Familienname:

Vorname:

Matrikelnummer:

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

Gesamtpunktzahl:

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

30. September 2014

Aufgabe 1 (2 Punkte). Was ist der Unterschied zwischen einer imperativen Programmiersprache (z.B. C) und einer objektorientierten Programmiersprache (z.B. C++)?

Aufgabe 2 (4 Punkte). Was ist der Output des folgenden Programms? Zeichnen Sie einen Zeitstrahl, wo Sie Lifetime und Scope der Variablen `x`, `y` und `z` auftragen. Kennzeichnen Sie am Zeitstrahl die einzelnen Blöcke und Funktionen.

```
1  #include <stdio.h>
2
3  int z = 3;
4
5  int max(int,int);
6
7  main() {
8    int x = 1;
9    int y = 2;
10
11   printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
12
13   {
14     int x = 100;
15     y = 3;
```

```
16        z = max(x,y);
17        printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
18      }
19
20      printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
21    }
22
23    int max(int x, int y) {
24
25      printf("(x,y,z) = (%d,%d,%d)\n",x,y,z);
26
27      if(x>=y) {
28        return x;
29      }
30      else {
31        return y;
32      }
33    }
```

Aufgabe 3 (2 Punkte). Schreiben Sie einen C-Strukturdatentyp `vector` zur Speicherung von Vektoren $x \in \mathbb{R}^N$. In der Struktur soll neben dem dynamischen Vektor der Koeffizienten auch die Dimension $N \in \mathbb{N}$ gespeichert werden.

Aufgabe 4 (2 Punkte). Schreiben Sie eine Funktion `newVector`, die einen Vektor $x \in \mathbb{R}^N$ allokiert und mit Null initialisiert. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Aufgabe 5 (1 Punkt). Schreiben Sie eine Funktion `getVectorCoeff`, die den Koeffizienten x_j eines Vektors $x \in \mathbb{R}^N$ zurückgibt. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Aufgabe 6 (4 Punkte). Schreiben Sie eine Funktion `normVector`, die für einen Vektor $x \in \mathbb{R}^N$ die Norm

$$\|x\| := \max_{j,k=1,\dots,N} \Big(|x_j - x_k| + |x_j + x_k| \Big)$$

berechnet und zurückgibt. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Aufgabe 7 (3 Punkte). Was versteht man unter Aufwand? Welchen Aufwand hat Ihre Implementierung der Norm aus Aufgabe 6? Erklären Sie in diesem Zusammenhang auch die Landau-$\mathcal{O}(\cdot)$-Notation.

Aufgabe 8 (3 Punkte). Schreiben Sie eine Funktion `minabsVector`, die für einen Vektor $x \in \mathbb{R}^N$ den letzten Index $k \in \{1, \dots, N\}$ mit

$$|x_k| = \min_{j=1,\dots,N} |x_j|$$

bestimmt und zurückgibt. Für $x = (1, -7, 4, 3, -1, 5)$ werde beispielsweise 5 zurückgegeben. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3.

Aufgabe 9 (4 Punkte). Schreiben Sie eine Funktion `posVector`, die für einen Vektor $x \in \mathbb{R}^N$ den Vektor $y \in \mathbb{R}^n$ mit $n \leq N$ zurückgibt, bei dem alle Einträge x_j mit $x_j \leq 0$ gestrichen werden. Für $x = (1, 4, -2, 3, -4)$ gilt beispielsweise $y = (1, 4, 3)$. Für $n = 0$ werde `NULL` zurückgegeben. **Hinweis.** Verwenden Sie den Strukturdatentyp aus Aufgabe 3. Sie dürfen die Existenz und kanonische Funktionalität einer Funktion `setVectorCoeff` voraussetzen. Bestimmen Sie in Ihrer Implementierung zunächst die Länge n, und legen Sie dann den Vektor $y \in \mathbb{R}^n$ mittels `newVector` an.

Aufgabe 10 (4 Punkte). Schreiben Sie eine C++ Klasse `Vector` zur Speicherung von Vektoren $x \in \mathbb{R}^N$. In der Klasse soll neben dem dynamischen Vektor der Koeffizienten (`double *`) auch die Dimension $N \in \mathbb{N}$ gespeichert werden. Ferner soll die Klasse die folgenden Methoden enthalten:

- Destruktor,
- Konstruktor zum Allokieren des Null-Vektors $x \in \mathbb{R}^N$ mit $x_j = 0$,
- Konstruktor zum Allokieren eines konstanten Vektors $x \in \mathbb{R}^N$ mit $x_j = C \in \mathbb{R}$,
- `getSize`-Methode, um die Länge N des Vektors auszulesen,
- `getEntry` und `setEntry`, um einen Koeffizienten x_j zu lesen oder zu schreiben.

Hinweis. An dieser Stelle sollen nur die Signaturen implementiert, nicht die Funktionalität der Methoden. Der Einfachheit halber verzichten wir in dieser Aufgabe auf die Dreierregel, d.h. Sie brauchen keinen Kopierkonstruktor bzw. Zuweisungsoperator zu implementieren.

Aufgabe 11 (3 Punkte). Schreiben Sie die beiden Konstruktoren der Klasse `Vector` aus Aufgabe 9. **Hinweis.** Verwenden Sie `new`.

Aufgabe 12 (1 Punkt). Schreiben Sie den Destruktur der Klasse `Vector` aus Aufgabe 9. **Hinweis.** Verwenden Sie `delete`.

Aufgabe 13 (3 Punkte). Überladen Sie den `+` Operator so, dass er die Summe $x + y$ zweier Vektoren $x, y \in \mathbb{R}^N$ gleicher Länge berechnet. **Hinweis.** Verwenden Sie die Klasse `Vector` aus Aufgabe 9.

Aufgabe 14 (4 Punkte). Was ist der Output des folgenden Codeabschnittes? Erklären Sie darüberhinaus die Bedeutung und die (allgemeine) Syntax von Zeile 15. Erklären Sie weiters was in Zeile 18 passiert.

```
1  #include<iostream>
2  using namespace std;
3
4  class Konto {
5    int nummer;
6  public:
7    Konto(int nr) : nummer(nr) {
8      cout << "Konto mit Nummer " << nummer << " erzeugt." << endl;
9    }
10   ~Konto() {
11     cout << "Konto mit Nummer " << nummer << " geloescht." << endl;
12   }
```

```
13  };
14
15  class SparKonto : public Konto {
16    double zinsSatz;
17  public:
18    SparKonto(int nr, double z) : Konto(nr), zinsSatz(z) {
19      cout << "Dient als Sparkonto mit Zins " << zinsSatz << endl;
20    }
21    ~SparKonto() {
22      cout << "Sparkonto geschlossen." << endl;
23    }
24  };
25
26  int main() {
27    SparKonto miete(555333111,0.001);
28
29    return 0;
30  }
31}
```