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

Gesamtpunktzahl:

Familienname:

Vorname:

Matrikelnummer:

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

1. März 2016

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? Was berechnet die Funktion `function`?

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

int function(int n, int k) {
  int val = 0;
  if (k == 1) {
    val = n;
  }
  else if (k == n) {
    val = 1;
  }
  else {
    val = function(n-1,k) + function(n-1,k-1);
  }
  cout << n << ", " << k << ", " << val << "\n";
  return val;
}

int main() {
  function(5,2);
  return 0;
}
```

Lösung zu Aufgabe 2.

Aufgabe 3 (2 Punkte). Der folgende C++ Code hat 4 verschiedene Syntax-Fehler. Markieren Sie diese und erläutern Sie, was warum falsch ist!

```cpp
#include<iostream>

using std::endl;
using std::string;

class Base {
private:
  int a;
public:
  Base(int a = 0) {
    this->a = a;
  }
  string ego() {
    return "Base";
  }
  void set(int a) {
    this->a = a;
    cout << ego() << ", a=" << a << endl;
  }
};

class Derived:public Base {
private:
  int b;
public:
  Derived(int a) {
    b = a;
    this->a = 2*a;
  }
  string ego() {
    return "Derived";
  }
  void set() {
    cout << ego() << endl;
  }

};

int main(){
  Base dp;
  Derived mr;
  dp.set(2);
  mr.set(2);
  return 0;
}
```

Lösung zu Aufgabe 3.

Hinweis. In den folgenden Aufgaben seien die Polynome $p(x) = \sum_{j=0}^{n} a_j x^j$ als Objekte der C++ Klasse `Polynomial` gespeichert, die unten definiert ist. Neben Konstruktor, Kopierkonstruktor, Destruktor und Zuweisungsoperator, gibt es eine Methode, um den Grad n auszulesen (`degree`), und eine, um die Ableitung von p zu berechnen (`diff`). Den j-ten Koeffizienten a_j von p erhält man mittels `p[j]`, den Funktionswert an einer Stelle $x \in \mathbb{R}$ durch `p(x)`.

