

**Debugging Code**

**LAB # 04**

**SECTION # 1**

**Jesus Horacio Soto Gonzalez**

**SUBMISSION DATE:**

**9/30/2022**

## Problem

Compiler Errors: lab04-1\_1

## Analysis

This program outputs if a integer will divide into another integer with no remainder. This first problem presents a simple program with some syntax errors that need to be fix. We can use the compiler to obtain some information the location and type of error in our program.

## Design

The design of the program was already provided. We had to analyze and correct any error that we found.

## Testing

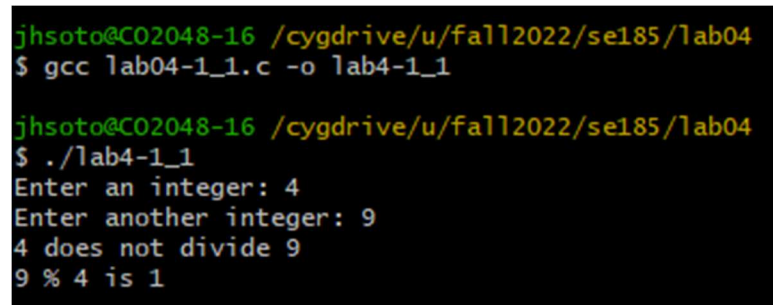
For many of these problems during the testing process I compiled and tested the program each time I made a correction. This help me get a better understanding of the different errors and how the program was intended to be process by the computer.

## Comments

This first problem was an introduction to some of the different syntax errors that we can encounter creating a program.

## Screen Shots

**SS #1**



```
jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ gcc lab04-1_1.c -o lab4-1_1

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab4-1_1
Enter an integer: 4
Enter another integer: 9
4 does not divide 9
9 % 4 is 1
```

## SS #2

```
C:\fall2022\se185\lab04\lab04-1_1.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window

lab04-1_1.c x lab04-1_2.c x lab04-1_3.c x lab04-1_4.c x lab04-1_5.c x lab04-2_1.c x

1  /*-----
2      SE 185: Lab 04 - Debugging Code
3      - Name:      Jesus Horacio Soto Gonzalez
4      - Section:   1
5      - NetID:     jhsoto
6      - Date:      9/23/2022
7  -----*/
8
9  /*-----
10     Includes
11     -----*/
12  #include <stdio.h>
13
14  /*-----
15     Notes
16     -----*/
17  // Compile with gcc lab04-1_1.c -o lab04-1_1
18  // Run with ./lab04-1_1
19  /* This program outputs if a integer will divide into another integer with no remainder. */
20
21  /*-----
22     Implementation
23     -----*/
24  int main(int argc, char *argv[])
25  {
26      int i, j;
27
28      printf("Enter an integer: "); // Missing semicolon
29      scanf("%d", &i);
30
31      printf("Enter another integer: "); // Missing quotation mark before the last parenthesis.
32      scanf("%d", &j); // Missing semicolon after scanf function.
33
34      if (j % i == 0)
35      {
36          printf("%d divides %d\n", i, j);
37
38      } else{ // need to add open brace to start else.
39
40          printf("%d does not divide %d\n", i, j); // Missing letter n for printf function.
41          printf("%d %% %d is %d\n", j, i, (j % i));
42      }
43
44      return 0;
45  }
```

## Problem

Compiler Errors: lab04-1\_2

## Analysis

This program takes two inputs, acceleration, and mass which outputs force = (mass) \* (acceleration). This was a problem with some simple solutions. We had to use the data provided by the compiler which indicated the type of errors in the program.

## Design

Once again, the design of the program was provided, we just had to make some simple modifications in order to compile the program successfully.

## Testing

For this problem I compiled the program as provided to acquire some data from the compiler. With that data provided I was able to determine the errors and presented a solution that outputted a correct value.

## Comments

This problem provided some errors that we had to be more observant to identify along with the data provided by the compiler.

