Assignment 2: File I/O

Implement two programs that perform the functionalities described below.

- You can not use any C/C++ library functions other than the ones defined in the standard C library.
- All the implementation 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.

Due

Submit your implementation before Apr 23, Friday, 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 simple merge and split utilities that support encoding and decoding data. You have to implement two programs.

1. **merge**

```
# Usage
$ ./merge merged-filename key source-file1 [source-file2 source-file3 ...]
```

- This program merges multiple source files into a file named with "*merged-filename*".
- It also encodes the data in the merged file with the key.
    - Assume that the user gives an integer key in [0, 255], i.e., a single byte value.
    - Use XOR-based encoding to store the file data, i.e., perform *xor*ing each byte with the key.
    - For example, if a four-byte file has "00 01 FE FF" and the key is 255(=0xFF), you can apply xor for each byte, producing "FF FE 01 00". Write the encoded data into the merged file.
- Hint:
    - You would also need to write file metadata such as names and sizes to split them later. Design your own strategy (i.e., a data structure in the merged file) to save the file metadata.

2. **split**

```
# Usage
$ ./split merged-filename key output-dir
```

- This program splits the merged files into a directory, *output-dir*, while decoding them back.
- If the output directory does not exist, create it.
- You should restore the file names with all original data. Overwrite files if exists.
- You can decode the encoded data by *xor*ing again.
    - For example, if the file is encoded with "FF FE 01 00" and the key is 255, applying xor for each byte produces "00 01 FE FF".

Sample

```
$ ./merge merge.dat 34 ipa1.txt ipa_fig1.jpg ipa2.txt ipa_fig2.jpg
ipa3.txt ipa_fig3.jpg
input key (one byte): 0x22
merge is done
$
$ ./split merge.dat 34 out
input key (one byte): 0x22
# of source files: 6
reading: ipa1.txt.1 79506
reading: ipa_fig1.jpg.1 3049600
reading: ipa2.txt.1 2129
reading: ipa_fig2.jpg.1 83293
reading: ipa3.txt.1 2163
reading: ipa_fig3.jpg.1 368683
split 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.
- We will check with other example inputs for grading.