ADF 2*x* & PRO 2*x* **Übungen zu Fortgeschrittenen Algorithmen & Datenstrukturen und OOP** **SS 23, Übung 3**

Abgabetermin: Mi, 12. 04. 2023

☒ **Gr. 1,** Dr. S. Wagner **Name** Elias Wurm **Aufwand in h** 2h 15min
☐ **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.

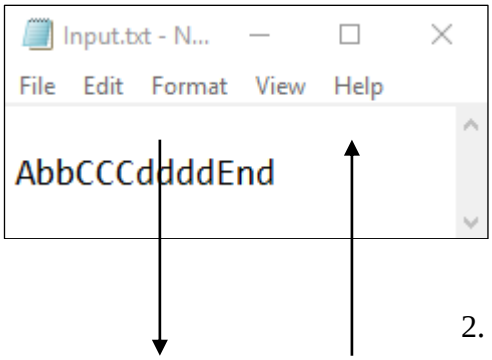
Implementieren Sie eine einfache Variante dieses Verfahrens in Form eines Pascal-Programms *RLE*, welches Textdateien, die nur Groß- und Kleinbuchstaben, Satz- und das Leerzeichen (aber keine Ziffern) enthalten, komprimieren und wieder dekomprimieren kann. Ihr Programm muss folgende Aufrufmöglichkeiten 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:

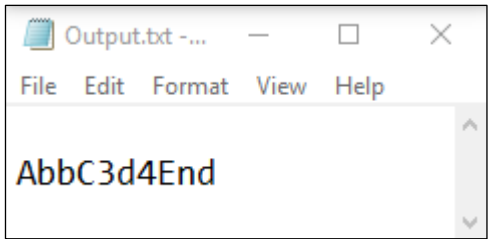


1. Komprimieren mit:

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

2. Dekomprimieren mit:

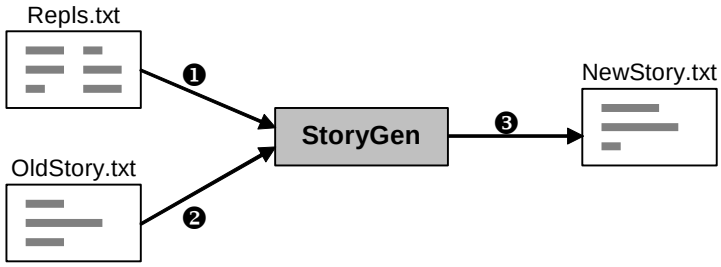
```
RLE -d Input.txt Output.txt
```



2. Geschichten vom … (14 Punkte)

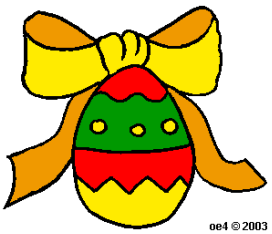
Heutzutage muss ja alles schnell gehen ;-). Wer hat schon noch die Muße, sich der Jahreszeit entsprechende Geschichten für die kleinen und großen Kinder auszudenken. Gesucht ist deshalb ein Pascal-Programm *StoryGen* (als Abkürzung für *story generator*), das es ermöglicht, z. B. aus einer Geschichte 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.

Aufgabe 1 - (De-)Kompression von Dateien

Lösungsidee:
Das inputfile zeilenweise lesen und je nachdem die compress oder decompress funktion auf diese zeile anwenden und die geänderte zeile dann je nachdem auf die console oder in das outputfile schreiben.

Für die compress Funktion gehe ich über alle chars der zeile drüber und falls ein char mehr als 3 mal in folge vorkommt schreib ich statt dem gegebenen die RLE schreibweise. Für decompress das gleiche nur umgekehrt.

