ADF 2*x* & PRO 2*x* Übungen zu Fortgeschrittenen Algorithmen & Datenstrukturen und OOP **SS 21, Übung 3**

Abgabetermin: Mi, 14. 04. 2021

☐ **Gr. 1,** Dr. S. Wagner **Name** Angelos Angelis **Aufwand in h** 12

☒ **Gr. 2,** Dr. D. Auer

☐ **Gr. 3,** Dr. G. Kronberger **Punkte** _____ **Kurzzeichen Tutor / Übungsleiter*in** _____ / _____

1. (De-)Kompression von Dateien (12 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.

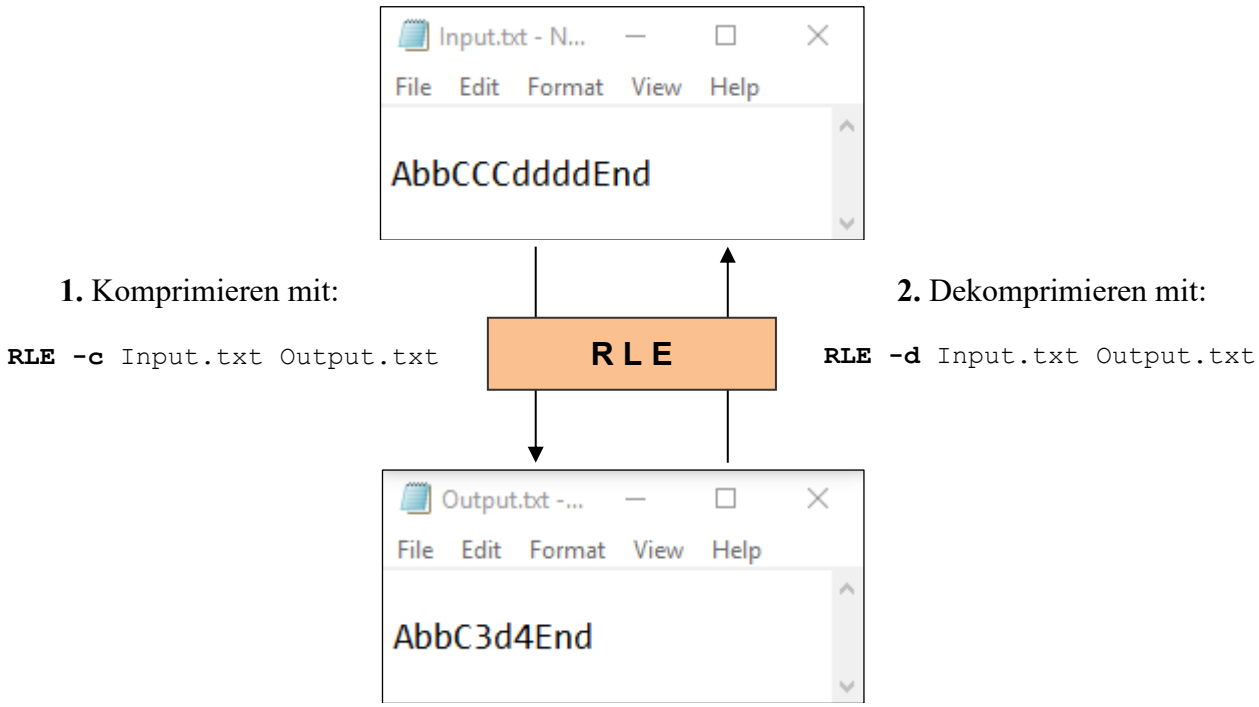
Implementieren Sie eine einfache Variante dieses Verfahrens in Form eines Pascal-Programms *RLE*, welches Textdateien, die nur Groß- und Kleinbuchstaben, Satz- und das Leerzeichen (aber keine Ziffern) enthalten, komprimieren und wieder dekomprimieren kann. Ihr Programm muss folgende Aufrufmöglichkeiten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

