Assignment 3 - Permutations

In dieser Aufgabe sollst Du den Umgang mit Pointern üben. Lies mit `std::cin` eine beliebige Anzahl von `double`s in ein dynamisch allokiertes Array ein. Dabei musst du eventuell die Größe des Arrays zur Laufzeit ändern, wenn es voll ist. Sobald der Benutzer etwas eingibt, was keine Zahl ist, wird die Eingabe beendet.

Nach dem erfolgreichen Einlesen des dynamischen Arrays sollst Du auch damit arbeiten. Lies ähnlich wie im ersten Schritt genau so viele Indizes (`int`) ein, wie der Benuzter Werte eingegeben hat. Diese Indizes sollen eine neue Reihenfolge (Permutation) der Daten beschreiben.

Überprüfe zunächst, ob die eingegebenen Indizes eine korrekte Permutation beschreiben. Implementiere dazu die Funktion `bool isPermutation(int* perm, int count)` und rufe sie auf. Die Funktion ist in `utils.hpp` deklariert und soll in `utils.cpp` definiert werden.

Wenn die Permutation korrekt ist, sollst Du überprüfen, ob eine Umsortierung der Daten des dynamischen Arrays mit der angegebenen Permutation eine aufsteigend sortierte Folge ergibt. Implementiere dazu die Funktion `bool isSorted(double* data, int count, int* perm)`, die ebenfalls in `utils.hpp` deklariert ist. Diese Funktion soll die Daten der eingegebenen Arrays nicht verändern und nicht kopieren!

Hinweise

- Für eine gültige Permutation muss jeder Index zwischen 0 und N-1 genau einmal vorkommen.
- Um herauszufinden, ob etwas eingegeben wurde, das keine Zahl ist, kannst Du die Funktion `std::cin.fail()` verwenden. (`std::cin` ist ein Input Stream. Weitere Informationen dazu findest Du hier)
- Da Du danach weiter mit `std::cin` arbeiten willst, musst Du es mit Hilfe von `std::cin.clear()` resetten und mit `std::cin.ignore()` das falsche Zeichen überspringen
- Du kannst so viele zusätzliche Funktionen schreiben, wie Du benötigst.

Optional

Schau Dir die folgenden Regeln aus den C++ Core Guidelines an. Wir halten uns in dieser Übung an viele der vorgeschlagenen Regeln noch nicht, weil wir die notwendigen Sprachfeatures noch nicht kennen, um es besser zu machen.

- I.13: Do not pass an array as a single pointer
- F.42: Return a T* to indicate a position (only)
- F.43: Never (directly or indirectly) return a pointer or a reference to a local object
- ES.47: Use nullptr rather than 0 or NULL

- ES.65: Don't dereference an invalid pointer
- ES.106: Don't try to avoid negative values by using unsigned