Aufgabe 1 (1 Punkt):
Aufgabe 2 (4 Punkte):
Aufgabe 3 (1 Punkt):
Aufgabe 4 (3 Punkte):
Aufgabe 5 (2 Punkte):
Aufgabe 6 (3 Punkte):
Aufgabe 7 (1 Punkt):
Aufgabe 8 (2 Punkte):
Aufgabe 9 (2 Punkte):
Aufgabe 10 (1 Punkt):
Aufgabe 11 (3 Punkte):
Aufgabe 12 (2 Punkte):
Aufgabe 13 (2 Punkte):
Aufgabe 14 (2 Punkte):
Aufgabe 15 (3 Punkte):
Aufgabe 16 (8 Punkte):

Gesamtpunkte (40 Punkte):

Familienname:

Vorname:

Matrikelnummer:

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

27. Februar 2018

Aufgabe 1 (1 Punkt). Was sind Lifetime und Scope einer Variable?

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 (1 Punkt). Was ist eine rekursive Funktion und was darf dabei nicht fehlen?

Lösung zu Aufgabe 3.

Aufgabe 4 (3 Punkte). Schreiben Sie eine rekursive Funktion `binomial`, die mit Hilfe des Additionstheorems

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \quad \text{für } k, n \in \mathbb{N} \text{ mit } 2 \leq k < n$$

den Binomialkoeffizienten berechnet. Beachten Sie dabei die Grenzfälle

$$\binom{n}{1} = n \quad \text{und} \quad \binom{n}{n} = 1 \quad \text{für alle } n \geq 1.$$

Stellen Sie mittels `assert` sicher, dass $1 \leq k \leq n$ gilt.

Lösung zu Aufgabe 4.

Hinweis. In den folgenden Aufgaben betrachten wir Matrizen $A \in \mathbb{R}^{n\times n}$ als Objekte der C++ Klasse `Matrix`, die unten definiert ist. Neben Konstruktor, Kopierkonstruktor, Destruktor und Zuweisungsoperator, gibt es eine Methode, um die Dimension n auszulesen (`dim`) und um zu bestimmen, ob A symmetrisch ist (`isSymmetric`). Die Koeffizienten A_{jk} erhält man mittels `A(j,k)`. Intern wird eine Matrix $A \in \mathbb{R}^{n\times n}$ spaltenweise in einem dynamischen Vektor der Länge n^2 gespeichert.

```cpp
class Matrix {
private:
  int n;
  double* entry;

public:
  Matrix(int n = 0);
  ~Matrix();
  Matrix(const Matrix&);
  const Matrix& operator=(const Matrix&);

  int dim() const;
  double& operator()(int j, int k);
  const double& operator()(int j, int k) const;

  bool isSymmetric() const;
};
```

Aufgabe 5 (2 Punkte). Matrizen sollen intern spaltenweise gespeichert werden, d.h. die Matrix $A \in \mathbb{R}^{n\times n}$ liegt als Vektor

$$a = (A_{00}, A_{10}, A_{20}, \dots, A_{n-1,0}, A_{01}, A_{11}, \dots, A_{n-1,1}, \dots, A_{n-1,n-1}) \in \mathbb{R}^{n^2} \tag{1}$$

im Speicher. Leiten Sie eine Formel her, die jedem Indexpaar (j,k) mit $j,k \in \{0,\dots,n-1\}$ den eindeutigen Index $\ell \in \{0,\dots,n^2-1\}$ zuordnet, sodass $a_\ell = A_{jk}$ gilt. Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 5.

Aufgabe 6 (3 Punkte). Schreiben Sie den Konstruktor der Klasse `Matrix`, der für $n \in \mathbb{N}_0$ eine Matrix $A \in \mathbb{R}^{n \times n}$ anlegt und die Koeffizienten mit Null initialisiert. Für $n = 0$ soll kein zusätzlicher Speicher angelegt werden. Stellen Sie mittels `assert` sicher, dass $n \geq 0$ gilt.

Lösung zu Aufgabe 6.

Aufgabe 7 (1 Punkt). Schreiben Sie den Destruktor der Klasse `Matrix`.

Lösung zu Aufgabe 7.

Aufgabe 8 (2 Punkte). Schreiben Sie den Kopierkonstruktor der Klasse `Matrix`.

Lösung zu Aufgabe 8.

Aufgabe 9 (2 Punkte). Schreiben Sie den Zugriffsoperator der Klasse `Matrix` für nicht-konstante Objekte, sodass `A(j,k)` für eine Matrix $A \in \mathbb{R}^{n\times n}$ den Eintrag A_{jk} zurückgibt. Stellen Sie mittels `assert` sicher, dass die Indizes j und k zulässig sind, d.h. $0 \leq j, k \leq n-1$. Nutzen Sie Ihre Formel aus Aufgabe 5.

Lösung zu Aufgabe 9.

Aufgabe 10 (1 Punkt). Erklären Sie die Bedeutung der folgenden C++ Zeile

```
const Matrix& A = *this;
```

wenn Sie diese in einer Methode der Klasse `Matrix` verwenden.

Lösung zu Aufgabe 10.

