Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (2 Punkte):
Aufgabe 2 (4 Punkte):
Aufgabe 3 (2 Punkte):
Aufgabe 4 (1 Punkte):
Aufgabe 5 (2 Punkte):
Aufgabe 6 (4 Punkte):
Aufgabe 7 (5 Punkte):
Aufgabe 8 (3 Punkte):
Aufgabe 9 (7 Punkte):

Gesamtpunktzahl:

Schriftlicher Test zu C (60 Minuten)
VU Einführung ins Programmieren für TM

25. November 2011

Aufgabe 1 (2 Punkte). Schreiben Sie einen Struktur-Datentyp `Matrix` zur Speicherung von Matrizen $A \in \mathbb{R}^{n \times n}$ beliebiger Dimension $n \in \mathbb{N}$. In der Struktur sollen neben der Dimension n die Koeffizienten A_{jk} spaltenweise in Form eines `double`-Vektor gespeichert werden.

ACHTUNG: Diese Struktur soll auch in allen nachfolgenden Aufgaben verwendet werden.

Aufgabe 2 (4 Punkte). Schreiben Sie eine Funktion `newMatrix`, die für gegebenes $n \in \mathbb{N}$ eine Matrix $A \in \mathbb{R}^{n \times n}$ allokiert und initialisiert.

Aufgabe 3 (2 Punkte). Schreiben Sie eine Funktion `delMatrix`, die den Speicher einer mittels `newMatrix` angelegten Matrix $A \in \mathbb{R}^{n \times n}$ freigibt und den `NULL`-Pointer zurückgibt.

Aufgabe 4 (1 Punkt). Schreiben Sie eine Funktion `getMatrixDimension`, die die Dimension $n \in \mathbb{N}$ einer mittels `newMatrix` angelegten Matrix $A \in \mathbb{R}^{n \times n}$ zurückgibt.

Aufgabe 5 (2 Punkte). Schreiben Sie eine Funktion `getMatrixCoeff`, die für gegebene Indizes j, k den Koeffizienten A_{jk} einer mittels `newMatrix` angelegten Matrix $A \in \mathbb{R}^{n \times n}$ zurückgibt.

Aufgabe 6 (4 Punkte). Schreiben Sie eine Funktion `isUMatrix`, die überprüft, ob eine gegebene Matrix $A \in \mathbb{R}^{n \times n}$ eine obere Dreiecksmatrix ist, d.h. $A_{jk} = 0$ für alle $j > k$.

Aufgabe 7 (5 Punkte). Was macht die folgende Funktion, bei Übergabe der Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ -1 & 3 & 2 \\ 4 & 5 & 6 \end{pmatrix}$. Geben Sie tabellarisch wieder, welchen Wert die Variablen zum angegebenen Zeitpunkt haben. Welche Funktionalität wird durch die Funktion bereitgestellt?

```
double function(Matrix* A) {
    int n = getMatrixDimension(A);
    double Aij = 0;
    double foo = 0;
    double bar = 0;
    int i = 0;
    int j = 0;
    for (i=0; i<n; ++i) {
        foo = 0;
        for (j=0; j<n; ++j) {
            Aij = getMatrixCoeff(A,i,j);
            foo = foo + fabs(Aij);
        }
        if (foo > bar) {
            bar = foo;
        }
        /* Wert der Variablen zu diesem Zeitpunkt? */
    }
    return bar;
}
```

Verlängern Sie die folgende Tabelle geeignet und füllen Sie sie aus.

i	j	Aij	foo	bar

Aufgabe 8 (3 Punkte). Was versteht man unter *Aufwand einer Funktion*? Geben Sie den Aufwand der Funktion `function` aus der vorausgegangenen Aufgabe in Landau-Notation an und begründen Sie ihre Aussage! Was bedeutet dieser Aufwand für die Laufzeit?

Aufgabe 9 (7 Punkte). Gegeben Sei eine Vektor-Struktur `Vector` und die zugehörigen Funktionen `newVector`, `delVector`, `getVectorLength`, `getVectorCoeff`, `setVectorCoeff`, die Sie *nicht* ausprogrammieren müssen. Sie können also davon ausgehen, dass Sie diese Funktionen einfach zur Verfügung haben. Schreiben Sie eine Funktion `solveU`, die eine Matrix $A \in \mathbb{R}^{n \times n}$ und einen Vektor $b \in \mathbb{R}^n$ übernimmt. Sofern A eine obere Dreiecksmatrix ist, soll die Lösung $x \in \mathbb{R}^n$ von $Ax = b$ berechnet und im Vektorformat zurückgegeben werden. Anderenfalls werde `NULL` zurückgegeben.

Hinweis: Um den Algorithmus herzuleiten, löse man die Matrix-Vektor-Multiplikation mit einer oberen Dreiecksmatrix

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

nach den x_j auf.