Exercise 6 - Simple Stack

Basierend auf unserem `struct Element` *vom letzten Übungsblatt, werden wir diese Woche eine Stack-ähnliche Struktur aufbauen*

Bei Fragen oder Problemen bitte in den Moodle schreiben!

Exercise 1 — Grundbausteine

Schreiben Sie für diese Aufgabe Ihre Methoden in die `Stack.cpp`*-Datei und deklarieren Sie die Methoden in* `Stack.h`.
Testen Sie Ihren Code wie üblich in `main.cpp`.

Um sich in das Thema einzuarbeiten, schauen Sie sich die `Stack.h`*-Datei an. In dieser findet Sie die* `class Stack` *mit* `private` *Feldern der Klasse und einer Deklarationd des* `struct Element`.

a) Schreiben Sie die Methode `void push(int value)`, die ein neues `Element` erzeugt, in dem die übergebene `value` gespeichert wird. Anschließend soll das neu erzeugte Element auf unseren Stack gepusht werden, es soll also `head` auf das neu erzeugte Element zeigen und der `next` Pointer des neuen `Element` soll auf den Wert zeigen, auf den `head` vorher zeigte. Vergessen Sie nicht, die `length` Ihres Stacks anzupassen.
Tipp: `new Element{value}` *erzeugt ein neues Element mit der übergebenen* `value`. \

b) Vervollständigen Sie die Methode `void pop()`, die Sie in der `Stack.cpp`-Datei finden können. Die Methode soll das aktuelle `head Element` vollständig aus dem `Stack` entfernen. Dazu muss das vorherige `second Element` des `Stack` nun in `head` referenziert werden, aber gleichzeitig auch das alte `Element` gelöscht werden. Vergessen Sie nicht, die `length` Ihres Stacks anzupassen.
Tipp: `delete <object-pointer>` `befreit (engl. free)` *das Objekt vom* `heap` *und hinterlässt nur den Pointer auf ein ungültiges Objekt.*

c) Schreiben Sie die Methode `int size()`, die die `length` des `Stack` zurückgibt.

Exercise 2 — Nützliche Funktionalitäten

Schreiben Sie für diese Aufgabe Ihre Methoden in die `Stack.cpp`*-Datei. Deklarieren Sie Ihre Funktionen in* `Stack.h`.
Testen Sie Ihren Code wie üblich in der `main.cpp`*-Datei.*

a) Schreiben Sie die Methode `void print()`, die über den ganzen `Stack` iteriert und dabei die `value` jedes Elements in der Konsole ausgibt. *Die Formatierung soll wie folgt aussehen:*

```
Stack my_Stack;
my_Stack.push(3);
my_Stack.push(2);
```

```
my_Stack.push(1);
my_Stack.print();
```

Output: [1,2,3]

b) Schreiben Sie die Methode `Stack primeStack(int upper_bound)`, die einen `Stack` zurück gibt, der mit allen Primzahlen kleiner der `upper_bound` gefühlt ist.

Sie können entweder Ihre eigene `isPrime Funktion` *benutzen oder unsere Implementation aus* `main.cpp`

c) Schreiben Sie abschließend den `deconstructor: ~Stack()`, der erneut die `delete <object-pointer>` Funktion benutzt. Die Methode soll über den gesamten `Stack` iterieren und dabei alle `Elemente` vom `heap` befreien.

Vielleicht geben Sie eine Nachricht in die Konsole aus, wenn Ihr `deconstructor` *aufgerufen wird, um zu sehen wann dieser automatisch aufgerufen wird.*