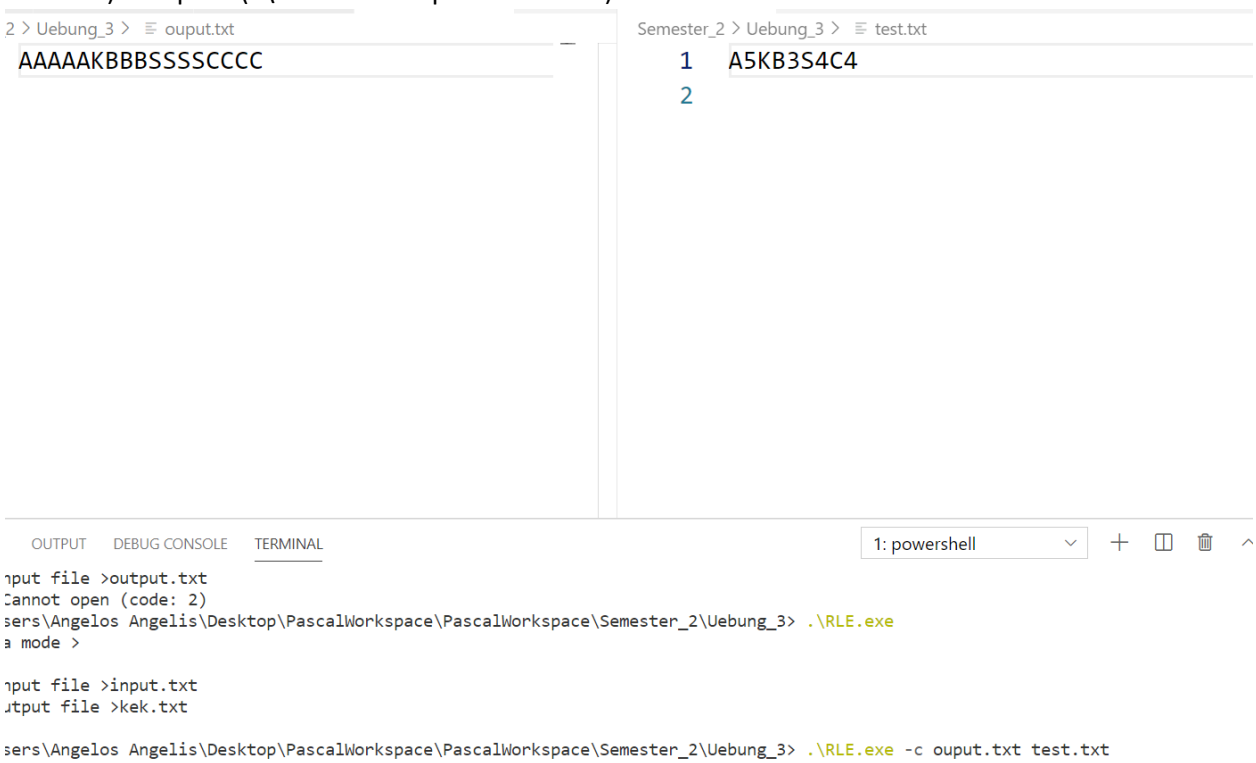
Beispiel:



1) Lösungsidee:
Man bearbeitet zu Beginn alle Paramater nacheinander durch. Zuerst der Modus. Es soll entweder -c oder -d eigegeben werden können. Das ist der erste Parameter daher ist cliIndex auch 1. Danach sollen 2 Textdateien geöffnet und „zubereitet“ werden(cliIndex 2 und 3) wobei die erste der Input ist und die zweite der Output.
Anschließend soll je nachdem was der User ausgewählt hat der Input „compressed“ (-c) oder „decompressed“ (-d) werden.
Beim „compress“ wird mit Hilfe einer for Schleife die Input Datei(bzw. eine Zeile) durchgelaufen werden. Falls ein Buchstabe mehrmals vorkommt soll das mit einer Zählervariable gezählt werden. Wenn dann ein neuer Buchstabe vorkommt wird mit dem Zählen aufgehört und Überprüft wie groß diese Zählervariable ist. Falls sie nur 1 groß ist soll in einem neuen String der Buchstabe nur einmal eingefügt werden. Bei count = 2 soll der Buchstabe 2-mal eingefügt werden. Bei mehr als 2 soll der Buchstabe nur 1-mal eingefügt werden und dann die count Variable. Dieser neue String wird dann ausgegeben.
Beim „decompress“ wird zu Beginn am Ende des inputs ein Spezialbuchstabe eingefügt (z.B #). Dieses Zeichen dient sozusagen als Anker damit man weiß wo der output Endet und so alle unnötigen Zeichen die eventuell eingefügt werden wieder löscht (ich weiß das ist nicht die beste Lösung ;o).Auf jeden Fall muss man beim „decompress“ im Input nach Zahlen suchen und dann den Buchstaben vor der Zahl so oft einfügen wie groß die Zahl ist.
Zuletzt muss man die Textdateien schließen.

