CSCI 2500: Computer Organization

Homework 2 C Programming

Date Assigned: Feb. 13, 2015 **Due date:** Mar. 6, 2015

This is a *individual* project (no team programming). The deadline for this assignment is **11 PM, Friday, March 6, 2015**. The assignment will not be accepted after that deadline. Important notes: The C source file for the program must be named nibzip.c. This file must be submitted using LMS (email submissions will not be accepted). You must follow the programming and documentation guidelines indicated in the class programming guide posted on LMS. Your program must have several functions in addition to main. Some suggestions regarding this will be provided in class. Grading breakdown will be as follows (total of 100 points):

Structure and Documentation: 15 points
Correctness: 85 points

You are required to write a command line C program called NibZip whose input is a text file containing a constrained (i.e. limited) character set and whose output is a compressed file that is half the size of the original. The program must also print a character distribution table (breakdown of the number and type of input characters). Details regarding input file, compression algorithm, and distribution table format/contents are given below.

Command Line Details: Suppose the executable version of your program is named nibzip.out. The program will be executed by a command line of the following form:

nibzip.out *flag inputfile outputfile*

In the command line above, the arguments *inputfile* and *outputfile* specify the names of the input file (which contains a text file with specific data in it) and the output file respectively. A valid flag argument is one of `-h`, `-a`, `-c`, or `-d`. If the flag is `-h`, your program must produce only help/usage information to standard out. If the flag is `-a`, (analyze only mode) then your program must produce only a print out of the character distribution table computed from the input file to standard output. If the flag is `-c` (`-d`), your program must produce both the character distribution table (again to standard output) and compress (decompress) the contents of the *inputfile* into the *outputfile*.

Details Regarding File Input for Compression: Your NibZip program is going to be designed to work on a special type of input text file. The input text files must contain only the following 15 characters:

0 1 2 3 4 5 6 7 8 9 period (.) dash (-) comma(,) single-space ( ) newline (\n)

Any other characters encountered in the *inputfile* should result in the program printing an error to standard error. Please note the program must throw this error *after* it completes an analysis of the file and not while reading the input file.

Example Input File Format (Hint: avoid generating your test files in Windows):

```
1234.56, -123, 0.9876
1234.57, -124, 0.98
1234.58, -125, 0.9921
1234.59, -126, 0.9711
1234.60, -127, 0.9321
1234.61, -128, 0.9412
```

(Please note each line above ends with a newline character '\n' only)

Compression Algorithm Information: The reason the aforementioned input files cannot have more than 15 unique characters is because no more than 16 different numbers can be represented in a nibble (half-byte). The compression algorithm must pack each character encountered in the input file into a nibble (hence the name NibZip, short for NibbleZip). In order to do this you will need to map each of the valid characters above to a number and than store that number in a nibble with a byte (then write the byte to the output file). You must to use the following character to decimal number mapping:

Character	Number	Character	Number
0	0	8	8
1	1	9	9
2	2	, (comma)	10
3	3	- (dash)	11
4	4	. (period)	12
5	5	' ' (space)	13
6	6	'\n' (newline)	14
7	7		

Table 1: Character to number mapping for NibZip

When your program decompresses a file it must read each number and then write the corresponding character to the output file. The original input file must be reproduced when decompressed (Hint: think about how you will handle case of an empty nibble that can occur when your program is presented with an odd number of bytes/characters in the input file). You should use the Unix command `od` (octal dump) to check the results of your compression.

Character Distribution Table: The character distribution table must show the count of each valid input character from the *uncompressed* version of the file. An example character distribution output is shown below (generated from the the example input file above):

```
Character   Count
0               7
1              18
2              15
3               8
4               8
5               5
6               5
7               4
8               4
9               8
,              12
-               6
.              12
               12
\n              6
others          0
```

Errors to be detected: You need to check only for the usual command line errors (e.g. wrong number of parameters on the command line, invalid flag, the input or the output file can't be opened). In such cases, your program must output a suitable error message to standard error and stop.

Additional Program Requirements: In order to compute the character distribution table you will need to read the entire contents of an uncompressed file into memory. You must allocate this memory dynamically (e.g. using malloc). The character distribution table must be compiled using an array of structs. You may assume a maximum of 500MB for input files. The character distribution table must also be output in the file decompression use case of your NibZip program (i.e. the table must describe the uncompressed data).

Example Program Usage:

Use Case: Help Message Only

```
% nibzip.out -h
usage: nibzip -acdh input_file [output_file]
 -a   Analyze/Check file (will output a data distribution table)
 -c   Compress input_file and write compressed data to output_file
 -d   Decompress the input_file and write the results to output_file
 -h   Display this message
```

Use Case: Analyze only mode

```
% nibzip.out -a input_file
Character   Count
0              7
1             18
2             15
3              8
4              8
5              5
6              5
7              4
8              4
9              8
,             12
-              6
.             12
              12
\n             6
others         0
```

Use Case: Compress a file (Using the example input file above)

```
% nibzip.out -c input_file output_file
Character   Count
0              7
1             18
2             15
3              8
4              8
5              5
6              5
7              4
8              4
9              8
,             12
-              6
.             12
              12
\n             6
others         0

Output File: output_file
   was compressed successfully. New size is 65 bytes.
```

Example hex dump from od (actually hex output) of the compressed `output_file`:

```
             3412    6ac5    12db    d03a    87c9    126e    c534    db7a
             4a12    c9d0    128e    c534    db8a    5a12    c9d0    1e92
             3412    9ac5    12db    d06a    71c9    121e    c634    db0a
             7a12    c9d0    1e32    3412    1ac6    12db    d08a    41c9
             002e
```

Use Case: Decompress a file (Note the character

```
% nibzip.out -d compressed_input_file uncompressed_out_file
Character   Count
0              7
1             18
2             15
3              8
4              8
5              5
6              5
7              4
8              4
9              8
,             12
-              6
.             12
              12
\n             6
others         0

Output File: uncompressed_out_file
   was uncompressed successfully. File size is 130 bytes.
```

Contents of `uncomp_out_file`:

```
1234.56, -123, 0.9876
1234.57, -124, 0.98
1234.58, -125, 0.9921
1234.59, -126, 0.9711
1234.60, -127, 0.9321
1234.61, -128, 0.9412
```