National Taiwan Normal University
CSIE Programming II

Instructor: Po-Wen Chi
Due Date: 2021.05.18 PM 11:59

Assignment 4

Policies:

- Zero tolerance for late submission.
- You need to prepare a README file about how to make and run your program. Moreover, you need to provide your name and your student ID in the README file.
- For the writing assignment, I only accept pdf. MS. doc/docx format is not acceptable. Moreover, please use Chinese instead of English except foreign students.
- Do not forget your Makefile. For your convenience, each assignment needs only one Makefile. Please put all executable programs in the directory the same with your Makefile.
- The executable programs should be hw0401, hw0402 ....
- You should pack your homework in one zip file. The file name should be StudentId_hw04.zip.

4.1 Split and Recover (25 pts)

Please write a program to split a large file into multiple small files.

```
$ ./hw0401 -s a.out --size 100000
a.out.1 a.out.2 a.out.3
$ ./hw0401 -r a.out a.out.1 a.out.2 a.out.3
$ ./hw0401 --help
Split:
./hw0401 -s [file] --size [Small File Size]
The default small file size is 1000 bytes.
Recover:
./hw0401 -r [output file] [small files]
The input small files may not be in order.
```

4.2 Code Obfuscation (25 pts)

In this class, I have taught you that the coding style is an important thing. This time, I want to do something different. I want you to make a code

hard to be understood. This is called **Code Obfuscation**. Programmers may deliberately obfuscate code to conceal its purpose (security through obscurity) or its logic or implicit values embedded in it, primarily, in order to prevent tampering, deter reverse engineering, or even to create a puzzle or recreational challenge for someone reading the source code. This can be done manually or by using an automated tool, the latter being the preferred technique in industry.

This assignment problem is ask you to do a source code obfuscater that can obfuscate a C code. Do not worry, I will show you some simple ways about how to obfuscate a C code. There are four levels you need to implement:

1. Add redundant blanks and newlines.
2. Change the variable's name to a length 16 random strings.
    - Ex: int a; → int boe1n292fsdfs2gs;
3. Change the function's name to a length 16 random strings except **main** and standard functions.
    - Ex: void f(); → void ifsmf3b12hensfnl();
4. For all integers, change the value to an equation with at least one addition and one multiplication.
    - Ex: int a=10; → int a=7+6/2;

Each level must contains the levels under its level. For example, level 2 must satisfies level 1 and level 2.

```
$ ./hw0402 -l <1-4> -i test.c -o test_output.c
$ ./hw0402 -h
./hw0402 -l [options] -i [input file] -o [output file]
./hw0402 -h
./hw0402 --help
    Display this help manual.
$ ./hw0402 --help
./hw0402 -l [options] -i [input file] -o [output file]
./hw0402 -h
./hw0402 --help
    Display this help manual.
```

I promise that the input file is a single, self-contained, compilable and runnable C code. Your modified code must also be compilable and has the same functions with the original code. For your simplicity, the only data types in this code are **int** and **char**. By the way, you need to know that the real obfuscater is more complex than what you do.

4.3 My Printf (25 pts)

Please develop your own printf, which is called **myprintf** and implemented in **myprintf.h** and **myprintf.c**. TA will prepare hw0403.c, which will include **myprintf.h** in it. So you need to build it in your Makefile. Your **myprintf** must has the same function with printf. For your simplicity, you should only support the only following conversion specifiers:

- d, i: The int argument is converted to signed decimal notation.
- x, X: The unsigned int argument is converted to unsigned hexadecimal (x and X) notation.
- s: Characters from the array are written up to (but not including) a terminating null byte.

The only function that you can use to print on the screen is **fputc**. Do not use printf in your function!!

4.4 ZIP (25 pts)

Do not worry! I will not ask you to implement a compression and decompression program. I just want you to read a ZIP file and get the files in it.

```
$ ./hw0404 hw05.zip
.
+-- hw05/
    +-- hw05.synctex.gz
    +-- hw05.pdf
    +-- hw05.aux
    +-- hw05.log
    +-- hw05.tex
+-- test
```

4.5 Bonus: valist (5 pts)

In this class, I have told you that the following are macros instead of functions. Please find their definitions and explain how they work.

```
#include <stdarg.h>

void va_start(va_list ap, last);
type va_arg(va_list ap, type);
void va_end(va_list ap);
void va_copy(va_list dest, va_list src);
```