Testfall1) Decompress(.\RLE.exe -d input.txt ouput.txt)

Testfall2) Compress(.\RLE.exe -c ouput.txt test.txt)



Testfall3) Kein Input (.\RLE.exe -d input.txt ouput.txt)

Quellcode:

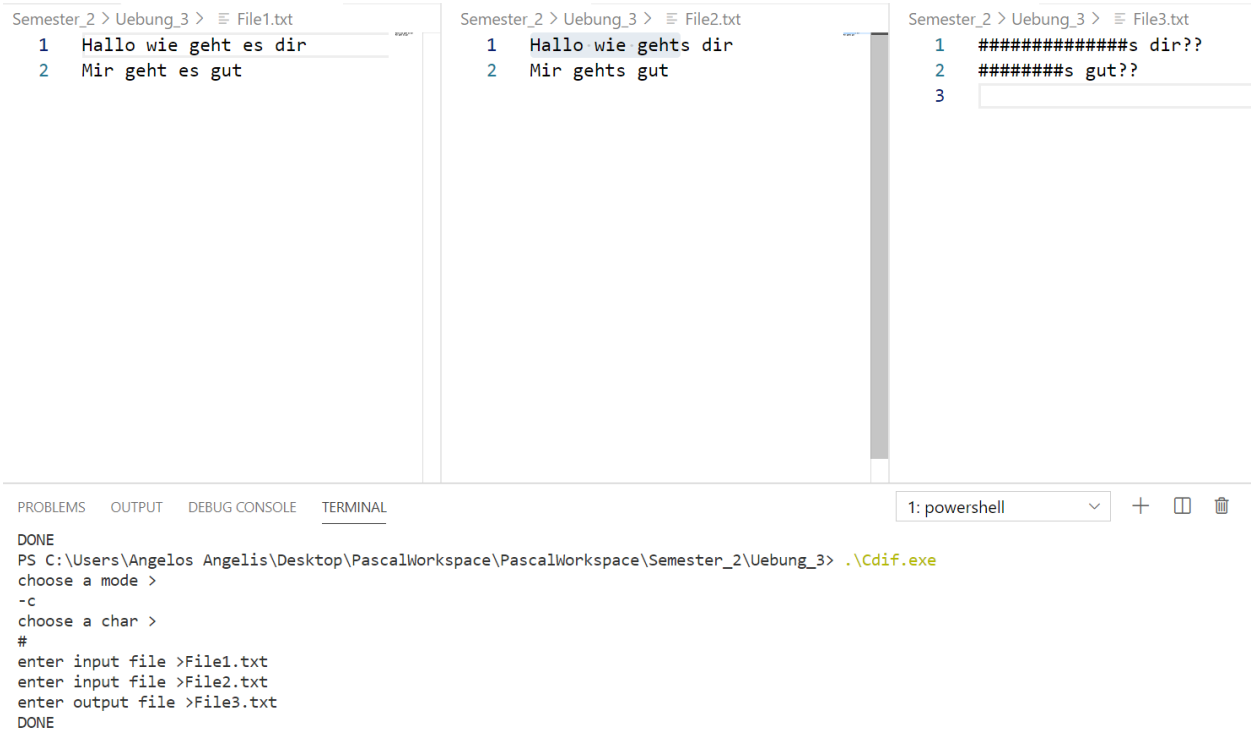
```
PROGRAM RLE;

  Type
    FileMode = (input,output);
    cmode = (compressmode,decompressmode);

  VAR
    compressionMode: cmode;

  PROCEDURE CheckIOError(message : STRING);
    VAR
      errorcode: INTEGER;
  BEGIN (* CheckIOError *)
    errorcode := IOResult;
    IF (errorcode <> 0) THEN BEGIN
      WriteLn('ERROR: ',message, ' (code: ', errorcode, ')');
      HALT;
    END; (* IF *)
  END; (* CheckIOError *)

Function Compress(text: STRING): STRING;
    VAR
      compressedText: STRING;
      count: INTEGER;
      countStr: STRING;
      i: INTEGER;
      prev: CHAR;
  BEGIN (* Compress *)
      compressedText := '';
      count := 0;
      FOR i := 1 TO Length(text)+1 DO BEGIN//man geht ja immer von prev aus, daher bis Length+1
        IF (i > 1) THEN BEGIN
          prev := text[i - 1];
        END; (* IF *)
        IF (i = 1) OR (prev = text[i]) THEN BEGIN
          Inc(count);
        END ELSE BEGIN
          IF (count = 1) THEN BEGIN
            compressedText := compressedText + prev;
          END ELSE IF (count = 2) THEN BEGIN
            compressedText := compressedText + prev + prev;
          END ELSE BEGIN
            (* convert count *)
            Str(count, countStr);
            compressedText := compressedText + prev + countStr;
          END; (* IF *)
          count := 1;
        END; (* IF *)
      END; (* FOR *)
      Compress := compressedText;
  END; (* Compress *)

  FUNCTION IsNumeric(c: CHAR): BOOLEAN;
  BEGIN (* IsNumeric *)