Aufgabe 11 (3 Punkte). Schreiben Sie die Methode `isSymmetric` der Klasse `Matrix`. Falls $A \in \mathbb{R}^{n\times n}$ symmetrisch ist, d.h. $A_{jk} = A_{kj}$ für alle $j, k \in \{0, \ldots, n-1\}$, soll die Methode `true` zurückgeben, anderenfalls `false`.

Lösung zu Aufgabe 11.

Aufgabe 12 (2 Punkte). Bestimmen Sie den Aufwand Ihrer Funktion `isSymmetric` aus Aufgabe 11. Falls die Methode für eine symmetrische Matrix $A \in \mathbb{R}^{n \times n}$ mit $n = 5.000$ eine Laufzeit von 2 Sekunden hat, welche Laufzeit erwarten Sie für $n = 20.000$?

Lösung zu Aufgabe 12.

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 (`dim`). 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* entry;

public:
  Vector(int n = 0);
  ~Vector();
  Vector(const Vector&);
  const Vector& operator=(const Vector&);

  int dim() const;
  double& operator()(int j);
  const double& operator()(int j) const;
};
```

Aufgabe 13 (2 Punkte). Überladen Sie den Operator * so, dass er für einen Skalar $\lambda \in \mathbb{R}$ und einen Vektor $x \in \mathbb{R}^n$ den skalierten Vektor

$$y := \lambda x = (\lambda\, x_0, \dots, \lambda\, x_{n-1}) \in \mathbb{R}^n$$

zurückgibt. Verwenden Sie die Signatur

```
const Vector operator*(double lambda, const Vector& x);
```

Lösung zu Aufgabe 13.

Aufgabe 14 (2 Punkte). Überladen Sie den Operator `*` so, dass er für zwei Vektoren $x, y \in \mathbb{R}^n$ das Skalarprodukt

$$x * y := \sum_{k=0}^{n-1} x_k y_k$$

berechnet. Stellen Sie mittels `assert` sicher, dass x und y dieselbe Dimension haben. Verwenden Sie die Signatur

```
double operator*(const Vector& x, const Vector& y);
```

Lösung zu Aufgabe 14.

Aufgabe 15 (3 Punkte). Überladen Sie den Operator `*` so, dass er für eine Matrix $A \in \mathbb{R}^{n \times n}$ und einen Vektor $x \in \mathbb{R}^n$ das Matrix-Vektor-Produkt $b = A * x \in \mathbb{R}^n$ berechnet, d.h.

$$b_j = \sum_{k=0}^{n-1} A_{jk} x_k \quad \text{für alle } j = 0, \ldots, n-1.$$

Stellen Sie mittels `assert` sicher, dass A und x dieselbe Dimension haben. Verwenden Sie die Signatur

```
const Vector operator*(const Matrix& A, const Vector& x);
```

Lösung zu Aufgabe 15.

Aufgabe 16 (8 Punkte). Die Power-Iteration ist ein numerisches Verfahren, um einen Eigenvektor zum größten Eigenwert einer symmetrischen Matrix A zu approximieren. Die Funktion `poweriteration`, die Sie schreiben sollen, nimmt als Input eine symmetrische Matrix $A \in \mathbb{R}^{n \times n}$, einen Startvektor $x^{(0)} \in \mathbb{R}^n \backslash \{0\}$ und eine Toleranz $\tau > 0$. Zunächst wird folgende Startrechnung durchgeführt:

- Berechne das Matrix-Vektor-Produkt $y^{(0)} = A * x^{(0)}$.
- Berechne $\lambda^{(0)} = (x^{(0)} * y^{(0)})/(x^{(0)} * x^{(0)}) \in \mathbb{R}$.

Anschließend werden für $k \in \mathbb{N}_0$ die folgenden Schritte ausgeführt:

- Berechne $\mu^{(k)} = 1/\sqrt{y^{(k)} * y^{(k)}} \in \mathbb{R}$.
- Berechne den Vektor $x^{(k+1)} = \mu^{(k)} * y^{(k)} \in \mathbb{R}^n$.
- Berechne das Matrix-Vektor-Produkt $y^{(k+1)} = A * x^{(k+1)} \in \mathbb{R}^n$.
- Berechne das Skalarprodukt $\lambda^{(k+1)} = x^{(k+1)} * y^{(k+1)} \in \mathbb{R}$.

Die Rechnung werde beendet, sobald für ein $k \in \mathbb{N}_0$ gilt

$$|\lambda^{(k+1)} - \lambda^{(k)}| \leq \begin{cases} \tau & \text{falls } |\lambda^{(k)}| \leq \tau, \\ \tau \, |\lambda^{(k)}| & \text{anderenfalls.} \end{cases}$$

In diesem Fall, soll die Funktion den Vektor $x^{(k+1)} \in \mathbb{R}^n$ zurückgeben. Stellen Sie mittels `assert` sicher, dass die Matrix A symmetrisch ist, dass A und $x^{(0)}$ dieselbe Dimension haben, dass $x^{(0)} * x^{(0)} > 0$ ist (d.h. $x^{(0)}$ ist nicht der Nullvektor) und dass $\tau > 0$ gilt.

Hinweis. Ihre Implementierung muss nur einen x-Vektor und einen y-Vektor und zwei λ-Werte gleichzeitig im Speicher halten! Sie können die Funktionen `fabs` (Absolutbetrag) und `sqrt` (Wurzel) aus der `cmath`-Bibliothek verwenden.

Lösung zu Aufgabe 16.