Zeiger und dynamische Speicherverwaltung

Die Projekte dieser Übung sollen in der Sprache C (nicht C++) auf Unix erstellt werden. Ein- und Ausgaben erfolgen nur über `stdin` und `stdout` (*Terminalprogramm*).

Lesen Sie den Kommentar des gegebenen Programms `demo.c`, übersetzen und starten Sie es.

Mit Ausnahme des Projekts **laenge**, sollen alle Aufgaben dieser Laborübung **ausschließlich mit Zeigern und Heapobjekten** gelöst werden, d.h. alle Werte werden in namenlosen Heapobjekten gespeichert. Nur Zeiger sollen auf dem Stack angelegt werden.

Aufgabe 1: **spargelmops**

Körpergewicht (kg) und Körpergröße (m) werden eingelesen, dann wird daraus der body mass index berechnet (bmi = Körpergewicht / Körpergröße^2). Für bmi-Werte unterhalb 22,5 lautet die Ausgabe *Spargel*, sonst *Mops*. So sieht es dann aus:

```
Gewicht in kg eingeben: 45
Koerpergroesse in m eingeben: 1.95

Spargel
```

Legen Sie für den Schwellwert 22,5 eine Konstante im Heap an, auf die ein konstanter Zeiger zeigt.

Geben Sie den belegten Speicher frei - obwohl es hier nicht erforderlich ist.

Aufgabe 2: **laenge**

Gegeben sind folgende Deklarationen:

```
int iFeld[2] = { 1, 2 };
double dFeld[2] = { 1.0, 2.0 };
```

Geben Sie die Adressen und Werte der vier Feldelemente aus, ohne den Indexoperator `[ ]` zu verwenden.
An den Adresswerten sieht man, dass die Feldelemente nacheinander im Speicher abgelegt werden. Dies kann man nutzen, um die Bytelänge eines Feldelements zu berechnen.

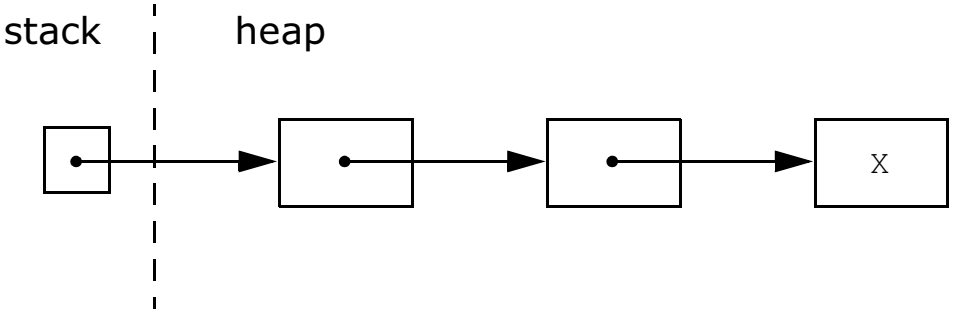
Berechnen Sie also die Länge (in Byte) eines `double`- und eines `int`-Speicherplatzes, ohne die Funktion `sizeof()` zu verwenden.

Aufgabe 3: dreiImHeap

Legen Sie ein Stackobjekt und drei Heapobjekte so an, dass das Stackobjekt auf das erste Heapobjekt, das erste Heapobjekt auf das zweite Heapobjekt und das zweite Heapobjekt auf das dritte Heapobjekt zeigt. Das dritte Heapobjekt bekommt den `char`-Wert *X* zugewiesen. Geben Sie die Werte und die Adressen der vier Objekte aus.

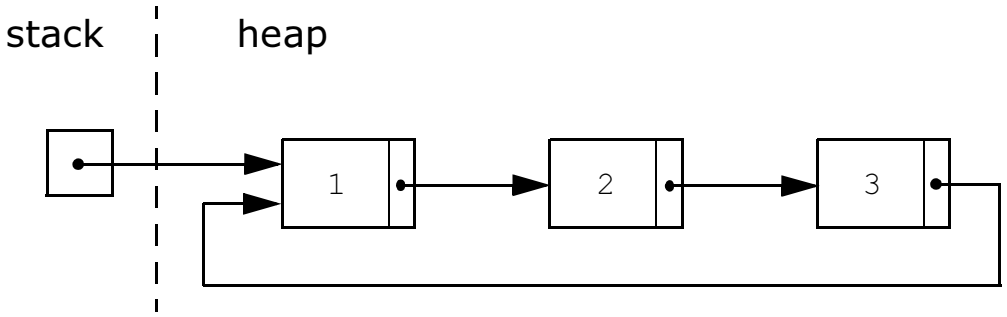
Hinweise:

- **Definieren Sie zuerst die Typen der vier Objekte.**
- Offenbar benötigt man „Zeiger auf Zeiger auf Zeiger ... Typen“. Bei der Deklaration und auch bei der Dereferenzierung wird der * einfach mehrfach verwendet.



Aufgabe 4: einsZweiDreiImHeap

Legen Sie ein Stackobjekt und drei Heapobjekte so an, dass zunächst folgende Datenstruktur entsteht:



Initialisieren Sie die Heapelemente mit den `int`-Werten 1, 2 und 3.

Ändern Sie nun das Stackobjekt, sodass es auf das Heapelement mit dem Inhalt 2 verweist.

Greifen Sie in diesem Zustand auf den Wert des ursprünglich ersten Heapelements zu und geben Sie ihn aus, d.h. es wird 1 ausgegeben.