    IsNumeric := (Ord(c) >= 48) AND (Ord(c) <= 57);
  END; (* IsNumeric *)

  Function Decompress(text: STRING): STRING;
    VAR
      textLength : INTEGER;
      temp, decompressedText: STRING;
      count: INTEGER;
```

```
    i,j,k: INTEGER;
    prev: CHAR;
BEGIN (* Decompress *)
    text := text + '#';
    textLength := Length(text);
    temp := '';
    decompressedText := '';
    count := 0;
    k:=1;
    FOR i := 2 TO textLength DO BEGIN
      prev := text[i - 1];
      IF (IsNumeric(text[i])) THEN BEGIN
        Val(text[i],count);
        Delete(text,i,1);
        FOR j := 1 TO count DO BEGIN
          temp := temp + prev;
        END; (* FOR *)
      END ELSE BEGIN
        temp := temp + prev;
      END; (* IF *)
    END; (* FOR *)
    WHILE temp[k] <> '#' DO BEGIN
      decompressedText := decompressedText + temp[k];
      Inc(k);
    END;
    Decompress := decompressedText;
END; (* Decompress *)


PROCEDURE OpenTextFile(VAR textfile: TEXT; cliIndex: INTEGER; mode: FileMode);
VAR
  filename: STRING;
BEGIN (* OpenFile *)
  IF (ParamCount >= cliIndex) THEN BEGIN
    fileName := ParamStr(cliIndex);
  END ELSE BEGIN
    Write('enter ', mode,' file >');
    ReadLn(fileName);
  END;
  {$I-}
  Assign(textfile,filename);
  IF (mode = input) THEN BEGIN
    Reset(textfile);
  END ELSE BEGIN
    Rewrite(textfile);
  END;
  CheckIOError('Cannot open');
  {$I+}
END; (* OpenFile *)

PROCEDURE CloseTextFile(VAR textfile: TEXT);
BEGIN (* CloseTextFile *)
  {$I-}
  Close(textfile);
  CheckIOError('Cannot close');
  {$I+}
END; (* CloseTextFile *)

PROCEDURE InvalidParams;
BEGIN
  WriteLn('Usage: RLE [ -c | -d ] [input | input output]');
  HALT;
END;

PROCEDURE CheckCmode(cliIndex : INTEGER);
VAR
```

