

**LAB # 9 Solution**

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
}
}
```

### Solution:

```
/* 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\n", myrand);
        }
    fclose(mypointer);
}

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

### **Program after correcting all 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"); */
    fopen_s(&mypointer, "my160file.txt", "w");
    for (k = 0; k <= max; k++)
    {
        myrand = my160function();
        printf("The % dth random number generated is % d\n", k, myrand);
        fprintf(mypointer, "%d\n", myrand);
    }
    fclose(mypointer);
}

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

```

```

Microsoft Visual Studio Debug Console
The 0th random number generated is 48
The 1th random number generated is 66
The 2th random number generated is 56
The 3th random number generated is 97
The 4th random number generated is 105
The 5th random number generated is 51
The 6th random number generated is 98
The 7th random number generated is 30

C:\Users\lxing\source\repos\exx2\Debug\exx2.exe (process 7528) exited with code 0.
Press any key to close this window . . .

```

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().

```

#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));
}

```

### Solution:

Adding the following before run the program:

```

#include "stdio.h"
#include "math.h"

```

The screenshot shows a window titled "Microsoft Visual Studio Debug Console". The output of the program is displayed as follows:

```

15.70
2.00
0.50
1
2
8

C:\Users\lxing\source\repos\exx2\Debug\exx2.exe (process 9820) exited with code 0.
Press any key to close this window . . .

```

### Justification:

$$\begin{aligned} 1) d &= \text{floor}(7.76 * 10) / 10 + 2.0 * 4.0 \\ &= \text{floor}(77.6) / 10 + 8.0 \\ &= 77.0 / 10 + 8.0 = 7.7 + 8.0 = 15.70 \end{aligned}$$

$$\begin{aligned} 2) e &= \text{ceil}(16 / 10) \\ &= \text{ceil}(1.6) = 2.00 \end{aligned}$$

$$\begin{aligned} 3) f &= \text{fabs}(-2.0 / 2) - 1.0 / 2 \\ &= 1.0 - 0.5 = 0.50 \end{aligned}$$

$$4) \text{m fib}(0) = 0 + 1 = 1$$

$$5) \text{m fib}(1) = 1 + 1 = 2$$

6)

$$\begin{aligned} \text{m fib}(4) &= \text{m fib}(3) * \text{m fib}(2) = 4 * 2 = 8 \\ \text{m fib}(3) &= \text{m fib}(2) * \text{m fib}(1) = 2 * 2 = 4 \\ \text{m fib}(2) &= \text{m fib}(1) * \text{m fib}(0) \\ &= 1 * 2 = 2 \end{aligned}$$

3. Run the program and understand and explain 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);
}

```

**Solution:**

```

Microsoft Visual Studio Debug Console
14
10
6
4
0
10
C:\Users\lxing\source\repos\exx2\Debug\exx2.exe (process 13188) exited with code 0.
Press any key to close this window . . .

```

**Justification:**

When  $j = 7, 5,$  and  $3,$  the for loop body will be executed, outputting  $j * 2$  on the screen. The loop terminates when  $j = 2$  due to the break statement.

$$ld\_a = 14 \% 10 = 4$$

$$ld\_b = 10 \% 10 = 0$$

$4 > 0$  so  $b$  is output on the screen, which is  $10$

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.**

**Solution:**

```
#include <stdio.h>

int code(int m);

void main(void)
{
    int a;
    int sum = 0;
    int codedsum;
    int k = 1; //counter

    while (k <= 10)
    {
        scanf_s("%d", &a);
        sum = sum + a;
        k++;
    }

    codedsum=code(sum);

    printf("The coded sum is: % d\n", codedsum);
}

int code(int m)
{
    int lsd;
    lsd = m % 10;
    m = m - lsd;
    return m;
}
```

**Testing:**

```
Microsoft Visual Studio Debug Console
1 3 5 7 9 10 12 14 16 18
The coded sum is: 90

C:\Users\lxing\source\repos\exx2\Debug\exx2.exe (process 12732) exited with code 0.
Press any key to close this window . . .
```

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.

**Solution:**

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

int gcd(int x, int y);

void main(void)
{
    int rand1;
    int rand2;
    int d;
    FILE *gcdfp;

    srand(time(NULL));

    rand1 = rand();
    rand2 = rand();
    d = gcd(rand1, rand2);

    fopen_s(&gcdfp, "gcd.txt", "w");

    fprintf(gcdfp, "%d\n%d\n%d\n", rand1, rand2, d);

    fclose(gcdfp);
}
```



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

### **Testing Run:**

Your disk drive should contain a file named `gcd.txt` and the file should have three numbers with the third number being the gcd of the first two numbers. Note that every time when you run the program, you may get different random numbers and thus different gcd. Below is an example of a test run result.

