OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Übungsaufgaben Blatt 09 - Speicherklassen, Sichtbarkeit, Speicherallokation

PG 1 - Einführung in die Programmierung mit C - Prof. Dr. Ruben Jubeh

---

Aufgabe 1 : Sichtbarkeit 1

Geben Sie an, was das folgende Codebeispiel ausgibt. Begründen Sie, warum die Ausgabe so erfolgt.

```c
#include <stdio.h>

int x = 1234;
double y = 1.234567;

void functionOne() {
   printf("From functionOne:\n  x=%d, y=%f\n", x, y);
}

int main() {
   int x = 4321;

   functionOne();
   printf("Within the main block:\n  x=%d, y=%f\n", x, y);
   /* a nested block */
   {
      double y = 7.654321;
      functionOne();
      printf("Within the nested block:\n  x=%d, y=%f\n", x, y);
   }
   return 0;
}
```

Aufgabe 2 : Sichtbarkeit 2

Geben Sie an, was das folgende Codebeispiel ausgibt. Begründen Sie, warum die Ausgabe so erfolgt.

```c
#include <stdio.h>

int main() {
   int i = 32;

   printf("Within the outer block: i=%d\n", i);

   {
     int i, j;
```

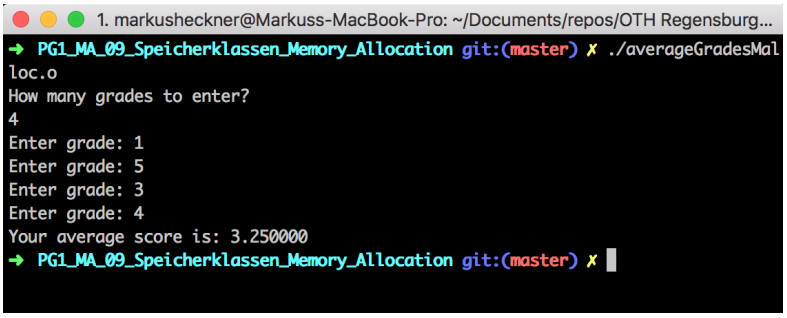
```
10
11      printf("Within the inner block:\n");
12      for (i=0, j=3; i<=3; i++, j--)
13          printf("i=%d, j=%d\n", i, j);
14    }
15    printf("Within the outer block:  i=%d\n", i);
16    return 0;
17  }
```

Aufgabe 3 : **Durchschnittsnoten berechnen mit dynamischer Speicherallokation**

In dieser Aufgabe sollen Sie ein Programm erstellen, das eine Durchschnittsnote berechnen soll. Die Noten sollen Sie vom Nutzer einlesen. Bevor der Nutzer die Noten eingibt, fragt das Programm nach der Anzahl der Noten, aus denen der Durchschnitt berechnet werden soll.

Verwenden Sie zur Ablage der Noten eine dynamische Speicherstruktur. Sie müssen die Größe mithilfe dynamischer Speicherallokation festlegen, nachdem der Nutzer die Anzahl der Noten eingegeben hat. Verschwenden Sie in dieser Aufgabe keinen unnötigen Speicherplatz!

Die folgende Abbildung zeigt eine mögliche Ausgabe des Programms:



Aufgabe 4 : **NumberDelimiter**

Beim Schreiben von besonders großen Zahlen ist es üblich ein Tausender-Trennzeichen zu benutzen, um die Ziffernfolge in Dreiergruppen zu unterteilen. Dieses Trennzeichen kann ein Leerzeichen, ein Komma oder ein Punkt sein.