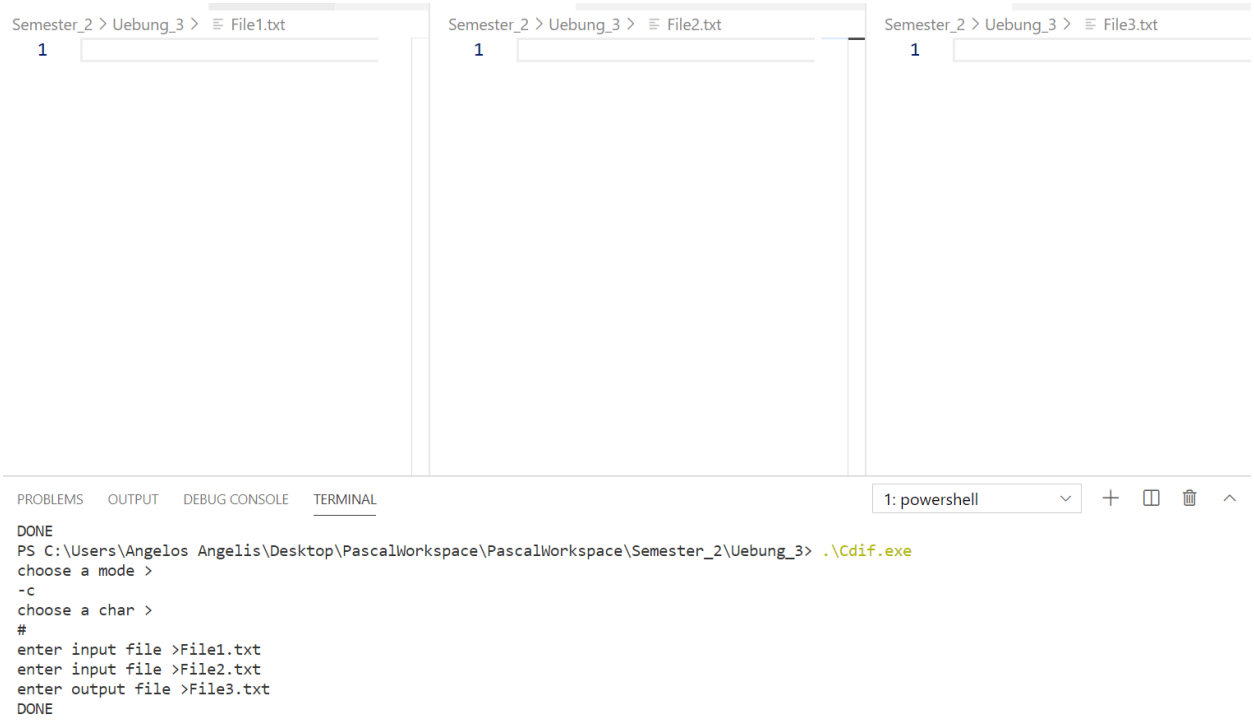
```
    mode: STRING;
  BEGIN (* CheckCmode *)
    IF (ParamCount >= cliIndex) THEN BEGIN
      mode := ParamStr(cliIndex);
      IF (mode = '-c') THEN BEGIN
        compressionMode := compressmode;
      END ELSE IF (mode = '-d') THEN BEGIN
        compressionMode := decompressmode;
      END ELSE BEGIN
        InvalidParams;
      END;
    END ELSE BEGIN
      WriteLn('choose a mode >');
      ReadLn(mode);
      IF (mode = '-c') THEN BEGIN
        compressionMode := compressmode;
      END ELSE IF (mode = '-d') THEN BEGIN
        compressionMode := decompressmode;
      END ELSE BEGIN
        InvalidParams;
      END;
    END;
  END; (* CheckCmode *)

  PROCEDURE CompressDecompress(VAR inputfile,outputFile: TEXT; line: STRING);
  BEGIN (* CompressDecompress *)
    IF (compressionmode = decompressmode) THEN BEGIN
    WHILE (NOT EOF(inputFile)) DO BEGIN
      ReadLn(inputFile, line);
      WriteLn(outputFile, Decompress(line));
    END; (* WHILE *)
  END; (* IF *)
  IF (compressionmode = compressmode) THEN BEGIN
    WHILE (NOT EOF(inputFile)) DO BEGIN
      ReadLn(inputFile, line);
      WriteLn(outputFile, compress(line));
    END; (* WHILE *)
  END; (* IF *)
  END; (* CompressDecompress *)

  VAR
    line: STRING;
    inputFile, outputFile: TEXT;

BEGIN (* RLE *)
  line:= '';
  CheckCmode(1);
  OpenTextFile(inputFile,2 , input);
  OpenTextFile(outputFile,3 , output);

  CompressDecompress(inputFile,outputFile,line);

  CloseTextFile(inputFile);
  CloseTextFile(outputFile);
  WriteLn('DONE');
END. (* RLE *)
```

