Technische Universität München
Fakultät für Informatik, I-14
Prof. Harald Räcke
Robin Münk

SoSe 2025
Übungsblatt 1
25. April 2025

Algorithmen und Datenstrukturen

Aufgabe 1 (P) Setup TestSuite [Optional]

Für die begleitenden Programmieraufgaben wird ein automatisches Testing Framework gestellt, mit dem die Aufgaben selbständig getestet werden können. In dieser Aufgabe werden wir es einrichten und uns ein bisschen damit vertraut machen.

a) Befolgen Sie die Anweisungen in der Datei `Setup Programmieraufgaben` in Moodle, um das Testing Framework für ihre IDE einzurichten. Für die Einrichtung kann CLion, VSCode oder ein beliebiger Editor verwendet werden.

b) Für die Programmieraufgaben einer Woche gibt es in Moodle eine Datei `blattX.zip`, welche die gesamte Ordnerstruktur enthält (siehe Erklärung in `Setup Programmieraufgaben`). Die Aufgaben jeder Woche können also komplett unabhängig voneinander bearbeitet werden. Laden Sie `blatt0.zip` herunter, entpacken Sie den Ordner und öffnen Sie ihn in Ihrer IDE.

c) Implementieren Sie die folgenden simplen Funktionen und überprüfen Sie die Implementierung auf Richtigkeit mit dem Testing Framework. Bearbeiten Sie dafür die Datei `blatt0.cpp` in dem Unterordner `exercises`. Beachten Sie bei fehlgeschlagenen Tests stets den Output - er kann bei der Fehlersuche helfen!

   (i) **ReturnTrue.** Gibt `true` zurück.

   (ii) **ReturnPositiveOdd.** Gibt eine positive, ungerade Zahl zurück.

   (iii) **ReturnPrime.** Gibt eine *zweistellige* Primzahl zurück.

Aufgabe 2 (P) C++ Programmieraufgaben

Die Programmieraufgaben können mit dem automatischen Testing Framework überprüft werden.

a) **Reverse.** Implementieren Sie eine Funktion, die die Reihenfolge der Zahlen in einem gegebenen `vector<int>` invertiert.

   Bsp.: [1, 4, -7, 19] → [19, -7, 4, 1]

b) **PlusOne.** Gegeben ist ein `vector<int> digits`, der einen positiven Integer repräsentiert. Ziffer *i* ist dabei an Stelle `digits[i]`, geordnet von höchst- zu niedrigstwertigem Bit von Anfang bis Ende der Liste. Es gibt keine führenden Nullen.

   Inkrementieren Sie die repräsentierte Zahl und geben Sie den zugehörigen `vector<int>` zurück.

   Bsp.: [3, 1, 2, 6] → [3, 1, 2, 7]

c) **FindPeaks.** Gegeben ist ein `vector<int> mountain`, der die Skyline eines Berges repräsentiert. Finden Sie alle Gipfel dieses Berges und geben Sie ihre Indices in einem `vector<int>` sortiert zurück.

Ein Gipfel is ein Element, das strikt größer ist, als beide seiner Nachbarn und nicht am Rand des Berges liegt.

Bsp.: [5, 1, 4, 3, 8, 5] $\rightarrow$ [2, 4]