Aufgabe 2

Aufgabe 2: Objektorientierte Programmierung [25 Punkte]

Sie wollen ein Programm erstellen, um Geocaches zu verwalten. Geocaches sind eine Art von moderner Schnitzeljagd.

Für mehr Informationen siehe den Wikipedia-Artikel dazu: https://de.wikipedia.org/wiki/Geocaching

Ihr Programm soll dazu aus zwei Klassen bestehen:

- `Station`: Diese Klasse enthält eine Geokoordinate bestehend aus Längen- und Breitengrad
- `Geocache`: Diese Klasse beschreibt einen einzelnen Geocache. Ein Geocache kann aus einer oder mehreren Geokoordinaten bestehen. Bei mehreren Koordinaten besitzt ein Geocache mehrere Stationen, die der Reihe nach abgelaufen werden müssen. Wenn ein Geocache mehr als eine Koordinate hat, ist dies ein *Multi*.

Das UML-Klassendiagramm liegt als zusätzliche Datei in diesem Ordner. Nicht über das grüne C neben dem Klassennamen wundern. Ich konnte hier aus technischen Gründen nicht yEd verwenden.

Die Klasse `Station` [12P]

Um den Geocache zu verorten, entwerfen Sie zuerst eine Klasse `Station`.

- Die Felder `lat` (Breitengrad) und `lon` (Längengrad) speichern die jeweilige Koordinate
- Zusätzlich soll der Koordinate ein Name gegeben werden können. Dafür ist das Feld `name` da.

Die Klasse verfügt über zwei Konstruktoren:

- Der Standardkonstruktor verortet die Koordinate bei `(0, 0)` und setzt den Namen auf `Null Island` (https://de.wikipedia.org/wiki/Null_Island).
- Der zweite Konstruktor akzeptiert `name`, `lat` und `lon` und weist diese Werte direkt den Feldern zu. Dieser Konstruktor soll eine Ausnahme (`std::runtime_error`) auslösen, wenn die Koordinaten ungültig sind. `lat` muss im Bereich `[-90, 90]` liegen und `lon` im Bereich `[-180, 180]`.

Die privaten Felder verfolgen über die folgenden Getter/Setter:

- `name` hat Getter und Setter
- `lat` hat einen Getter
- `lon` hat einen Getter

Abschließend soll noch von `calc_distance` die Distanz zwischen zwei Geokoordinaten berechnet und zurückgegeben werde. Da die Erde keine Scheibe ist, ist die Berechnung von Distanzen kompliziert. Ein Freund hat Ihnen die Berechnung netterweise programmiert. Diese befindet sich in `distance.h`. Rufen Sie diese einfach mit den richtigen Parametern auf.

Die Klasse `Geocache` [9P]

Ein Geocache wird über dessen Koordinaten (`stations`), die Geländewertung (`terrain`) und die Größe (`size`) spezifiziert.

- Der `std::vector stations` enthält die Koordinaten bzw. Stationen des Geocaches. Diese Stationen sind in der Reihenfolge angeordnet, wie sie angelaufen werden müssen.
- `terrain` beschreibt numerisch die Schwierigkeit des Terrains, um den Cache zu erreichen
- `size` beschreibt numerisch die Größe des Geocaches (also des Behälters, der zu suchen ist)

Die Klasse verfügt über einen Konstruktor. Dieser akzeptiert `terrain` und `size`. Diese Werte haben eine untere Grenze von 0 und eine obere Grenze: Wenn `terrain` größer als 5 ist, wird es auf 5 gesetzt. Wenn `size` größer als 3 ist, wird es auf 3 gesetzt. Anschließend werden die Werte den passenden Feldern der Klasse zugewiesen.

Zudem verfügt die Klasse über drei Methoden:

- Die Methode `calc_difficulty` gibt den Schwierigkeitsgrad zurück. Dieser wird nach folgender Formel berechnet: `size * 4 - terrain`.
- Mit `is_multi` kann ermittelt werden, ob der Geocache ein sogenannter *Multi* ist. Ein Geocache ist ein Multi, wenn er über mindestens zwei Koordinaten verfügt.
- Mit `calc_length` wird die aufsummierte Distanz zwischen allen Stationen bzw. Koordinaten ermittelt.

Testen (in `main`) [4P]

Erzeugen Sie eine Instanz von `Geocache` mit `terrain` 4 und `size` 2.

Fügen Sie dann der Klasse die folgenden zwei Wegpunkte (`Station`) hinzu:

- `A` bei `(U, V)`
- `B` bei `(X, Y)`

Die Werte sind folgendermaßen:

- `A`: Ihr Vorname
- `B`: Ihr Nachname
- `U`: Zweite Ziffer Ihrer Matrikelnummer
- `V`: Dritte Ziffer Ihrer Matrikelnummer
- `X`: Vierte Ziffer Ihrer Matrikelnummer
- `Y`: Fünfte Ziffer Ihrer Matrikelnummer

Geben Sie abschließend den Schwierigkeitsgrad auf der Konsole aus.