National Taiwan Normal University
CSIE Computer Programming I

Instructor: Po-Wen Chi
Due Date: Jan 04, 2022, PM 11:59

Assignment 6

Policies:

- Zero tolerance for late submission.
- 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.
- Please provide a README.

6.1 Point Mirroring (20 pts)

Given a point P in 2-D plane and a line, please implement a program to find the image of that point Q formed due to the mirror. Figure 6.1 is an example.

You need to implement the following functions:

```c
// Use (x1,y1) and (x2,y2) to determine a line.
// If the input is not a valid line, use previous valid one.
void set_line( double x1, double y1, double x2, double y2 );
// Q(c,d) is the mirror of P(a,b) according to the pre-determined line.
// If there is no valid line, return -1; otherwise, return 0.
int32_t get_mirror( double a, double b, double *c, double *d );
```

You need to prepare a header file called **mirror.h** with **mirror.c**. TA will prepare hw0601.c for you. **You MUST build hw0601.c to hw0601 in your Makefile!!** The objective of this problem is to train you how to develop a library.

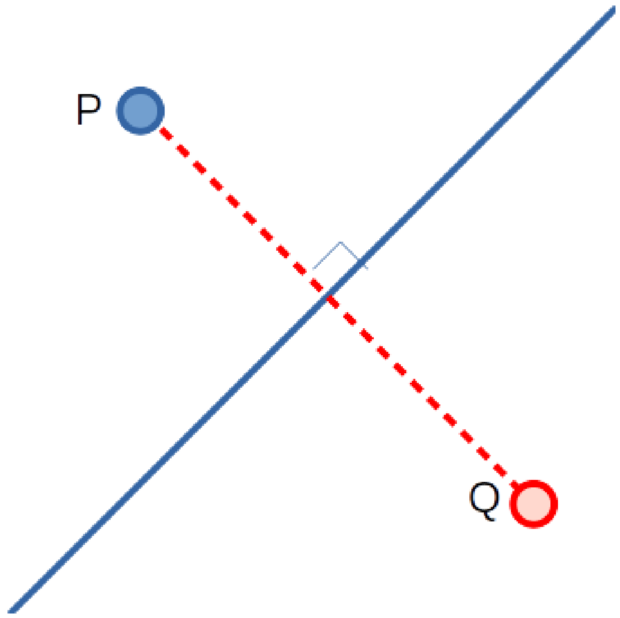

FIGURE 6.1: Point Mirroring.

6.2 Extended Euclidean Algorithm (20 pts)

You all know the Extended Euclidean algorithm（輾轉相除法）, right? If no, do not worry, you can reference the wikipedia.

https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

Extended Euclidean algorithm can not only help you to get the greatest common divisor (gcd), but also make you derive the modular multiplicative inverse. A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus m. Note that a, x are between 0 and $m - 1$. For example, given a modulus 7, $2^{-1} = 4$ since $2 \times 4 = 8 \equiv 1 (\mathrm{mod}\ 7)$. I will explain how to do this.

Extended Euclidean algorithm tells you that given two integers a and b, you can find integers x and y such that

$$ax + by = \gcd(a, b).$$

Suppose a and b are coprime, we can find integers x and y such that

$$ax + by = 1.$$

So if the modular multiplicative inverse of b is y when modulo a since

$$ax + by (\mathrm{mod}\ a) = by (\mathrm{mod}\ a) = 1 (\mathrm{mod}\ a).$$

Now, please implement the following function:

```c
// If a < b, return -1 and c is meaningless.
// If a and b are co-prime, return 1 and c is the
    multiplicative inverse of b mod a.
// If a and b are not co-prime, return 0 and c is the gcd.
int32_t ext_euclidean( uint32_t a, uint32_t b, uint32_t *c );
```

You need to prepare a header file called **ext.h** with **ext.c**. TA will prepare hw0602.c for you. **You MUST build hw0602.c to hw0602 in your Makefile!!** The objective of this problem is to train you how to develop a library.