Lösungsidee:
Zu Beginn müssen wieder alle Parameter bearbeitet werden. Der erste (cliIndex = 1) ist der Modus. Hierbei ist nur -c möglich falls was anderes eingegeben wird eine Fehlermeldung angezeigt.
Der zweite (cliIndex = 2) ist das Auswählen des Strings mit dem die gleichen Buchstaben in den zwei Strings ersetzt werden.
Der dritte und vierte ist das Öffnen und „Zubereiten“ der zwei Input Dateien und der fünfte ist das Öffnen und „Zubereiten“ der output Datei (cliIndex = 3,4,5).
Mit Hilfe einer While Schleife und der Cdif Funktion werden die Zwei inputs verglichen und wenn bei den Buchstaben ein Match aufkommt wird bei der output Datei der ausgesuchte String eingefügt.
Letzten Endes werden die zwei Input und eine Output Dateien geschlossen.

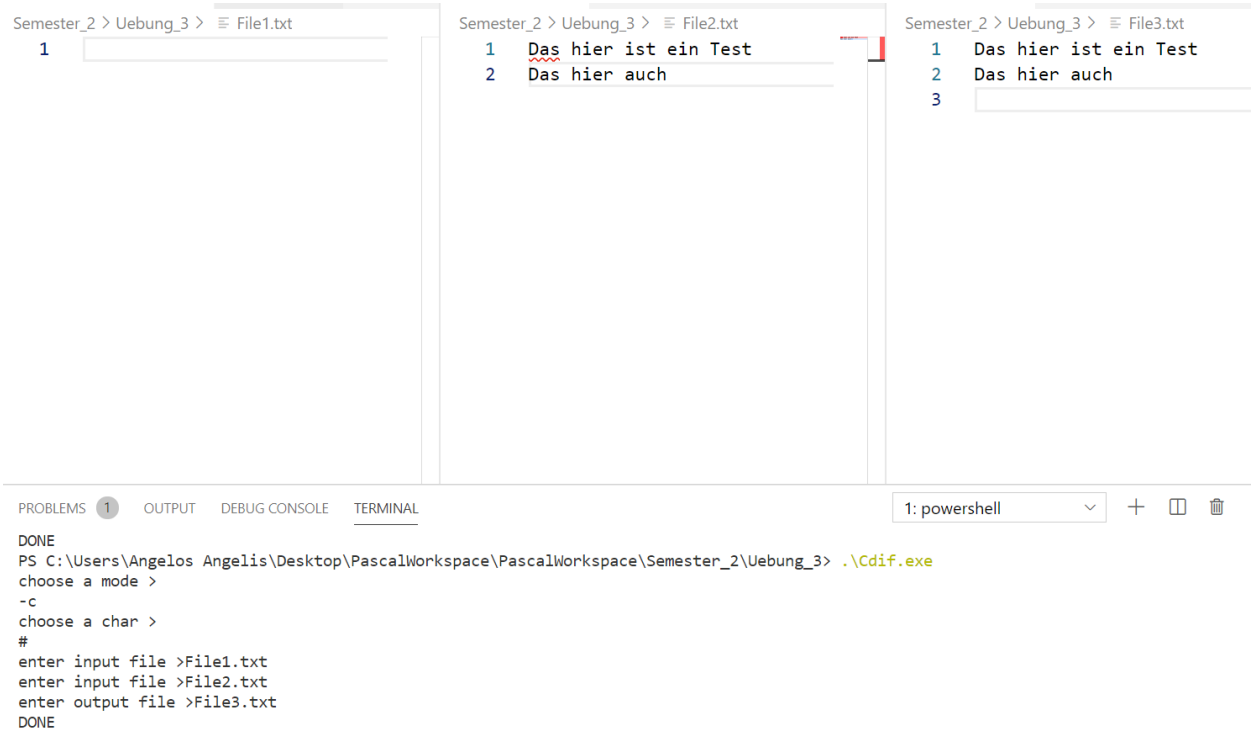
Testfall1)



