C Programming II
2022 Spring
Homework 01

Instructor: Po-Wen Chi

Due: 2022.03.08 PM 11:59

Policies:

- **Zero tolerance** for late submission.
- **Plagiarism is not allowed.** Both source and copycat will be **zero**.
- 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.
  - Your Name and Your ID.
  - The functional description for each code.
  - Anything special.
- Please pack all your submissions in one zip file. **RAR is not allowed!!**
- For convenience, your executable programs must be named following the rule hwXXYY, where the red part is the homework number and the blue part is the problem number. For example, hw0102 is the executable program for homework #1 problem 2.
- I only accept **PDF**. MS Word is not allowed.
- Do not forget your Makefile. For convenience, each assignment needs only one Makefile.

1 Another Character Encoding (20 pts)

In this class, I have taught you what ASCII encoding is. Now there is some unconventional guy who wants to encode a character according to the **reverse bit order** of ASCII. Table 1 is an example.

Please develop a program for a user to input a hex string and print its string according to this reverse encoding. The hex string should be ended with 00.

Table 1: Encoding Comparison Example for Character 'A'

Encoding	Binary	Hex
ASCII	0100 0001	41
Reverse	1000 0010	82

```
$ ./hw0101
Please enter the hex string: 8242C200
ABC
```

Note that all output characters must be **printable**. If the input hex string is invalid, please print a warning message. Your program should be case-insensitive.

2 Replacement (20 pts)

Text replacement is a very important tool in a text editor. Now I want you to implement this function.

The user should input three strings. The first string is the full text. The second string is the keyword string that will be replaced with the last input string. The first string length is at most 4096 and the last two strings are at most 64. Please output the original text and the modified text. The keywords and the new word should be colored.

```
$ ./hw0102
Please enter the string:
To be, or not to be, that is the question.
Please enter the keyword:
be
Please enter the new word:
eat
Original:
To be, or not to be, that is the question.
New:
To eat, or not to eat, that is the question.
```

Note that the above example is not colored but you need to color **be** and **eat** yourself. More than that, in this problem, the keyword search is **case-insensitive**.

3 My String Library v1 (20 pts)

In this class, I have shown you some standard string functions. Of course, you should use them when coding. However, sometimes you may need some functions that are not included in the standard string library. Do not worry, you can implement on your own. For your practice, I want you to implement some existing string functions in your own way.

```
long int mystrtol(const char *nptr, char **endptr, int base);
```

The usage of these functions should be the same with the standard version, including their return values. All requirements are in the manual. You need to prepare **mystring.h**

and TA will prepare **hw0103.c**. Of course, Makefile is your own business. Do not forget to make **hw0103.c** in your Makefile.

You cannot call the corresponding standard functions directly in your implementation.

4 My String Library v2 (20 pts)

Again, please implement some existing string functions in your own way.

```
char *mystrchr(const char *s, int c);
char *mystrrchr(const char *s, int c);
size_t mystrspn(const char *s, const char *accept);
size_t mystrcspn(const char *s, const char *reject);
char *mystrpbrk(const char *s, const char *accept);
char *mystrstr(const char *haystack, const char *needle);
char *mystrtok(char *str, const char *delim);
```

The usage of these functions should be the same with the standard version, including their return values. All requirements are in the manual. You need to prepare **mystring.h**, which can be the same file with the last problem, and TA will prepare **hw0104.c**. Of course, Makefile is your own business. Do not forget to make **hw0104.c** in your Makefile.

You cannot call the corresponding standard functions directly in your implementation.

5 Split (20 pts)

As I have told you, I do not like to process texts with C. Actually, I prefer python. For example, **split()** is one of convenient functions in python. You can see how to use it in the following link.

`https://docs.python.org/3/library/stdtypes.html#str.split`

This time, I want you to implement a similar function in C.

```
int mystrsplit(char ***pppList, int *pCounter, const char *pStr, const char *
    pSeparator);
```

- ppList:
  - Output.
  - An array of **char ***.
  - You need to allocate the required memory yourself.
- pCounter:
  - Output.
  - The number of items in the returned array.
- pStr:
  - Input string.

- pSeparator:
  - Input separator.

If there is any invalid input, return -1; otherwise, return 0. You need to prepare **mysplit.h**, which can be the same file with the last problem, and TA will prepare **hw0105.c**. Of course, Makefile is your own business. Do not forget to make **hw0105.c** in your Makefile.

6 Bonus: perror (5 pts)

In this class, I have shown you how to use a useful function called **strerror**. Actually, there is another similar function called **perror**. Please check the manual and see how to use it. Describe the difference between **strerror** and **perror**. Also provide an example code for **perror**.