CS2263 Lab 4

Stephen Cole

3553803

**makefile**

COMPILER = gcc

C\_FLAGS = -Wall -Wextra

stest: Strings.o stringTest.o

$(COMPILER) $(C\_FLAGS) -o stest stringTest.o Strings.o

Strings.o: Strings.c

$(COMPILER) $(C\_FLAGS) -c Strings.c

stringTest.o: stringTest.c

$(COMPILER) $(C\_FLAGS) -c stringTest.c

**Strings.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Strings.h"

#define MAX\_STRING\_LENGTH 100

char\* mallocString(int stringsize){

char \*strMem = (char \*)malloc(sizeof(char) \* stringsize);

if (strMem == (char \*)NULL)

{

fprintf(stderr, "Memory allocation failed. Program terminating.");

return (char\*)NULL;

}

return strMem;

}

void freeString(char \*s){

free(s);

}

// create a duplicate string of s

// return it

// return (char\*)NULL on failure

// should call mallocString(), and then strcpy()

char \*duplicateString(char \*s){

int stringSize = 0;

while (s[stringSize] != '\0')

stringSize++;

char \*dupeString = mallocString(stringSize);

strcpy(dupeString, s);

if(dupeString == (char \*)NULL)

{

fprintf(stderr, "Memory allocation failed. Program terminating.");

return (char \*)NULL;

}

return dupeString;

}

int fputString(FILE\* pFOut, char\* s)

{

int i;

for(i=0; i<MAX\_STRING\_LENGTH; i++)

{

if(s[i] == '\n')

{

fprintf(pFOut, "%d\n", i);

break;

}

fprintf(pFOut, "%c", s[i]);

}

return i;

}

char\* fgetString(FILE\* pFIn)

{

char\* str;

str = mallocString(MAX\_STRING\_LENGTH);

if( fgets(str, 100, pFIn) == NULL)

return NULL;

char\* dupStr;

dupStr = duplicateString(str);

free(str);

return dupStr;

}

**Strings.h**

#ifndef STRINGS\_H

#define STRINGS\_H

// a cover function for malloc()

// malloc and return memory for a string of stringsize characters

// return (char\*)NULL on failure

char\* mallocString(int stringsize);

// just a cover function for free()

void freeString(char\* s);

// create a duplicate string of s

// return it

// return (char\*)NULL on failure

// should call mallocString(), and then strcpy()

char\* duplicateString(char\* s);

int fputString(FILE\* pFOut, char\* s);

char\* fgetString(FILE\* pFIn);

#endif

stringTest.c

#include <stdio.h>

#include <stdlib.h>

#include "Strings.h"

#define MAX\_STRING\_LENGTH 100

int main(int argc, char\* argv[]){

char\* string1;

char\* string2;

FILE\* fp = fopen(argv[1], "w");

string1 = mallocString(MAX\_STRING\_LENGTH);

string2 = mallocString(MAX\_STRING\_LENGTH);

int length1;

int length2;

if( fgets(string1, 100, stdin) != NULL )

{

length1 = fputString(fp, string1);

}

if( fgets(string2, 100, stdin) != NULL )

{

length2 = fputString(fp, string2);

}

printf("length1: %d, length2: %d\n", length1, length2);

free(string2);

free(string1);

fclose(fp);

char\* str1;

char\* str2;

FILE\* fpr = fopen(argv[1], "r");

str1 = fgetString(fpr);

str2 = fgetString(fpr);

printf("String1: %sString2: %s", str1, str2);

return EXIT\_SUCCESS;

}

**in.txt**

first line

second line

Q0

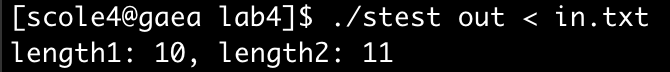
A screenshot of a cell phone

Description automatically generated

Q1

A picture containing drawing

Description automatically generated



**out.txt**

first line10

second line11

Q2

A picture containing drawing

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Q4

A screenshot of a cell phone screen with text

Description automatically generated