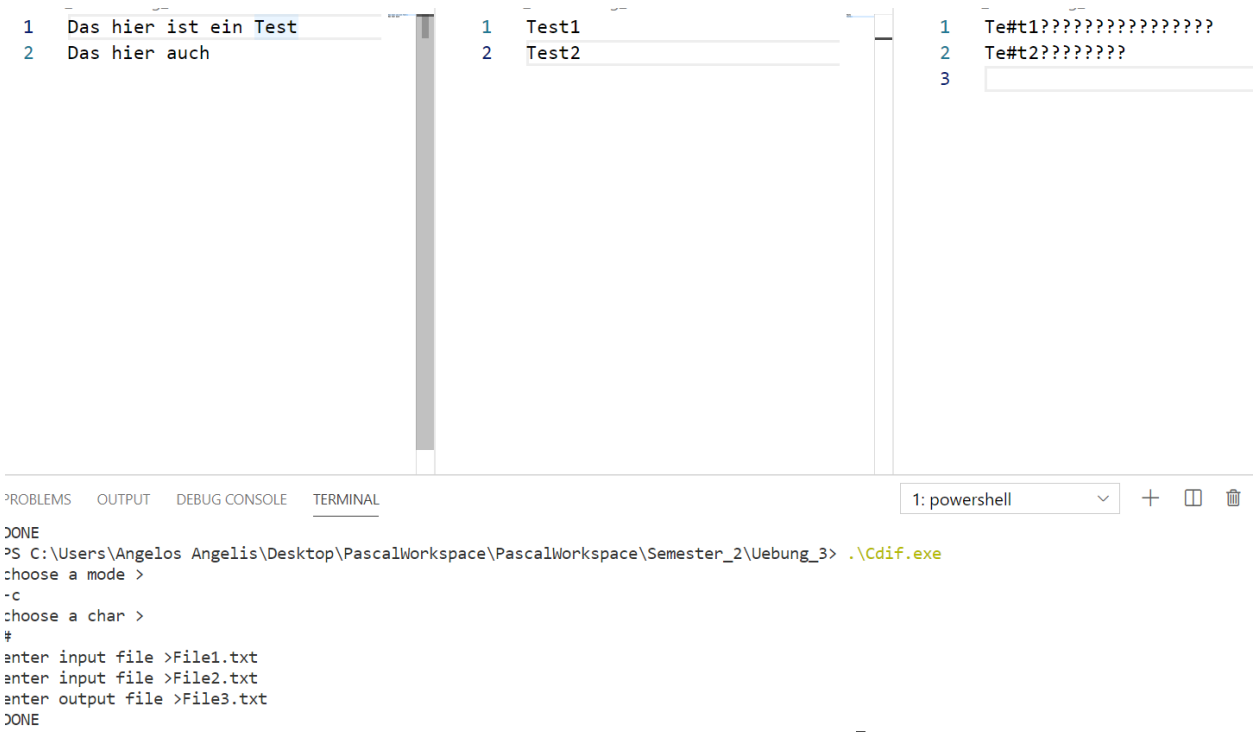
Testfall2) Leere Inputs



Testfall3) Input1 Leer



Testfall3) Input2 ist kürzer als Input1



Quellcode:

```
PROGRAM TextFilter;

  Type
    FileMode = (input,output);

  PROCEDURE CheckIOError(message : STRING);
    VAR
      errorcode: INTEGER;
  BEGIN (* CheckIOError *)
    errorcode := IOResult;
    IF (errorcode <> 0) THEN BEGIN
      WriteLn('ERROR: ',message, ' (code: ', errorcode, ')');
      HALT;
    END; (* IF *)
  END; (* CheckIOError *)

  FUNCTION Cdif(line1,line2,ch: STRING): STRING;
  VAR
    i,j: INTEGER;
    line3: STRING;
  BEGIN (* Cdif *)
    line3 := '';
    i := 1;
    WHILE (i <> Length(line1)+1) AND (i <> Length(line2)+1) DO BEGIN
      IF (line1[i] = line2[i]) THEN BEGIN
        line3 := line3 + ch;
      END ELSE BEGIN
        line3 := line3 + line2[i];
```

```
    END;
    Inc(i);
  END; (* WHILE *)
  IF (i = Length(line1)+1) THEN BEGIN
    FOR j := i TO Length(line2) DO BEGIN
      line3 := line3 + line2[j];
    END; (* FOR *)
  END ELSE BEGIN
    FOR j := i TO Length(line1) DO BEGIN
      line3 := line3 + '?';
    END; (* FOR *)
  END;
  Cdif := line3;
END; (* Cdif *)

PROCEDURE OpenTextFile(VAR textfile: TEXT; cliIndex: INTEGER; mode: FileMode);
VAR
  filename: STRING;
BEGIN (* OpenFile *)
  IF (ParamCount >= cliIndex) THEN BEGIN
    fileName := ParamStr(cliIndex);
  END ELSE BEGIN
    Write('enter ', mode,' file >');
    ReadLn(fileName);
  END;
  {$I-}
  Assign(textfile,filename);
  IF (mode = input) THEN BEGIN
    Reset(textfile);
  END ELSE BEGIN