## Screen Shots

### SS #1

```
jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ gcc lab04-1_2.c -o lab4-1_2

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab4-1_2
Enter an acceleration in m/s^2: 47
Enter the mass of the object in kg: 500

You entered 47.000000 m/s^2.
You entered 500.000000 kg.

The force is approximately 23500.00 Newtons.
```

## SS #2

```

C:\fall2022\se185\lab04\lab04-1_2.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run

lab04-1_1.c x lab04-1_2.c x lab04-1_3.c x lab04-1_4.c x lab04-1_5.c x

1  /*-----
2      SE 185: Lab 04 - Debugging Code
3      Name:      Jesus Horacio Soto Gonzales
4      Section:   1
5      NetID:     jhseto
6      Date:      9/23/2022
7  -----*/
8
9  /*-----
10     Includes
11     -----*/
12  #include <stdio.h>
13
14  /*-----
15     Prototypes
16     -----*/
17  void force(double mass, double acceleration); //Change to double
18
19  /*-----
20     Notes
21     -----*/
22  // Compile with gcc lab04-1_2.c -o lab04-1_2
23  // Run with ./lab04-1_2
24  /* This program takes two inputs, acceleration and mass,
25     * and outputs the force = mass * acceleration */
26
27  /*-----
28     Implementation
29     -----*/
30  int main(int argc, char *argv[])
31  {
32      double mass;
33      double acceleration; // Declare variable.
34
35      printf("Enter an acceleration in m/s^2: ");
36      scanf("%lf", &acceleration);
37
38      printf("Enter the mass of the object in kg: ");
39      scanf("%lf", &mass);
40
41      printf("\nYou entered %lf m/s^2.\n", acceleration);
42      printf("You entered %lf kg.\n\n", mass);
43
44      force(mass, acceleration);
45
46      return 0;
47  }
48
49  /**
50   * Given mass and acceleration, calculates the force exerted.
51   *
52   * @param mass - The given mass of an object in kilograms.
53   * @param acceleration - The acceleration of an object in m/s^2.
54   */
55  void force(double mass, double acceleration)
56  {
57      printf("The force is approximately %.2lf Newtons.\n", mass * acceleration);
58  }
59
60

```

## Problem

Compiler Errors: lab04-1\_3

## Analysis

This simple program takes a user input and prints out a message based on that input. For this problem the errors were mostly the absence of the <stdio.h> and <stdlib.h> folders. This helped me learned another type of error that can occur while programming.

## Design

The design of the program was already provided, we made some slight adjustments on the design to correct the code and obtain the desire output. Mostly we had to add a function and some header files.

## Testing

For the testing process, once again the compiler helped us identify some of the errors in the problem. The compiler provided a location and error type, but this area was not that accurate since the problem stems from the lack of the header files.

## Comments

This problem made us realize that even though we can obtain really important and useful information about errors from compiler sometimes it can output some data that might lead you into a different direction from the solution.