```
1 class Polynomial {
2 private:
3   int n;
4   double* a;
5 public:
6   Polynomial(int n=0);
7   Polynomial(const Polynomial&);
8   ~Polynomial();
9   Polynomial& operator=(const Polynomial&);
10  int degree() const;
11  Polynomial diff() const;
12  double operator()(double x) const;
13  const double& operator[](int j) const;
14  double& operator[](int j);
15 };
```

Aufgabe 4 (1 Punkt). Erläutern Sie die Bedeutung der beiden `const` in Zeile 13 der Klassendefinition.

Lösung zu Aufgabe 4.

Aufgabe 5 (3 Punkte). Schreiben Sie den Konstruktor der Klasse `Polynomial`, der den Koeffizientenvektor mit Null initialisiert. Stellen Sie mittels `assert` sicher, dass $n \geq 0$ ist.
Hinweis. Beachten Sie, dass der Koeffizientenvektor ein Vektor der Länge $n + 1$ ist.

Lösung zu Aufgabe 5.

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

Lösung zu Aufgabe 6.

Aufgabe 7 (1 Punkt). Schreiben Sie die Methode `degree` der Klasse `Polynomial`.

Lösung zu Aufgabe 7.

Aufgabe 8 (2 Punkte). Schreiben Sie den Koeffizientenzugriff der Klasse `Polynomial` für `const`-Objekte. Stellen Sie mittels `assert` sicher, dass der Index $0 \leq j \leq n$ erfüllt, wenn $p = \sum_{j=0}^{n} a_j x^j$ ein Polynom vom Grad n ist.

Lösung zu Aufgabe 8.

Aufgabe 9 (3 Punkte). Schreiben Sie den Zuweisungsoperator der Klasse `Polynomial`.

Lösung zu Aufgabe 9.

Aufgabe 10 (1 Punkt). Es sei $p(x) = \sum_{j=0}^{n} a_j x^j$ ein Polynom vom Grad n. Geben Sie eine explizite Formel an für die Ableitung von $p'(x)$! Was passiert im Fall $n = 0$?

Lösung zu Aufgabe 10.

Aufgabe 11 (3 Punkte). Schreiben Sie die Methode `diff` der Klasse `Polynomial`. Beachten Sie den Sonderfall, dass $p = \sum_{j=0}^{n} a_j x^j$ ein Polynom vom Grad $n = 0$ ist.

Lösung zu Aufgabe 11.

Aufgabe 12 (4 Punkte). Überladen Sie den + Operator so, dass er die Summe $r = p + q$ zweier Polynome $p(x) = \sum_{j=0}^{m} a_j x^j$ und $q(x) = \sum_{k=0}^{n} b_k x^k$ berechnet und zurückgibt. Beachten Sie, dass die Polynome p und q unterschiedlichen Grad $m \neq n$ haben können.

Lösung zu Aufgabe 12.

Aufgabe 13 (3 Punkte). Überladen Sie den $*$ Operator so, dass er die Skalarmultiplikationen $r = \lambda p$ bzw. $r = p\lambda$ berechnet, wobei $p(x) = \sum_{j=0}^{n} a_j x^j$ ein Polynom ist und $\lambda \in \mathbb{R}$ ein Skalar, d.h. $r(x) = \sum_{j=0}^{n} b_j x^j$ mit $b_j = \lambda a_j$.

Lösung zu Aufgabe 13.

Aufgabe 14 (3 Punkte). Überladen Sie den $*$ Operator so, dass er das Produkt $r = pq$ zweier Polynome $p(x) = \sum_{j=0}^{m} a_j x^j$ und $q(x) = \sum_{k=0}^{n} b_k x^k$ berechnet und zurückgibt.

Hinweis. Beachten Sie, dass r ein Polynom vom Grad $m + n$ ist. Die Koeffizienten von $r(x) = \sum_{\ell=0}^{m+n} c_\ell x^\ell$ sind gerade gegeben durch

$$c_\ell = \sum_{\substack{j+k=\ell \\ j\in\{0,\dots,m\} \\ k\in\{0,\dots,n\}}} a_j b_k.$$

Lösung zu Aufgabe 14.

Aufgabe 15 (2 Punkte). Bestimmen Sie den Aufwand Ihrer Funktion aus Aufgabe 14 für zwei Polynome p und q vom selben Grad n. Falls die Funktion für $n = 10^2$ eine Laufzeit von 1 Sekunden hat, welche Laufzeit erwarten Sie aufgrund des Aufwands für $n = 10^3$? Begründen Sie Ihre Antwort!

Lösung zu Aufgabe 15.

Aufgabe 16 (4 Punkte). Eine Möglichkeit, eine Nullstelle eines Polynoms $p(x)$ zu berechnen, ist das Newton-Verfahren. Ausgehend von einem Startwert x_0 definiert man induktiv eine Folge (x_n) durch

$$x_{k+1} = x_k - p(x_k)/p'(x_k).$$

Schreiben Sie eine Funktion `root`, die zu gegebenem Polynom $p(x)$, Startwert x_0 und Toleranz $\tau > 0$ das Newton-Verfahren durchführt, wobei die Iteration endet, falls

$$|p(x_n)| \le \tau \quad \text{und} \quad |x_n - x_{n-1}| \le \tau$$

gelten. In diesem Fall werde der letzte Wert x_n als Approximation der Nullstelle zurückgegeben. **Hinweis.** Verwenden Sie die Klasse `Polynomial`. Stellen Sie mittels `assert` sicher, dass $\tau > 0$ gilt. Vermeiden Sie es, alle Folgenglieder zu speichern, weil der Algorithmus ja in jedem Schritt nur die Folgenglieder x_{n-1} und x_n benötigt.