6.3 Integer Editor (20 pts)

Give a 64-bit signed integer, please develop a program to make the user edit the integer by giving a specific byte position and its new value. The most significant bit is the first bit of the first item. Note that you must use pointer in this problem.

```
$ ./hw0603
Please input an integer: 1
The integer: 1
(1) 0x00 (2) 0x00 (3) 0x00 (4) 0x00 (5) 0x00 (6) 0x00 (7) 0x00
    (8) 0x01
Please enter the position (1-8, 0: End): 8
Please enter the new value (0-255): 255
---
The integer: 255
(1) 0x00 (2) 0x00 (3) 0x00 (4) 0x00 (5) 0x00 (6) 0x00 (7) 0x00
    (8) 0xFF
Please enter the position (1-8, 0: End): 7
Please enter the new value (0-255): 255
---
The integer: 65535
(1) 0x00 (2) 0x00 (3) 0x00 (4) 0x00 (5) 0x00 (6) 0x00 (7) 0xFF
    (8) 0xFF
Please enter the position (1-8, 0: End): 0
```

Reminder: Your computer is **little-endian**.

6.4 Finite State Machine (20 pts)

This problem is definitely the same with your homework 3. I have told you that it is very common to use **switch** case to implement the state machine transition. But this time, you are **not allowed** to use **switch** or **if** in the transition part.

Now you need to develop a program for a user to input integers until the final state.

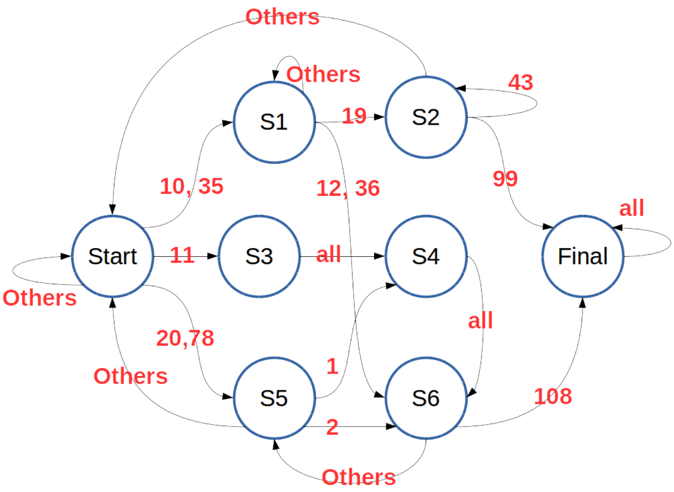

FIGURE 6.2: Deterministic Finite Automata.

```
$ ./hw0604
Start
Please enter an integer: 35
S1
Please enter an integer: 19
S2
Please enter an integer: 43
S2
Please enter an integer: 99
Final
```

Hint: array + function pointer.

6.5 Dynamic Memory Allocation (20 pts)

In this class, I have taught you how to allocate memory dynamically. What if the allocated memory size is not enough or is too large? In standard C, there is a function called **realloc** and you can check it from the manual. This time, I want you to write a function with the similar feature. I want you to adjust the size of an allocated memory.

Please implement the following function:

```c
// *pptr is the original pointer.
// before is the original size and after is the wanted size.
// You need to make sure that the first min( before, after )
   bytes are the same.
void my_realloc( void **pptr, size_t before, size_t after );
```

You need to prepare a header file called **mymem.h** with **mymem.c**. TA will prepare hw0605.c for you. **You MUST build hw0605.c to hw0605 in your Makefile!!** The objective of this problem is to train you how to develop a library. **DO NOT FORGET FREE the OLD MEMORY!!**

6.6 Bonus: printf + %n (5 pts)

In this class, you have used printf many times and used lots of conversions. Then, what is **%n**? Please write down your answer and provide a small example code to prove your answer.

Merry Christmas