**Abgabetermin: Mi, 12. 04. 2023**

☐ **Gr. 1,** Dr. S. Wagner **Name** _____ **Aufwand in h** _____
☐ **Gr. 2,** Dr. D. Auer
☐ **Gr. 3,** Dr. G. Kronberger **Punkte** _____ **Kurzzeichen Tutor / Übungsleiter*in** _____ / _____

## 1. (De-)Kompression von Dateien (10 Punkte)

Die sogenannte *Lauflängencodierung* (engl. *run length encoding*, kurz *RLE*) ist eine einfache Datei-Kompressionstechnik, bei der jede Zeichenfolge, die aus mehr als zwei gleichen Zeichen besteht, durch das erste Zeichen und die Länge der Folge codiert wird.

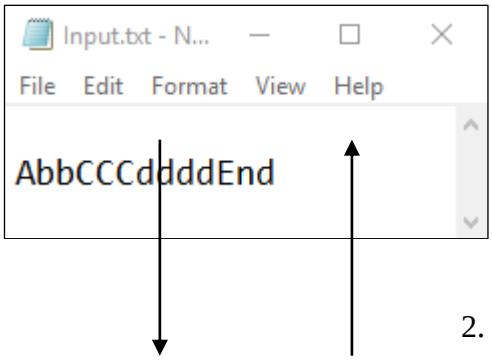
Implementieren Sie eine einfache Variante dieses Verfahrens in Form eines Pascal-Programms *RLE*, welches Textdateien, die nur Groß- und Kleinbuchstaben, Satz- und das Leerzeichen (aber keine Ziffern) enthalten, komprimieren und wieder dekomprimieren kann. Ihr Programm muss folgende Aufrufmöglichkeiten von der Kommandozeile aus bieten (die Metasymbole [...] und ...|... stehen für Option bzw. Alternativen):

```
RLE [ -c | -d ] inFile [outFile]
```

Bedeutung der Parameter:

| | |
|---|---|
| `-c` | die Eingabedatei soll komprimiert werden (Standardannahme), |
| `-d` | die Eingabedatei soll dekomprimiert werden, |
| `inFile` | Name der Eingabedatei und |
| `outFile` | Name der Ausgabedatei, sonst Standardausgabe. |

*Beispiel*:

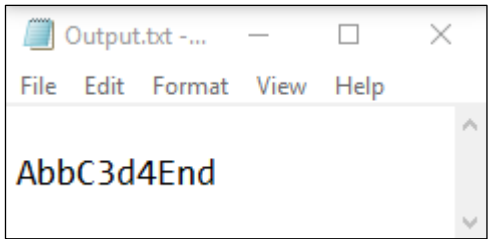



1. Komprimieren mit:

```
RLE -c Input.txt Output.txt
```

2. Dekomprimieren mit:

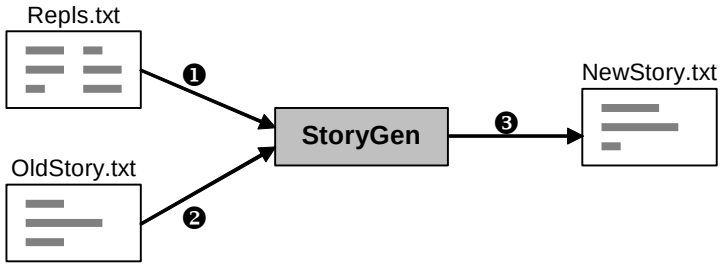
```
RLE -d Input.txt Output.txt
```



## 2. Geschichten vom … (14 Punkte)

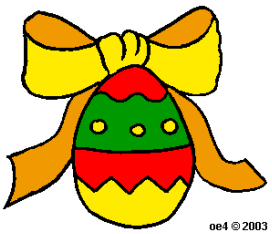
Heutzutage muss ja alles schnell gehen ;-). Wer hat schon noch die Muße, sich der Jahreszeit entsprechende Geschichten für die kleinen und großen Kinder auszudenken. Gesucht ist deshalb ein Pascal-Programm *StoryGen* (als Abkürzung für *story generator*), das es ermöglicht, z. B. aus einer Geschichte vom Osterhasen, die gerade noch aktuell war, schnell eine Geschichte vom Krampus oder vom Christkind zu machen, indem spezielle Wörter ausgetauscht werden (z. B. Osterhase durch Christkind und Ostern durch Weihnachten).

Damit man *StoryGen* flexibel anwenden kann, müssen die durchzuführenden Ersetzungen in einer Textdatei (z. B. *Repls.txt* für *Replacements*) definiert sein und die alte Geschichte in einer zweiten Textdatei (z. B. *OldStory.txt*) stehen. Daraus kann die neue Geschichte in einer dritten Textdatei (z. B. *NewStory.txt*) erzeugt werden, so wie in folgender Abbildung dargestellt:




*StoryGen* muss zuerst die Ersetzungen einlesen und in einer geeigneten Datenstruktur speichern. Dann muss die alte Geschichtendatei zeilenweise gelesen, alle Ersetzungen in der aktuellen Zeile vorgenommen und die geänderte Zeile in die neue Geschichtendatei geschrieben werden. Die Datei mit den Ersetzungen soll beliebig viele Ersetzungen aufnehmen können, in jeder Zeile aber nur eine (mit der Syntax *oldWord newWord*) enthalten. Eine rudimentäre Ersetzungsdatei für den Übergang von Ostern nach Weihnachten könnte dann folgendermaßen aussehen:

```
Osterhase Christkind
Ostern Weihnachten
...
```




*StoryGen* muss von der Kommandozeile mit drei Parametern (den Dateinamen der drei beteiligten Textdateien) versorgt werden, so dass es für obiges Beispiel wie folgt aufgerufen werden kann:

```
StoryGen  Repls.txt  OldStory.txt  NewStory.txt
```

Zum Testen finden Sie in *Ostern.txt* eine Geschichte vom Osterhasen. Machen Sie daraus eine möglichst „gute“ Weihnachtsgeschichte.

### Hinweise:

1. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
2. Dokumentieren und kommentieren Sie Ihre Algorithmen.
3. Bei Programmen: Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Programm funktioniert, und dass es auch in Fehlersituation entsprechend reagiert.