Department of Electrical and Computer Engineering
University of Massachusetts Dartmouth

ECE160: Foundations of Computer Engineering I (Spring 2023)
Instructor: Dr. Liudong Xing

LAB # 9

Monday, March 27 (L1) and Wednesday, March 29 (L2)

OBJECTIVES

- To review loops and functions (Exam#2 problems)
- To practice how to open a file using fopen()/fopen_s() and different modes ("r", "w", "a") (L#18)
- To practice how to write to a file using fprintf() (L#18)

SUBMISSION REQUIREMENT

1. Please follow "Submission Guidelines" in the lab section of the course website to submit your solution (program files) to the class M: drive by 5pm, Wednesday, March 29
2. Suggested format for naming your solution files: lab#-your last name-p#.cpp
   For example: lab9-xing-p1.cpp for problem 1; lab9-xing-p2.cpp for problem 2; …

EXERCISES

1. Correct errors in the following program and then run the program to understand what task the program can perform. (Hint: 14 errors)

```
/* This is a debugging problem in Exam 2 */
#include "stdlib.h"
#include "stdio.h"
#include "time.h"

int my160function(void)

void main(void);
{
        int max=7;
        int k, myrand;
        FILE mypointer;
        srand(time(NULL));
        mypointer = fopen("my160file.txt", w);
        for (k=0, k<=Max, k++)
```

```
        {
                myrand = my160function(void);
                fprintf("The %dth random number generated is %d\n", k, myrand);
                fprintf(mypointer, "%d%d\n", myrand);
        }
       fclose(myfile.txt);
}

void my160function(void);
{
        int rand1;
        rand1== rand()%101+30;
        return rand1
}
```

2. **Add missing header files** that include the standard library functions called in the main() function of the following program. Then run the program and understand the output of each printf().

```c
#include "stdlib.h"

int mfib(int n);

void main(void)
{
    double a = 7.76;
    double b = 4.0;
    double c = 3.1;
    double d, e, f;

    d = floor(a * 10) / 10 + sqrt(b) * ceil(c);
    printf("%5.2f\n", d);

    e = ceil(pow(2.0, 4.0) / 10);
    printf("%4.2f\n", e);

    f = fabs(floor(-1.8) / 2) - floor(1.8) / 2;
    printf("%4.2f\n", f);

    printf("%d\n", mfib(0));

    printf("%d\n", mfib(1));

    printf("%d\n", mfib(4));

}
```

```
int mfib(int n)
{
    if ((n == 0) || (n == 1))
        return (n + 1);
    else
        return (mfib(n - 1) * mfib(n - 2));
}
```

3. Run the program and understand the output on the screen.

```
#include "stdio.h"

void main(void)
{
	int j;
	int a = 14;
	int b = 10;
	int ld_a, ld_b;

	for (j = 7; j > 0; j--)
	{
		if ((j == 4) || (j == 6))
			continue;

		if (j == 2)
			break;

		printf("%d\n", j * 2);
	}


	ld_a = a % 10;

	ld_b = b % 10;

	printf("%d\n%d\n", ld_a, ld_b);

	if (ld_a < ld_b)
		printf("%d\n", a);
	else
		printf("%d\n", b);

}
```

4. **Write a complete C program** that can perform the following tasks:
   a. Input 10 numbers from the keyboard and compute the sum of these numbers. A loop is required to implement this task.

b. Then, code the summation result by extracting the least significant digit (LSD) of the sum using the modulo % operation (e.g., 95%10=5) and then subtracting this LSD from the sum. You are required to define and call a function to do this coding task.
c. Finally, output the coded sum on the screen.

You may refer to **Lecture#13, Slide 14 & solution** for an example of the program implementing task a; refer to **Lab#8 Exercise#1** for an example of the code function in task b.

Testing: If you input 1, 3, 5, 7, 9, 10, 12, 14, 16, 18, the sum is 95; the LSD of the sum is 5, and the coded sum will be 90, which is obtained by subtracting the LSD 5 from 95. **The number 90 should be displayed on the screen when the program runs.**

5. Modify your program in Lab8, Exercise 4 so that the program can perform the following tasks
   a. Generate 2 random numbers (refer to L#17, Slides 10 &11)
   b. Then compute the greatest common divisor (gcd) of these two integers using the Euclidean algorithm,
   c. Then write the two random numbers generated and the computed gcd (**one per line**) to a file named gcd.txt. (refer to L#18, Slides 26 & 27)

After you run the program, check the file gcd.txt in your folder to confirm the content.

For example, if two random numbers generated are 16938 and 19168, then the gcd of them is 2. The following data should be displayed as the content of the file gcd.txt

16938
19168
2