    Rewrite(textfile);
  END;
  CheckIOError('Cannot open');
  {$I+}
END; (* OpenFile *)

PROCEDURE CloseTextFile(VAR textfile: TEXT);
BEGIN (* CloseTextFile *)
  {$I-}
  Close(textfile);
  CheckIOError('Cannot close');
  {$I+}
END; (* CloseTextFile *)

PROCEDURE InvalidParams;
(* prints a help text and quits *)
BEGIN
  WriteLn('CDiff [-cch] inFile1 inFile2 outFile');
  HALT;
END;

PROCEDURE CheckCmode(cliIndex : INTEGER;c: STRING);
BEGIN (* CheckCmode *)
  IF (ParamCount >= cliIndex) THEN BEGIN
    IF NOT(c='-c') THEN BEGIN
      InvalidParams;
      HALT;
    END;
  END ELSE BEGIN
    WriteLn('choose a mode >');
    ReadLn(c);
    IF NOT(c='-c') THEN BEGIN
      InvalidParams;
      HALT;
    END;
  END;
```

```
  END; (* CheckCmode *)

  PROCEDURE ChooseCh(cliIndex: INTEGER;VAR ch: STRING);
  BEGIN (* ChooseCh *)
    IF NOT (ParamCount >= cliIndex) THEN BEGIN
      WriteLn('choose a char >');
      ReadLn(ch);
    END;
    IF (ch = '') THEN BEGIN
      ch := ' ';
    END; (* IF *)
  END; (* ChooseCh *)

  VAR
    ch: STRING;
    line1,line2: STRING;
    File1, File2, File3: TEXT;


BEGIN (* TextFilter *)
  ch := ' ';
  CheckCmode(1,'-c');
  ChooseCh(2,ch);
  OpenTextFile(File1,3 , input);
  OpenTextFile(File2,4 , input);
  OpenTextFile(File3,5 , output);
  WHILE (NOT EOF(File2)) DO BEGIN
    ReadLn(File1, line1);
    ReadLn(File2, line2);
    WriteLn(File3, Cdif(line1,line2,ch));
  END; (* WHILE *)
  CloseTextFile(File1);
  CloseTextFile(File2);
  CloseTextFile(File3);
  WriteLn('DONE');
END. (* TextFilter *)
```