeit erwarten Sie aufgrund des Aufwands für $n = 3 \cdot 10^3$? Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 14.