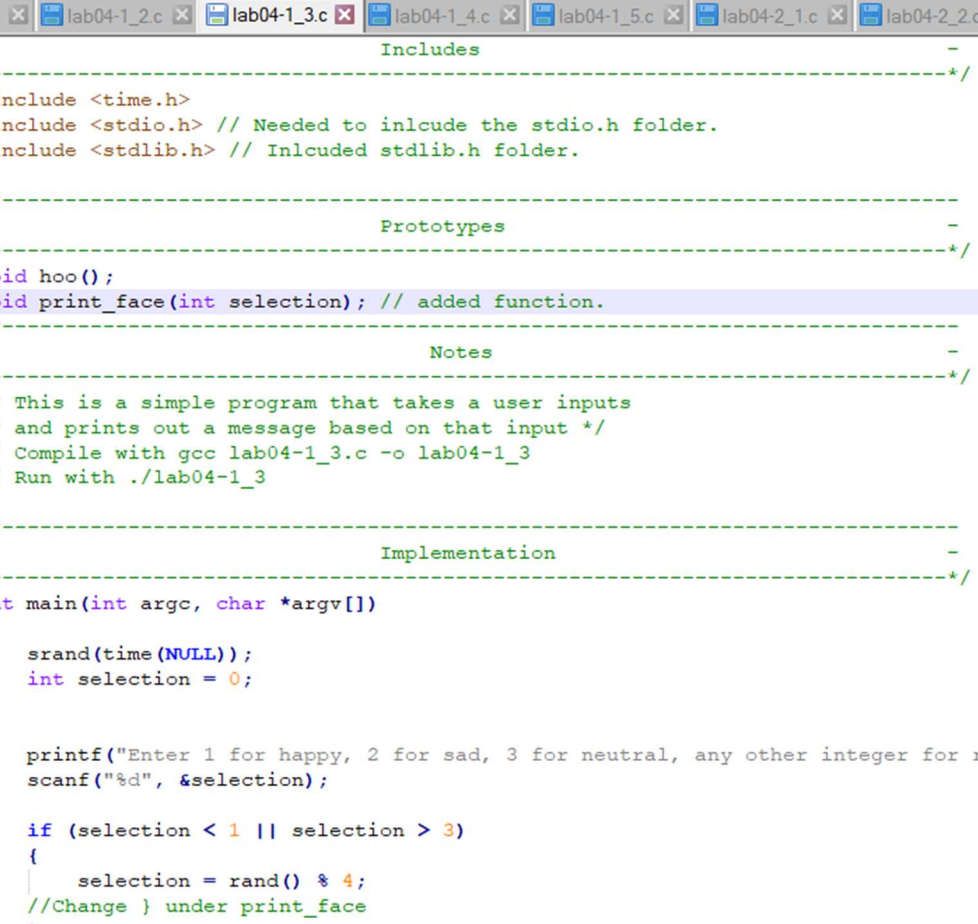
## Screen Shots

SS #1

```
jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ gcc lab04-1_3.c -o lab04-1_3

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab04-1_3
Enter 1 for happy, 2 for sad, 3 for neutral, any other integer for random: 1
Have a nice day! :)
```

**(Not Complete Program) -> Complete program uploaded in zipfile.**



```
10 - Includes -
11 -----*/
12 #include <time.h>
13 #include <stdio.h> // Needed to include the stdio.h folder.
14 #include <stdlib.h> // Included stdlib.h folder.
15
16 /*-----
17 - Prototypes -
18 -----*/
19 void hoo();
20 void print_face(int selection); // added function.
21
22 /*-----
23 - Notes -
24 -----*/
25 /* This is a simple program that takes a user inputs
26 * and prints out a message based on that input */
27 // Compile with gcc lab04-1_3.c -o lab04-1_3
28 // Run with ./lab04-1_3
29
30 /*-----
31 - Implementation -
32 -----*/
33 int main(int argc, char *argv[])
34 {
35     srand(time(NULL));
36     int selection = 0;
37
38     printf("Enter 1 for happy, 2 for sad, 3 for neutral, any other integer for random: ");
39     scanf("%d", &selection);
40
41     if (selection < 1 || selection > 3)
42     {
43         selection = rand() % 4;
44         //Change } under print_face
45     }
46     print_face(selection);
47
48     return 0;
49 }
```

## Problem

## Compiler Errors: lab04-1\_4

## Analysis

This is a program that calculates the energy of one photon, but it contained several simple errors consisting of symbols or character mistakes.

## Design

This problem mostly included some design errors in functions with symbols, misspelling, and more. The design was already provided we just had to make sure that every function had the correct format.

## Testing

This was another type of program where the compiler was completely necessary to identify the location of each error. Most of the errors were easy to fix.

## Comments

This was a good problem to get more comfortable observing the output provided by the compiler and to identify each error line by line.

## Screen Shots

## SS #1

```
jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ gcc lab04-1_4.c -o lab04-1_4

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab04-1_4
Welcome! This program will give the energy, in Joules,
of 1 photon with a certain wave-length.
Please input a wave-length of light in nano-meters.
Please do not enter a negative, or zero, wave-length.
12345
A photon with a wave-length of 12345.000 nano-meters, carries
approximately 0000.00000000000000000000160911 joules of energy.
```



## SS #2

```
C:\fall2022\se185\lab04\lab04-1_4.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

lab04-1_1.c x lab04-1_2.c x lab04-1_3.c x lab04-1_4.c x lab04-1_5.c x lab04-2_1.c x lab04-2_2.c x lab04-2_3.c x lab04-2_4.c x

13 #include <math.h>
14
15 /*-----
16 - Notes -
17 -----*/
18 // Compile with gcc lab04-1_4.c -o lab04-1_4
19 // Run with ./lab04-1_4
20 /* This program calculates the energy of one photon
21 * of user-inputted wave-length of light */
22
23 /*-----
24 - Implementation -
25 -----*/
26 int main(int argc, char *argv[])
27 {
28     double speed_of_light; // ! cannot be used
29     double wave_length; // - cannot be used, used _ instead
30     double length_in_meters; // symbol before lenght_in_meters cannot be used
31     double plank_const; // remove space and join words together with _
32     double energy; // Delete 0
33
34     plank_const = 6.62606957 * pow(10, -34); // Planck's constant
35     speed_of_light = 2.99792458 * pow(10, 8); // Constant for the speed of light
36     wave_length = 0;
37     length_in_meters = 0;
38     energy = 0; // Delete 0
39
40     printf("Welcome! This program will give the energy, in Joules,\n");
41     printf("of 1 photon with a certain wave-length.\n");
42     printf("Please input a wave-length of light in nano-meters.\n");
43     printf("Please do not enter a negative, or zero, wave-length.\n");
44
45     scanf("%lf", &wave_length); // Interchange - for _
46
47     if (wave_length > 0.0) // Interchange - for _
48     {
49         length_in_meters = wave_length / pow(10, 9); // Converting nano-meters to meters
50         energy = (plank_const * speed_of_light) / length_in_meters; // Calculating the energy of 1 photon
51         printf("A photon with a wave-length of %08.3lf nano-meters, carries "
52             "\napproximately %030.25lf joules of energy.", wave_length, energy);
53     } else
54     {
55         printf("Sorry, you put in an invalid number.");
56         printf("Please rerun the program and try again.");
57     }
58
59     return 0;
60 }
```