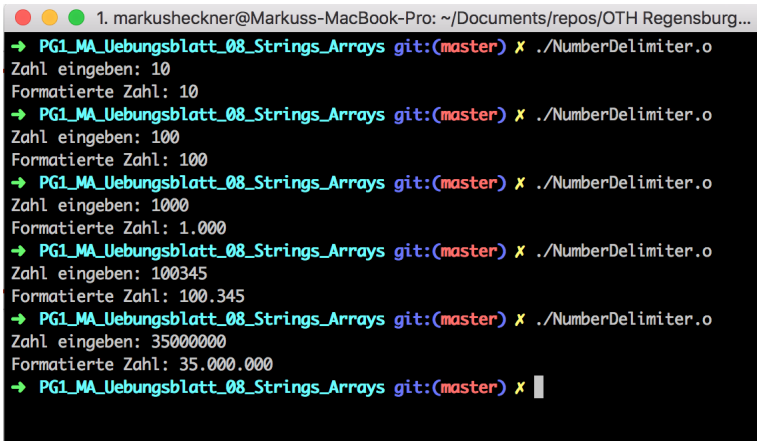
Eine Million würde zum Beispiel folgendermaßen geschrieben:

1,000,000
1.000.000
1 000 000

Um die Darstellung solcher Zahlen zu vereinfachen sollen Sie nun eine Funktion `addSeparatorToNumericString(char number[], char formattedNumber[], char separator)` implementieren, die den String aus `char number[]` Zeichen für Zeichen prüft und in `char formattedNumber[]` den neuen korrekt formatierten String speichert. Achtung: Diese Aufgabe ist möglicherweise komplexer als sie auf den ersten Blick scheint. Unter Umständen müssen Sie eines der beiden Arrays umdrehen, um ein korrektes Ergebnis zu erzeugen.

Tipp: Eventuell kann Ihnen der Debugger gute Dienste bei der Lösung dieser Aufgabe leisten.

Ihr Programm sollte bei korrekter Implementierung folgende Ergebnisse liefern:



Aufgabe 5 : Sudoku Feld

Hinweis:
Verwenden Sie für diese Aufgabe die Datei `SudokuStart.c` und öffnen diese in Codeblocks. Programmieren und testen Sie Ihre Lösung in dieser Datei.

Die Aufgabe in einem Sudoku-Spiel besteht darin eine Matrix aus 9x9 Feldern, die bereits teilweise mit Zahlen zwischen 1 bis 9 befüllt ist, so zu ergänzen, dass in jeder Zeile, jeder Spalte und in allen *Teilfeldern* dieser Matrix, die aus jeweils 3x3 Feldern bestehen, jede Ziffer nur einmal vorkommt.

In dieser Aufgabe sollen Sie ein vereinfachtes Problem lösen, das sich nur mit einem *Teilfeld* (bestehend aus 3x3 Feldern) des Sudoku-Spiels beschäftigt. Die folgenden Abbildung zeigt Beispiele für diese Teilfelder. Lösung (a) ist korrekt, da die Ziffern im Bereich 1 bis 9 liegen und nur einmal vorkommen, Lösung (b) ist inkorrekt, da bestimmte Zahlen doppelt vorkommen (hier: 1 und 8), Lösung (c) ist inkorrekt, da die Zahlen -1 und 10 nicht dem zulässigen Wertebereich entsprechen.

1	2	3	1	1	3	10	2	3
7	8	9	8	8	9	7	-1	9
4	5	6	4	5	6	4	5	6

(a) korrekt | (b) inkorrekt - wiederholende Ziffern | (c) inkorrekt - ungültige Ziffern

Abbildung 1: Beispiele ausgefüllter Sudokus

Implementieren Sie die folgende Funktion:

```
1 int checkSudokuSquare(int sudokuSquare[][3])
```

Diese Methode erhält als Parameter ein zwei-dimensionales Array aus Ganzzahlen, mit drei Spalten und drei Zeilen, das bereits mit Ziffern aus einem Sudoku-Lösungsversuch befüllt ist. Diese Methode gibt `1` zurück, wenn das übergebene Array jede Ziffer zwischen 1 und 9 genau einmal enthält. Die Methode gibt `0` zurück, wenn die Ziffern außerhalb dieses Bereichs liegen oder wenn das Array Duplikate (d.h. identische Ziffern) enthält.