IMS
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover

Programmierpraktikum Technische Informatik (C++)
Aufgabe 3

Hinweise
Abgabe: Stand des Git-Repositories am 12.5.2020 um 9 Uhr.
Die Dateien zur Bearbeitung dieser Aufgabe erhalten Sie, indem Sie die neue Aufgabe aus dem Aufgabenrepository in Ihr lokales mergen. Dies geschieht mit `git pull cpp2020 master` innerhalb Ihres Repositories. Die Lösungen committen Sie bitte in Ihr lokales Repository (`git commit -a` oder `git add` gefolgt von `git commit`) und pushen sie in Ihr Repository auf dem git-Server des Instituts (`git push`).

Teilaufgabe 1 (3 Punkte)

Um eine geheime Nachricht zu codieren, wird in dieser Aufgabe eine Variante der Caesar-Verschlüsselung benutzt. Anstatt dass jedes Zeichen um die gleiche Distanz verschoben wird, wird jedes Zeichen um eine zufällige Weite verschoben. Die Weiten werden dabei für jedes Zeichen in einer `std::map` vermerkt.

a) Implementieren Sie in der Datei `decipherer.cpp` die Funktion `std::string decipherMessage(const std::string& codedMessage, const std::map<char,char>& cipher)`! Diese Funktion soll den übergebenen String mit Hilfe der übergebenen `std::map` decodieren und danach zurückgeben.

Hinweise:

- Die übergebene Map enthält alle in dem String vorkommenden Zeichen als Schlüssel.
- Die Werte, die mit den Schlüsseln assoziiert sind, liegen ebenfalls als `char` vor.
- Die Decodierung ist jedoch nicht eine 1-zu-1-Übersetzung zwischen dem Schlüssel und dem Wert!
- Ein einzelner Character wird decodiert, indem die Summe von dem Schlüssel und dem dazugehörigen Wert gebildet wird.
- Um auf einen Wert, der zu einem Schlüssel gehört, zuzugreifen, ist die Methode `at` der `std::map` hilfreich.

Appelstraße 4 | 30167 Hannover
fon 0511 762-19691 | fax 0511 762-19694
www.ims.uni-hannover.de | info@ims.uni-hannover.de

b) Während der Übertragung sind Fehler in der Nachricht erschienen. Es ist davon auszugehen, dass Zeichen, die verfälscht übertragen werden, deutlich häufiger vorkommen als fehlerhafte.

Implementieren Sie in der Datei `decipherer.cpp` die Funktion `std::string removeErrors(const std::string& messageWithErrors)`! Diese Funktion soll das am seltensten erscheinende Zeichen im übergebenen String entfernen und einen korrigierten String zurückgeben.

Hinweise:

- Als erstes sollten alle Zeichen gezählt werden. Eine `std::map`, die die Zeichen als Schlüssel und die Häufigkeit als Werte beinhaltet, kann dafür hilfreich sein.
- Um ein neues Schlüssel-Wert-Paar der Map hinzuzufügen, besitzt die `std::map` die Methode `emplace`.
- Um zu überprüfen, ob die Map bereits einen bestimmten Schlüssel besitzt, gibt es die Methode `find`.
- Als nächstes muss das seltenste Zeichen bestimmt werden.
- Beim Durchlaufen einer `std::map` mit einer Range-based-for-Schleife erhält man die Elemente der Map als `std::pair<KeyType, ValueType>`.
- Auf die Elemente eines `std::pair` kann mit den Methoden `first` und `second` zugegriffen werden.
- Als letztes muss das seltenste Zeichen aus der fehlerhaften Nachricht enfernt werden und der so entstandene String zurückgegeben werden.

Teilaufgabe 2 (2 Punkte)

Ziel der Aufgabe ist es, Eingaben des Spiels Schere, Stein, Papier entgegenzunehmen, auszuwerten und das Ergebnis der Runde auszugeben (s.a. `https://en.wikipedia.org/wiki/Rock_paper_scissors`). Über die Kommandozeile sollen die Eingaben der beiden Spieler übergeben werden. Ein Aufruf soll dabei so aussehen können:

```
$ ./rps rock paper
Player1: rock, Player2: paper, Winner: Player2
```

a) Implementieren Sie die Funktion `Move parseInput(const std::string& input)`! Diese Funktion soll entsprechend den als `std::string` übergebenen Zug als enum `Move` zurückgeben. Wenn keine gültige Eingabe übergeben wurde, soll der Move

`Error` zurückgegeben werden. Als gültige Eingaben gelten `rock, paper und scissors` jeweils nur in Kleinbuchstaben.

Hinweise:

- Die Struktur `Move` ist in der Datei `rps.h` vorgegeben.
- Die Dokumentation von `enum class` können sie unter `https://en.cppreference.com/w/cpp/language/enum` nachlesen.

b) Implementieren sie die Funktion `Result rockPaperScissors(const std::string& p1, const std::string& p2)` und vervollständigen sie die `main`-Funktion derart, dass das Spiel funktionsfähig ist. Die Funktion `rockPaperScissors` bekommt die Eingaben aus der Komandozeile übergeben, wertet diese gemäß der Regeln des Spieles dann aus und gibt das Ergebnis zurück. Dabei soll der Fall eines Siegers, eines Unentschiedens oder fehlerhafter Eingabe ermittelt werden. Beispiel einer fehlerhaften Eingabe:

```
$ ./rps brunnen rock
Invalid Input
```

In der `main`-Funktion soll das Ergebnis dann sinnvoll auf der Standardausgabe `std::cout` ausgegeben werden.

Hinweise:

- Die Kommandozeilenparameter werden in dem `std::vector<std::string> args` gespeichert und können dort auch abgerufen werden.
- In der Funktion `rockPaperScissors` sollen die Parameter zuerst über die Funktion `parseInput` in ein `Move` umgewandelt und diese dann ausgewertet werden.