Zeitaufwand: 1h 30min

Code:

```
program RLE;

uses SysUtils;

type
  OperationType = (compress, decompress);
  OutputType = (toFile, toConsole);

function CompressString(str: string): string;
var
  i, j: integer;
  currChar: char;
  compressed: string;

  // inner procedure to avoid code duplication because this has to be run
inside the loop and after the loop
  procedure Update;
  begin
    if j > 2 then
      compressed := Concat(compressed, currChar, IntToStr(j))
    else if j = 2 then
      compressed := Concat(compressed, currChar, currChar)
    else
      compressed := Concat(compressed, currChar);
    j := 1;
  end;
begin
  compressed := '';
  j := 1;
  i := 1;
  currChar := str[i];
  for i := 2 to Length(str) do
    if str[i] = currChar then
```

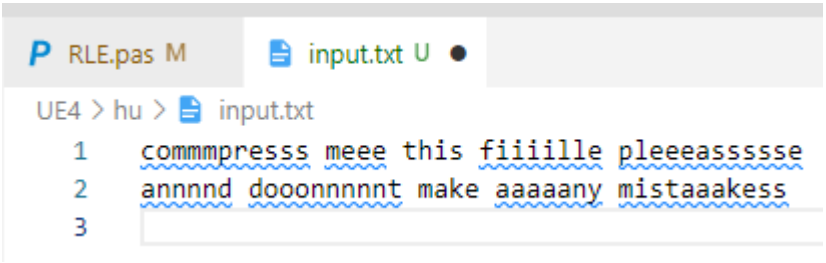
```
      Inc(j)
    else
    begin
      Update();
      currChar := str[i];
    end;

  Update();
  CompressString := compressed;
end;

function DecompressString(str: string): string;
var
  i, j: integer;
  decompressed: string;
begin
  decompressed := '';
  i := 1;
  while i <= Length(str) do
  begin
    if (str[i] in ['0'..'9']) then
    begin
      if(i = 1) then break;
      for j := 2 to StrToInt(str[i]) do
        decompressed := Concat(decompressed, str[i-1]);
    end else decompressed := Concat(decompressed, str[i]);
    Inc(i);
  end;

  DecompressString := decompressed;
end;

procedure RunRLE(operation: OperationType; outputType: OutputType; const
inFileName: string; const outFileName: string);
var
  line: STRING;
  inFile, outFile: TEXT;
begin
  Assign(inFile, inFileName);
  Reset(inFile);
  if(outputType = toFile) then
  begin
    Assign(outFile, outFileName);
    Rewrite(outFile);
  end;

  while(not Eof(inFile)) do
  begin
```

```
    ReadLn(inFile, line);

    if(operation = compress) then
      line := CompressString(line)
    else if(operation = decompress) then
      line := DecompressString(line);

    if(outputType = toFile) then
      writeln(outFile, line)
    else
      writeln(line);
  end;

  Close(inFile);
  if(outputType = toFile) then
    Close(outFile);
end;

var
  Command: string;
  InFileName, OutFileName: string;
  outType: OutputType;
begin
  if ParamCount > 0 then
    Command := ParamStr(1)
  else begin
    write('enter if you want to compress (-c) or decompress (-d) > ');
    ReadLn(Command);
  end;
  if (Command <> '-c') and (Command <> '-d') then
  begin
    WriteLn('Error: Unkowm Command - ', Command);
    writeln;
    Halt(1);
  end;

  if ParamCount > 1 then
    InFileName := ParamStr(2)
  else begin
    write('enter infilename > ');
    ReadLn(InFileName);
  end;
  if not FileExists(InFileName) then
  begin
    WriteLn('Error: input file does not exist - ', InFileName);
    writeln;
    Halt(1);
  end;
```

```
  outType := toFile;
  if ParamCount > 2 then
    OutFileName := ParamStr(3)
  else outType := toConsole;
  if ((outType = toFile) and (inFilename = outFilename)) then
  begin
    WriteLn('Error: output file can not be the same as input file - ',
InFileName);
    writeln;
    Halt(1);
  end;

  if Command = '-c' then
    RunRLE(compress, outType, InFileName, OutFileName)
  else
    RunRLE(decompress, outType, InFileName, OutFileName);
end.
```

Tests:

Inputfile used for those tests:



Test without any params:

```
enter if you want to compress (-c) or decompress (-d) > -c
enter infilename > input.txt
com3pres3 me3 this fi4lle ple3as5e
an4d do3n5t make a5ny mista3kess
Heap dump by heaptrc unit of C:\_data\fh-repos\2023SS_ADF\UE4\hu\RLE.exe
135 memory blocks allocated : 3168/3536
135 memory blocks freed     : 3168/3536
0 unfreed memory blocks : 0
True heap size : 163840 (96 used in System startup)
True free heap : 163744
```

