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

LAB # 8 Solution
(Relevant Lecture: #16-17)

1. The program below contains some semantics/logic error. Please identify the error, fix the error and run the program. Note that there is no syntax error in the program (i.e., the program can be built successfully). *Hint: the error is related to the parameter passing* (refer to **Lecture#16 Slides 5-10 if necessary**).

The program is supposed to code a number by extracting the least significant digit (LSD) of the number using the modulo % operation and then adding this digit to the number.

Example (Test Case): if you input the original number 1254, the coded number will be 1258, which is obtained by adding the LSD 4 to 1254. The expected correct output from the program should be:

```
The value of a before the function call is: 1254
The value of a after the coding is: 1258
```

You may use the debugging functions in Visual Studio you have learned from **Lab#7** to help you identify the error if necessary.

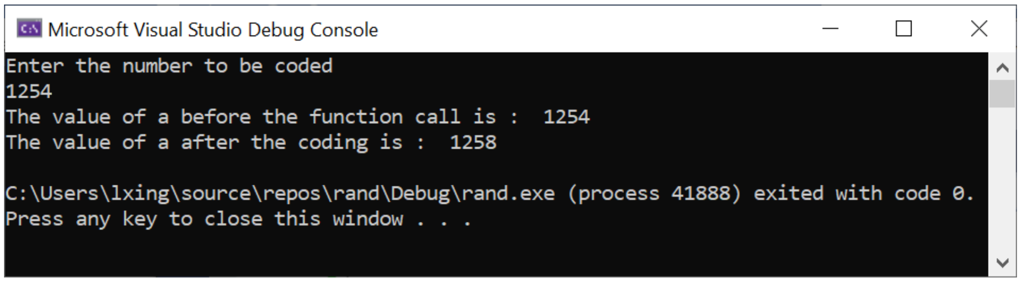
```
#include <stdio.h>

void code(int m);

void main(void)
{
    int a;
    printf("Enter the number to be coded\n");
    scanf_s("%d", &a);
    printf("The value of a before the function call is: %d\n", a);
    code(a);
    printf("The value of a after the coding is: %d\n", a);
}

void code(int m)
{
    int lsd;

    lsd = m % 10;
    m = m + lsd;
}
```

Solution (program after correcting the error):

```
#include <stdio.h>

void code(int *m);

void main(void)
{
    int a;
```

```
        printf("Enter the number to be coded\n");
        scanf_s("%d", &a);
        printf("The value of a before the function call is: %d\n", a);
        code(&a);
        printf("The value of a after the coding is: %d\n", a);
    }

    void code(int *m)
    {
        int lsd;

        lsd = *m % 10;
        *m = *m + lsd;
    }
```

Testing:


Microsoft Visual Studio Debug Console

```
Enter the number to be coded
1254
The value of a before the function call is :  1254
The value of a after the coding is :  1258

C:\Users\lxing\source\repos\rand\Debug\rand.exe (process 41888) exited with code 0.
Press any key to close this window . . .
```

Explanation (Refer to Lecture #16 Slides 5-10):

"Pass by value" is used in the original wrong program, where a copy of the data (argument's value) is passed to the called function code() and the function cannot modify the original variable's value in the caller main().

"Pass by Reference" should be used to correct the error. The called function code() can modify the original variable's value in the caller main(). Any reference to a parameter is the same as a reference to the variable in the calling function main(). The address operator (&) and indirection operator (*) should be used (Refer to the examples on Slide 8 and Slide 10 of Lecture 16).

2. The program below generates a random number and prints it out. However, the same random number will be obtained each time you run the program. Please run the program twice to confirm this. Modify the program so that the program generates a different random number each time you run the program.

```
#include "stdio.h"
#include "stdlib.h"

void main(void)
{
    int rand1;

    srand(997);

    rand1 = rand();
    printf("The number is %d\n", rand1);
}
```

Solution (program after the modification):

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"

void main(void)
{
    int rand1;

    srand(time(NULL));

    rand1 = rand();
    printf("The number is %d\n", rand1);
}
```

3. Modify the program in the last exercise so that the program generates a different random number **in the range of 30 ~ 90** each time you run the program. Run your program at least twice to test your program.

   Solution:

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"

void main(void)
{
    int rand1;

    srand(time(NULL));

    rand1 = rand()%61+30;
    printf("The number is %d\n", rand1);
}
```

4. Write a program that can do the following tasks:
   1) input two integer numbers from the keyboard.
   2) compute the greatest common divisor (**gcd**) of these two integers using the Euclidean algorithm as described in the equation below. Note that remainder(*x*,*y*) can be evaluated using the modulo operator %.

