EECS2301E

Lab 9
Fall 2018

Due: last day of classes

Lab Objectives

In this lab you will get familiar with dealing with binary and text files and the use of structs.
You have to write 4 programs

1. Write a program to read a text file and stores it as a record file.
2. Write a program to read the record file and display it as text file.
3. Write a program to read a text file and store it as a compressed binary file
4. Write a program to read the compressed binary file and display it as text file

Part 1

In this part, you are given a text file with the following format. The file consists of records, each record has three fields. A string, an integer and a floating point number. Each one of these in a separate line. There are many records in the text file. You have to read and process till EOF. An example is name.txt

```
John Doe
12
34.786543
Jane Doe
17
64.234567
```

Write a program that reads this file, make a record (use structs) of each three lines (the struct's fields are array of char with a maximum length of 50, an integer, and a floating point number). Write all the structs to a binary file called name.bin. Compare between the size of name.txt and name.bin, which is bigger? Why?

Submit 2031 lab_9 l9.c (the file name starts with the letter l as in larry).

Then write another program that reads the binary file name.bin and display the contents each field in a line by itself (similar to name .txt).

Submit 2031 lab_9 readR.c

Part 2

In this part we would like to compress the file. For example although the maximum length for the string is 50, many names are less than fifty characters. Or a floating point number could be 12.45678 which takes 8 bytes to represent in a text format, but the floating point number could be represented I 4 bytes only. This is useful when you save the file or when you transmit it between two machines.

The way we compress the file is as follows:
We store only the string in an as many bytes as it actually needs, for example the string John Doe needs
'J' 'o' 'h' 'n' ' ' 'D' 'o' 'e'
The problem is when we read this, we don't know how many characters are used in this string.
So, you read the string (the entire name line) and count the number of characters. Then you write in the file the number of chactaers foloowed by the actual characters. For example if the name is John Doe as above, that is 8 characters. Then you right 8 bytes to the file. The first byte is the number 8 (of course as a char not integer since integer reacquires 4 bytes) then followed by the 8 characters of “John Doe”

8	'J'	'o'	'h'	…..	'o'	'e'

After that you write to the file the integer and the floating point numbers following the name (each require 4 bytes) for a total of 16 bytes for the first record.
Compare with this with how many characters do we need to store the file in a text format assuming the following

John Doe	9 bytes one byte for \n
56437	6 bytes one byte for \n
12.345678	10 bytes one byte for \n

Total	25 bytes

Compress

Write a program that reads from the files name.txt and compresses the file using the above method to a file test_C.bin

Submit 2031 lab_9 compress.c

Uncompress

Write a program that takes the file test_C.bin, uncompresses it and display the result to the standard output

Submit 2031 lab_9 CRead.c