Test compression with output file:

```
C:\_data\fh-repos\2023SS_ADF\UE4\hu>RLE -c input.txt compressed.txt
```

RLE.pas M compressed.txt U

UE4 > hu > compressed.txt

```
1  com3pres3 me3 this fi4lle ple3as5e
2  an4d do3n5t make a5ny mista3kess
3
```

Test decompression:

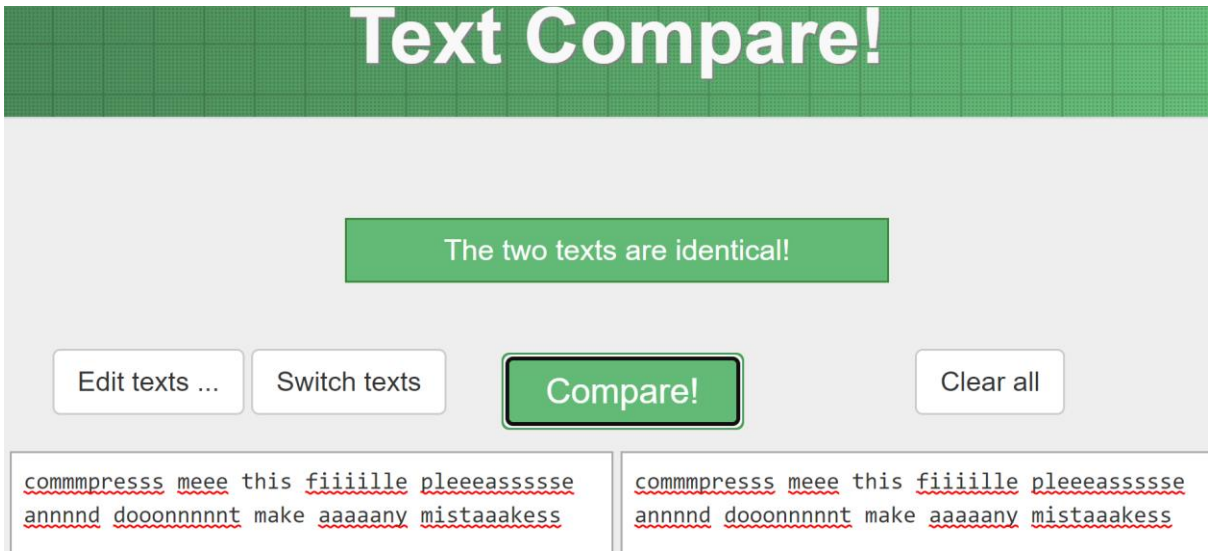
```
C:\_data\fh-repos\2023SS_ADF\UE4\hu>RLE -d compressed.txt decompressed.txt
```

RLE.pas M decompressed.txt U

UE4 > hu > decompressed.txt

```
1  commmpresss meee this fiiiille pleeeasssse
2  annnnd dooonnnnnt make aaaaany mistaaakess
3
```

Test Conclusion: as we can see the compressed and again decompressed file equals the initial input.txt



Src: https://text-compare.com/

Aufgabe 2 - Geschichten vom …

Lösungsidee:
Zuerst das Replacements file zeile für zeile lesen und jede zeile mit split auf die beiden wörter aufteilen und diese in einem record repl mit old- und newword speichern.

Danach zeilenweise über das inputfile iterieren und die funktion StringReplace für jedes repl auf jeder zeile anwenden und die veränderte zeile ins outputfile schreiben.

Ich denke das funktion wie SplitString und ReplaceString wahrscheinlich nicht so gern gesehen sind, aber meiner Meinung nach solange es nicht in der Angabe steht sollte es mir erlaubt sein alle Möglichkeiten der Sprache Pascal zu nutzen. (gleiche gilt für dynamic arrays)

Zeitaufwand: 45min

Code:

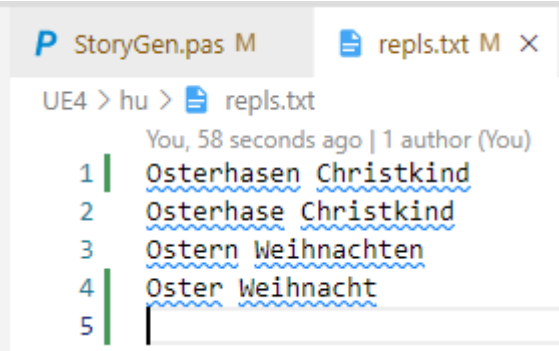
```
program StoryGen;

uses SysUtils, StrUtils;

type
  Repl = record
    OldWord: string;
    NewWord: string;
  end;

procedure CheckIfFileExists(fileName: string);
begin
  if not FileExists(fileName) then
  begin
    WriteLn('Error: file does not exist - ', fileName);
    writeln;
    Halt;
  end;
end;

procedure CheckFilenamesIdentical(file1, file2: string);
begin
  if (file1 = file2) then
  begin
    WriteLn('Error: file can not be the same - ', file1);
    writeln;
    Halt;
  end;
end;
```

```
function GetFilename(paramInt: Integer; msg: string): string;
var
  fileName: string;
begin
  if ParamCount > (paramInt - 1) then
    fileName := ParamStr(paramInt)
  else begin
    write(msg, ' > ');
    ReadLn(fileName);
  end;
  GetFilename := fileName;
end;

var
  repls: array of Repl;

procedure readRepls(fileName: string);
var
  replFile: TEXT;
  line: string;
  words: array of AnsiString;
begin
  assign(replFile, fileName);
  reset(replFile);

  while not eof(replFile) do
  begin
    readln(replFile, line);

    words := SplitString(line, ' ');
    if (not (High(words) = 1)) then
    begin
      WriteLn('Error: incorrect format of replacements file');
      writeln;
      Halt;
    end;

    SetLength(repls, Length(repls) + 1);

    repls[Length(repls) - 1].OldWord := words[0];
    repls[Length(repls) - 1].NewWord := words[1];
  end;

  close(replFile);
end;
```

```
procedure openFiles(var inFile, outFile: TEXT; inFileName, outFileName:
string);
begin
  Assign(inFile, inFileName);
  Reset(inFile);
  Assign(outFile, outFileName);
  Rewrite(outFile);
end;

procedure closeFiles(var inFile, outFile: TEXT);
begin
  Close(inFile);
  Close(outFile);
end;

procedure replaceWords(var line: string);
var
  i: integer;
begin
  for i := 0 to Length(repls) - 1 do
    line := StringReplace(line, repls[i].OldWord, repls[i].NewWord,
[rfReplaceAll, rfIgnoreCase]);
end;

procedure runReplacements(inFileName, outFileName: string);
var
  line: string;
  inFile, outFile: TEXT;
begin
  openFiles(inFile, outFile, inFileName, outFileName);

  while(not Eof(inFile)) do
  begin
    ReadLn(inFile, line);
    replaceWords(line);
    writeln(outFile, line);
  end;

  closeFiles(inFile, outFile);
end;

var
  replsFileName, inFileName, outFileName: string;
begin
  replsFileName := GetFilename(1, 'Enter fileName with the replacements');
  CheckIfFileExists(replsFileName);

  inFileName := GetFilename(2, 'enter text infilename');
```

```
  CheckIfFileExists(inFileName);
  CheckFilenamesIdentical(inFileName, replsFileName);

  outFileName := GetFilename(3, 'enter text outfilename');
  CheckFilenamesIdentical(outFileName, replsFileName);
  CheckFilenamesIdentical(outFileName, inFileName);

  readRepls(replsFileName);
  runReplacements(inFileName, outFileName);
end.
```

Tests:



Natürlich könnte man noch mehr Einträge in der Replacement Datei vornehmen, um die Geschichte authentischer wirken zu lassen, aber denke das ist nicht der Sinn der Übung :D

```
C:\_data\fh-repos\2023SS_ADF\UE4\hu>StoryGen repls.txt Ostern.txt Weihnachten.txt
Heap dump by heaptrc unit of C:\_data\fh-repos\2023SS_ADF\UE4\hu\StoryGen.exe
1651 memory blocks allocated : 82649/88200
1651 memory blocks freed     : 82649/88200
0 unfreed memory blocks : 0
True heap size : 294912 (176 used in System startup)
True free heap : 294736
```