$$\gcd(x,y) = \begin{cases} x & \text{if } y = 0 \\ \gcd(y, \operatorname{remainder}(x,y)) & \text{if } y > 0 \end{cases}$$

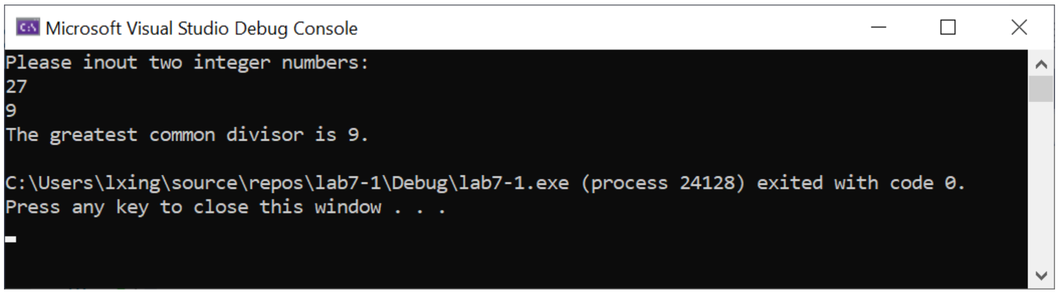
   3) print the **gcd** out.

   You are required to compute the greatest common divisor in **a user-defined recursive function and call the function in the main().**

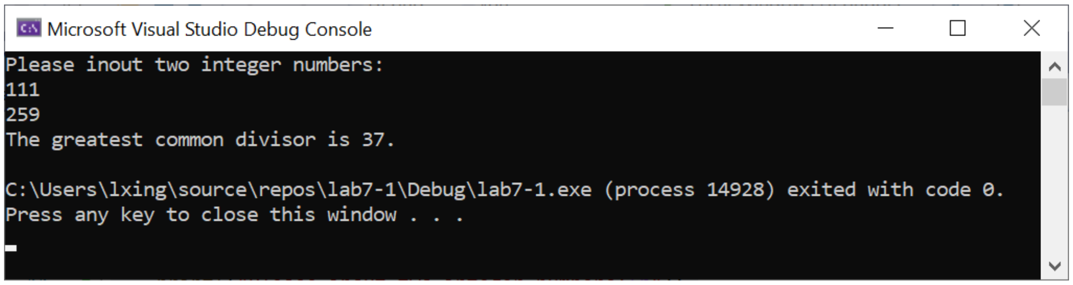
Solution:

```c
#include "stdio.h"
int gcd(int x, int y);

void main(void)
{
    int a;
    int b;
    int d;

    printf("Please inout two integer numbers:\n");
    scanf_s("%d%d", &a, &b);
    d = gcd(a, b);
    printf("The greatest common divisor is %d.\n", d);
}

int gcd(int x, int y)
{
    if (y == 0)
        return x;
    else
        return (gcd(y, x % y));
}
```

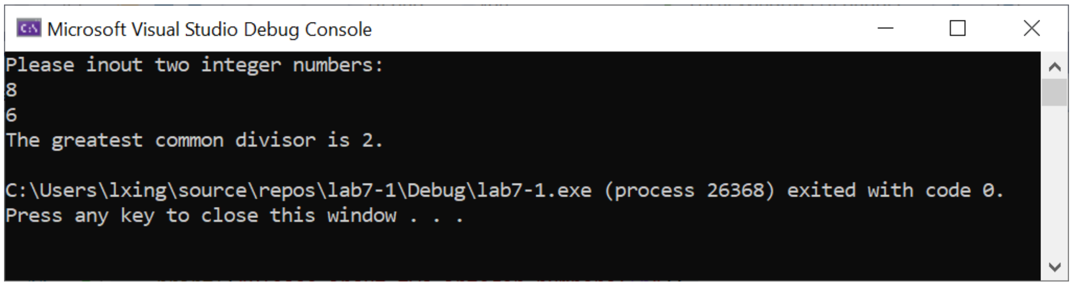
Testing:



```
Please inout two integer numbers:
27
9
The greatest common divisor is 9.

C:\Users\lxing\source\repos\lab7-1\Debug\lab7-1.exe (process 24128) exited with code 0.
Press any key to close this window . . .
```



```
Please inout two integer numbers:
111
259
The greatest common divisor is 37.

C:\Users\lxing\source\repos\lab7-1\Debug\lab7-1.exe (process 14928) exited with code 0.
Press any key to close this window . . .
```



```
Please inout two integer numbers:
8
6
The greatest common divisor is 2.

C:\Users\lxing\source\repos\lab7-1\Debug\lab7-1.exe (process 26368) exited with code 0.
Press any key to close this window . . .
```