## Problem

Compiler Errors: lab04-1\_5

## Analysis

This program was meant to calculate the sum of 1 to x, where x is a user input. The problem contained more data than necessary. We had to understand when and where the main function needs to be utilized.

## Design

This problem contained a design error and we had to modify the design of the program to obtain the correct output. There were two main functions in the design of the program provided.

## Testing

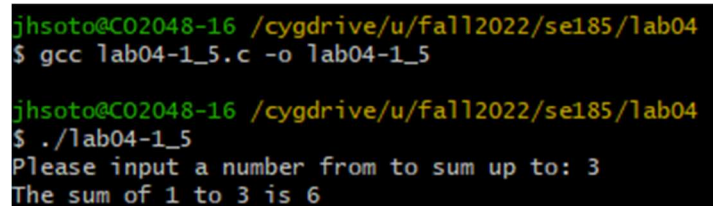
For the testing process the program included two main functions and an extra printf function that we had to delete in order to compile the program correctly.

## Comments

This was a different type of error to demonstrate that not only missing or incorrect values can cause syntax errors, but also additional data might output syntax errors.

## Screen Shots

### SS #1



```
jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ gcc lab04-1_5.c -o lab04-1_5

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab04-1_5
Please input a number from to sum up to: 3
The sum of 1 to 3 is 6
```

## SS #2

```

C:\fall2022\se185\lab04\lab04-1_5.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window

lab04-1_1.c x lab04-1_2.c x lab04-1_3.c x lab04-1_4.c x lab04-1_5.c x lab04-2_1.c x

7  /*-----*/
8
9  /*-----
10  -                               Includes                               -
11  -----*/
12  #include <stdio.h>
13
14  /*-----
15  -                               Prototypes                             -
16  -----*/
17  int sum_function(int number);
18
19  // Deleted main function
20
21  /*-----
22  -                               Notes                                 -
23  -----*/
24  // Compile with gcc lab04-1_5.c -o lab04-1_5
25  // Run with ./lab04-1_5
26  /* This program calculates the sum of 1 to x, where x is a user input */
27
28  /*-----
29  -                               Implementation                         -
30  -----*/
31  int main(int argc, char *argv[])
32  {
33      int input;
34
35      printf("Please input a number from to sum up to: ");
36
37      scanf("%d", &input);
38
39      printf("The sum of 1 to %d is %d\n", input, sum_function(input));
40
41
42      return 0;
43  }
44  // Deleted printf statement
45  // Deleted second main function
46
47
48  /**
49   * Calculates the sum of 1 to number of a given number.
50   *
51   * @param number - The number that determines what the sum will stop adding at.
52   * @return - The sum of 1 to the given number.
53   */
54  int sum_function(int number)
55  {
56      return (number * (number + 1)) / 2;
57  }

```

## Problem

Unintended Results (Logic Errors): lab04-2\_1

## Analysis

This program accepts a user input and determines if the integer is an odd or an even number. For this problem there we had to analyze more in depth what the output of our program needed to be.

## Design

The design for the program provided included some errors mostly with some symbols such as == that we needed to modify into a single = to obtain the desire result.

## Testing

The testing process for logic errors can be problematic since the program is able to compile successfully but the desire output for our program is not effective. For this problem the testing process required deeper thinking and observation as well as for most of the incoming problems.

## Comments

This problem was a really appropriate start for logic errors since they were mostly easy to identify; I was able to understand the purpose of the program which led me to find an effective solution.

