```
/************
Monica Klosin
CIS 343 02
**************/

//tf.c:

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include "tfW.h"
#include "tfJ.h"

//declaring
void WParse(char *, int);
void JParse(char *, int);
//Points to the width of the text line for the program
//struct PartC{
//int w;
//}width;

/************
function Main, where user puts input for what type of parsing they want
Monica Klosin
**************/
int main(int argc, char *argv[])
{

  //length of file for program
  char lineLength[100];
  int LLength;

  //lorem.txt
  char textfile[100];

  //accpeting the length of the line from terminal -w # or -j #
```

```
int k;
for (k = 1; k < argc; k++)
{
  //use -w
  if (strcmp(argv[k], "-w") == 0)
  {
    if (argv[k + 1] == NULL)
    {

      fprintf(stderr, "R U sure?\n");
      exit(0);
    }
    strcpy(lineLength, argv[k + 1]);
    LLength = atoi(lineLength);
    if (LLength == 0)
    {
      fprintf(stderr, "R U sure?\n");
      exit(0);
    }
    //take to tfW.h
    WParse(&textfile[LLength], LLength);
  }
  //use -j
  else if (strcmp(argv[k], "-j") == 0)
  {
    if (argv[k + 1] == NULL)
    {
      fprintf(stderr, "R U sure?\n");
      exit(0);
    }
    strcpy(lineLength, argv[k + 1]);
    LLength = atoi(lineLength);
    if (LLength == 0)
    {
      fprintf(stderr, "R U sure?\n");
      exit(0);
    }
```

```
      //take to tfJ.h
      JParse(&textfile[LLength], LLength);
    }
    else
    {
      fprintf(stderr, "R U sure?\n");
      exit(0);
    }
  }
  return 0;
}
```

```
//-------------------------------------------------------------------------------
//tfW.h:

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
/************
function to parse the text file based on the number following -w #
-print out the text file with a certain length (#)
-call out any words that are longer then the certain length (#)
**************/
void WParse(char *textfile, int LLength)
{
```

```
//confirming we have size of line length from terminal
// struct PartC width = LLength;
//printf("Line length: %d\n",LLength);

//array to hold the string with a certain line length
//char array[width.w];
//char array[LLength];
//get size of the string
int size = 0;
int i = 0;
bool overflow = true;
//print everything word by word
while (fgets(textfile, 100, stdin))
{

  //printing space between paragraphs
  if (strlen(textfile) == 1)
  {
    printf("\n\n");
    size = 0;
  }

  char *toks;
  toks = strtok(textfile, "\n ");

  //splitting the line by space
  while (toks != NULL)
  {

    //if total token size is less then the length provided

    if (strlen(toks) > LLength)
    {
      //printf("bad tok: %s", toks); //test
      i = i + 1;
      overflow = false;
    }
```

```
      if ((size + strlen(toks)) <= LLength)
      {

        size = size + 1 + strlen(toks);
        printf("%s ", toks);
      }
      else
      {
        // printf("| max width: %i, size: %i, next token: %s\n", LLength, size, toks); //test
        printf("\n%s ", toks);
        size = strlen(toks) + 1;

        //  printf("(size of tok %s: %i, size: %i)",toks, (strlen(toks), size)); //test
      }
      toks = strtok(NULL, "\n ");
    }
  }
  if (!overflow)
  {
    printf("\n\nWarning: %i overfull line(s)", i);
  }
  printf("\n");
  exit(0);
}
//---------------------------------------------------------------------------

//tfJ.h:

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
/************
function to parse the text file based on the number following -j #
-Distribute the words equally given the space (#) provided
**************/
```

```
char *strtok_r(char *, const char *, char **);

void JParse(char *textfile, int LLength)
{

  //get size of the string
  int size = 0;
  int i = 0;

  bool overflow = true;

  //array to hold the string with a certain line length
  char partA[LLength];

  //number of words in line
  int wordsize = 0;

  //print everything word by word
  while (fgets(textfile, 100, stdin))
  {

    //printing space between paragraphs
    if (strlen(textfile) == 1)
    {
      for (int j = 0; j < size; j++)
      {
        printf("%c", partA[j]);
      }
      printf("\n\n");
      //clears
      strcpy(partA, "");
      size = 0;
      wordsize = 0;
    }

    char *toks;
    char *rest;
```

```
toks = strtok_r(textfile, "\n ", &rest);

//splitting the line by space
while (toks != NULL)
{

 //if total token size is less then the length provide
 if (strlen(toks) > LLength)
 {
  i = i + 1;
  overflow = false;
 }

 if (wordsize == 0)
 {
  strcpy(partA, toks);
  wordsize += 1;
  size += strlen(toks);
 }
 else
 {

  if ((size + strlen(toks)) + 1 <= LLength)
  {
   //add token to array
   strcat(partA, " ");
   strcat(partA, toks);
   wordsize += 1;
   size += 1 + strlen(toks);
  }
  else
  {
   //printf("toks:%s, size: %d, wordSize: %d\n\n", toks, size, len);
   int spaceToUse = LLength - size;
   int numofGaps = wordsize - 1;

   int idealGap = spaceToUse / numofGaps;
```

```
    int numerator = 0;
    int denominator = numofGaps;

    char *toksA;
    toksA = strtok(partA, "\n ");

    while (toksA != NULL)
    {
      //print one word
      printf("%s ", toksA);

      for (int i = 0; i < idealGap; i++)
      {
        printf(" ");
      }
      //get extra of how many will need to be printed
      numerator += spaceToUse % numofGaps;

      if (numerator >= denominator)
      {
        numerator -= denominator;
        printf(" ");
      }

      toksA = strtok(NULL, "\n ");
      //printf("toksA:%s\n\n", toksA);
    }
    printf("\n");
    strcpy(partA, toks);
    wordsize = 1;
    size = strlen(toks);
  }
  //toks = strtok(NULL, "\n ");
  //toks = strtok_r(NULL, "\n ", &rest);
}
toks = strtok_r(NULL, "\n ", &rest);
```

```
    }
  }

  for (int j = 0; j < size; j++)
  {
    printf("%c", partA[j]);
  }
  if (!overflow)
  {
    printf("\n\nWarning: %i overfull line(s)", i);
  }

  exit(0);
}
```