1 Pointer als Funktionsparameter

```c
struct point_t {
    double x;
    double y;
};
```

Gegeben sei die Struktur `point_t`. Schreiben Sie eine Funktion, die zwei Pointer zu Punkten als Parameter akzeptiert und den Abstand zwischen den Punkten berechnet.

```c
double distance (point_t *p1, point_t *p2);
```

2 Pointer- und Variablenadressen

Bewertete Aufgabe!

2.1 Pass By Reference

C ist eine Sprache, die grundsätzlich immer nur die Werte von Variablen an eine aufgerufene Funktion übergibt, die diese Werte dann in eigenen lokalen Variablen speichert (pass by value). Damit haben Funktionen grundsätzlich keinen Zugriff auf Variablen der aufrufenden Funktion. Es sei denn, dieser Zugriff wird ihnen von der aufrufenden Funktion explizit eingeräumt. Z.B. um der Funktion die Rückgabe mehrerer Ergebnisse zu ermöglichen (pass by reference).
Schreiben Sie eine Funktion `divide`, die nicht nur das Ergebnis der Integerdivision zurückgibt, sondern auch den zugehörigen Rest in Form eines Rückgabeparameters zurückgibt.
Füllen Sie dafür die Lücken in folgendem Code aus:

```c
int divide(/*TODO*/) { /*TODO*/ }

int main() {
    int b = 53;
    int a = 7*b + 42;
    int rest = 0;
    int c = divide(/*TODO*/);
    assert(c == 7);
    assert(rest == 42);
}
```

2.2 Pass By Reference, Mehrfache Indirektion

Wie in der vorigen Aufgabe sollen Sie hier eine Funktion schreiben, die mehrere Werte zurückgibt. In diesem Fall ist der zurückzugebende Wert allerdings selbst ein Pointer: Die Funktion soll einen Pointer auf einen C-String entgegennehmen, und das erste Vorkommen eines gegebenen Buchstabens bestimmen. Der Rückgabewert der Funktion ist die Position des gefundenen Buchstabens im String, und optional soll die Funktion über einen Rückgabeparameter auch einen Pointer auf den gefundenen Buchstaben zurückgeben können. Wird der Buchstabe nicht gefunden, so soll -1 zurückgegeben werden und, falls vorhanden, der Rückgabeparameter auf NULL gesetzt werden.
Füllen Sie dafür die Lücken in folgendem Code aus:

```
1  int findCharPosition(char* string, char searchCharacter, /*TODO*/
   ↪ out_pointer) {
2      /*TODO*/
3  }
4
5  int main() {
6      char* string1 = ”foo”;
7      char* string2 = ”bananas and ananas”;
8
9      int result = findCharPosition(string1, 'a', NULL);
10     assert(result == -1);
11
12     result = findCharPosition(string2, ' ', NULL);
13     assert(result == 7);
14
15     char* charPosition;
16     result = findCharPosition(string2, 'd', /*TODO*/);
17     assert(result == 10);
18     assert(charPosition);
19     assert(*charPosition == 'd');
20     assert(charPosition == string2 + result);
21
22     result = findCharPosition(string1, 'a', /*TODO*/);
23     assert(result == -1);
24     assert(!charPosition);
25 }
```

3 Iterieren mit Pointern

Schreiben Sie eine Funktion, die, gegeben einen String der Länge l, l Zeilen ausgibt. Die erste Zeile soll den gesammten String enthalten, die zweite Zeile soll den gesammtem String ohne den ersten Buchstaben enthalten. Bei der dritten Zeile sollen die ersten beiden Zeichen weggelassen werden. Und so weiter bis in der l-ten Zeile nur noch der letzte Buchstabe des Strings steht. Der Aufruf `textTriangle("Abrakadabra")` soll also die Ausgabe

```
Abrakadabra
brakadabra
rakadabra
akadabra
kadabra
adabra
dabra
abra
bra
ra
a
```

produzieren.
Es dürfen dabei keine Indexvariablen verwendet werden, oder Kopien des Strings im Speicher angelegt werden. Die Aufgabe ist ausschließlich mit Pointerarithmetik und `puts()` oder `printf()` Aufrufen zu lösen. Es ist noch nicht einmal ein `strlen()` Aufruf notwendig.

4 Implementation der Summary-Funktion

Schreiben Sie eine Funktion `summary`, die Minimum, Maximum und Mittelwert von einem **`int`**-Array berechnet.

```
1 void summary (int *arr, size_t size, int *max, int *min, int *mean);
```

Berechnen Sie die Statistiken für die folgenden Arrays:

```
1 int cars_speed[50] = {4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12,
     ↪ 12, 13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 16, 16, 17, 17,
     ↪ 17, 18, 18, 18, 18, 19, 19, 19, 20, 20, 20, 20, 20, 22, 23, 24,
     ↪ 24, 24, 24, 25};
2
3 int car_dist[50] = {2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20,
     ↪ 24, 28, 26, 34, 34, 46, 26, 36, 60, 80, 20, 26, 54, 32, 40, 32,
     ↪ 40, 50, 42, 56, 76, 84, 36, 46, 68, 32, 48, 52, 56, 64, 66, 54,
     ↪ 70, 92, 93, 120, 85};
```

Hier sind die richtigen Werte zur Kontrolle:

```
1 > summary(cars)
2         speed           dist
3 Min.   : 4.0   Min.   :  2.00
4 1st Qu.:12.0   1st Qu.: 26.00
5 Median :15.0   Median : 36.00
6 Mean   :15.4   Mean   : 42.98
7 3rd Qu.:19.0   3rd Qu.: 56.00
8 Max.   :25.0   Max.   :120.00
```