Assignment 2: File I/O

Implement two programs that perform the functionalities described below.

- You should implement your program in C.
  - We will use gcc for the grading.
- You should use the provided library, fastlz.
  - You can find the original source code here: https://github.com/xroche/fastlzlib
  - We provide a simple usage of the library, which would be enough for your homework implementation.
- All the implementations should be executed without any error if there is no assumption, e.g., your implementation should check all corner cases that may cause segmentation faults.

With this homework, you will learn (i) how to perform the binary file IO, (ii) how to work with other open-source libraries, (iii) how to write a Makefile, and (iv) how to design your own data structure to implement the target program.

Due

Submit your implementation before Mar 11, Wednesday, 11:59:59pm, to LMS. We DO NOT allow any late submission.

Submission

Submit a zip file that has your **source code** and **Makefile**. Before the submission, name the zip file with this format: hw2_STUDENTID.zip (e.g., hw2_200012345.zip)

Program Description

Your job is to implement a simple compression utility, which compresses/decompresses multiple files using an external library.

Before going to the detailed program description, the followings show the basic usage of the provided library, which you need to use for this homework.

1. Download the provided source code and unzip it
2. Compile and run it with the following commands

```
$ ls -R
.:
lib_fastlz  main.c

./lib_fastlz:
fastlz.c  fastlz.h  LICENSE

$ gcc main.c lib_fastlz/fastlz.c -o main

$ ./main
512 bytes are compressed to 114
512 bytes are decompressed to 114
Succeed!
```

3. Check the source code of main.c

```c
#include <stdio.h>
#include "lib_fastlz/fastlz.h"

void just_for_your_understanding() {
        unsigned char orig_buf[512];

        // 0. Initialize the buffer with some values
        int i;
        for (i = 0; i < 512; ++i) {
                orig_buf[i] = i % 100;
        }


        // 1. Try compression:
        // It compresses the data in orig_buf to another buffer (comp_buf.)
        // You need to give the size of the data compressed.
        // You will get the size of the compressed data as the return value.
        // Note that the compressed size might be larger than the original.
        unsigned char comp_buf[1024];
        int comp_size = fastlz_compress(orig_buf, 512, comp_buf);
        printf("512 bytes are compressed to %d\n", comp_size);


        // 2. Try decompression:
        // It decompresses the data in comp_buf to decomp_buf.
        // You need to give the compressed data size of comp_buf.
        // You should also give the decomp_buf size, here 2048 bytes,
        // to specify the maximum size of the output buffer.
```

```
        unsigned char decomp_buf[2048];
        int decomp_size = fastlz_decompress(
                        comp_buf, comp_size, decomp_buf, 2048);
        printf("%d bytes are decompressed to %d\n", decomp_size, comp_size);

        // 3. Check all things are correctly decompressed
        if (decomp_size != 512) {
                printf("Something went wrong. ");
                printf("The decompressed size differs from ");
                printf("the original buffer size.\n");
                return;
        }
        for (i = 0; i < 512; ++i) {
                if (orig_buf[i] != decomp_buf[i]) {
                        printf("Something went wrong. ");
                        printf("The decompressed data differ from ");
                        printf("the original buffer.\n");
                        return;
                }
        }

        printf("Succeed!\n");
}
```

- This source code shows the usage of the provided library.
- You can compress and decompress using the two functions: fastlz_compress and fastlz_decompress.
- I put the comments to give the simple usage for each function; but you may also need to find the function descriptions in the header file of official source code:

```
/**
  Compress a block of data in the input buffer and returns the size of
  compressed block. The size of input buffer is specified by length. The
  minimum input buffer size is 16.
  The output buffer must be at least 5% larger than the input buffer
  and can not be smaller than 66 bytes.
  If the input is not compressible, the return value might be larger than
  length (input buffer size).
  The input buffer and the output buffer can not overlap.
*/
int fastlz_compress(const void* input, int length, void* output);


/**
  Decompress a block of compressed data and returns the size of the
  decompressed block. If error occurs, e.g. the compressed data is
  corrupted or the output buffer is not large enough, then 0 (zero)
  will be returned instead.
  The input buffer and the output buffer can not overlap.
  Decompression is memory safe and guaranteed not to write the output buffer
  more than what is specified in maxout.
 */
int fastlz_decompress(const void* input, int length, void* output, int maxout);
```

Now, you have to implement two programs for the homework.

1. **comp**

```
# Usage
$ ./comp compressed-filename source-file1 [source-file2 source-file3 ...]
```

- This program compresses multiple source files into a single file named with "*compressed-filename*".
    - It should use the provided library to compress the data.
    - In many cases, the output-filename would be smaller than the total size of the given source files.
- Hint:
    - You would also need to write file metadata such as names and sizes to decompress them later. Design your own strategy (i.e., a data structure stored in the output file) to save the metadata.

2. **decomp**

```
# Usage
$ ./decomp compressed-filename output-dir
```

- This program decompresses the *compressed-filename* into a directory, *output-dir*.
- If the output directory does not exist, create it.
- You should restore the file names with all original data. Overwrite files if exists.
- Of course, it should also use the provided library to uncompress the data.

Makefile

- You also need to write your own Makefile.
- Typing "make" in the shell should create the two binaries, comp and uncomp.
- You should assume that the provided library files are included in the "lib_fastlz", as a subfolder of the Makefile

Sample

```
$ make
$ ./comp output.dat ipa1.txt ipa_fig1.jpg ipa2.txt ipa_fig2.jpg ipa3.txt
ipa_fig3.jpg
Compression is done
$
$ ./decomp output.dat out
# of source files: 6
reading: ipa1.txt
reading: ipa_fig1.jpg
reading: ipa2.txt
reading: ipa_fig2.jpg
reading: ipa3.txt
reading: ipa_fig3.jpg
Decompression is done
$
$ diff ipa1.txt out/ipa1.txt
$ diff ipa2.txt out/ipa2.txt
$ diff ipa3.txt out/ipa3.txt
$ diff ipa_fig1.jpg out/ipa_fig1.jpg
$ diff ipa_fig2.jpg out/ipa_fig2.jpg
$ diff ipa_fig3.jpg out/ipa_fig3.jpg
```

Note:

- The “diff” command compares two files and shows whether the two files are different. If the files are the same, it prints nothing on the screen.
- The two programs you implement may print anything on the screen (stdout) for debugging purposes. The printed messages above are examples. (That is, you don’t need to print the same output.)
- We will check with other example inputs for grading.