## Screen Shots

SS #1

```
jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ gcc lab04-2_1.c -o lab04-2_1

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab04-2_1
Please input an integer: 4
4 is an even number!

jhsoto@C02048-16 /cygdrive/u/fall2022/se185/lab04
$ ./lab04-2_1
Please input an integer: 5
5 is an odd number!
```

## SS #2

```
32 int main(int argc, char *argv[])
33 {
34     int input = 0; // Change == to just =
35
36     printf("Please input an integer: ");
37     scanf("%d", &input);
38
39     if (is_odd(input) == 1) // Add ==
40     {
41         printf("%d is an odd number!\n", input);
42     }
43
44     if (is_even(input) == 1) // Add ==
45     {
46         printf("%d is an even number!\n", input);
47     }
48
49     return 0;
50 }
51
52 /**
53  * Determines whether the given number is even.
54  *
55  * @param number - The number in question of even status.
56  * @return - True if the given number was even.
57  */
58 int is_even(int number)
59 {
60     return !(number % 2);
61 }
62
63 /**
64  * Determines whether the given number is odd.
65  *
66  * @param number - The number in question of odd status.
67  * @return - True if the given number was odd.
68  */
69 int is_odd(int number)
70 {
71     return number % 2;
72 }
73
```

## Problem

Unintended Results (Logic Errors): lab04-2\_2

## Analysis

This program calculates the number of digits in a number from 1 to 10000000. This was a simple program with an error that provided the same output no matter what value the user entered. The solution for error took me some time to figure out.

## Design

The design of the program is very simple, the only problem that the design had was a wrong data type. The solution for the problem was to change the data type to integer or even delete the data type all together at the beginning of the if statements since it was already included previously.

## Testing

For this problem I decided to do an initial testing of the program as given to get an idea of what the error was. Even though it took me some time to figure out after I modify the data types of each if statement the program provided a successful and correct output.

## Comments

Interesting problem that had me thinking, analyzing, and observing every single line of code to make sure that I understood each process of the program.

## Screen Shots

### SS #1

```
jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ gcc lab04-2_2.c -o lab04-2_2

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_2
Please input an integer from 1 up to 10000000: 1
1 digit

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_2
Please input an integer from 1 up to 10000000: 23
2 digits

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_2
Please input an integer from 1 up to 10000000: 123456
6 digits

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_2
Please input an integer from 1 up to 10000000: 1234567
7 digits
```

## SS #2

```
C:\fall2022\se185\lab04\lab04-2_2.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
lab04-2_2.c
46 }
47
48 /**
49  * This function divides a number by the 10^n, to
50  * see if the divided number has "n" digits
51  *
52  * @param number - The number to determine how many whole digits exist within.
53  */
54 void how_many_whole_digits(int number)
55 {
56     if ((int) number / 10000000 != 0) // changed variable type from double to int
57     {
58         printf("8 digits\n");
59     } else if ((int) number / 1000000 != 0) // changed variable type from double to int
60     {
61         printf("7 digits\n");
62     } else if ((int) number / 100000 != 0) // changed variable type from double to int
63     {
64         printf("6 digits\n");
65     } else if ((int) number / 10000 != 0) // changed variable type from double to int
66     {
67         printf("5 digits\n");
68     } else if ((int) number / 1000 != 0) // changed variable type from double to int
69     {
70         printf("4 digits\n");
71     } else if ((int) number / 100 != 0) // changed variable type from double to int
72     {
73         printf("3 digits\n");
74     } else if ((int) number / 10 != 0) // changed variable type from double to int
75     {
76         printf("2 digits\n");
77     } else if ((int) number / 1 != 0) // changed variable type from double to int
78     {
79         printf("1 digit\n");
80     }
81 }
82
```



## Problem

Unintended Results (Logic Errors): lab04-2\_3

## Analysis

This program accepts two integers as user input and swaps their values using two different methods. This was a slightly more complicated program but the solution to the logic error was simple.

## Design

As I stated in the analysis, the design of this program was more complicated than the previous ones, but the solution required to get a successful answer was simple to find. Since format specifier errors are especially common it is one of the things that I typically check first. The design of this program included an erroneous format specifier for an integer.

## Testing

The testing process for this program was short, since before trying to compile the program given, I decided to analyze it line by line. Soon, I was able to find the format specifier error and after analyzing the program further I decided to test it and it outputted the correct values.

## Comments

This problem instructs us to analyze and verify the smaller things that we often ignore but are crucial for any program to provide the correct outputs.

## Screen Shots

### SS #1

```
jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ gcc lab04-2_3.c -o lab04-2_3

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_3
Please input two integers separated by a space: 4 6

Now doing a swap using an extra variable:
Before Swap: First: 4, Second: 6
After Swap: First: 6, Second: 4

Now doing a swap using addition and subtraction:
Before Swap: First: 4, Second: 6
After Swap: First: 6, Second: 4
```



## SS #2

```
int main(int argc, char *argv[])
{
    int first = 0, second = 0;
    printf("Please input two integers separated by a space: ");

    scanf("%d %d", &first, &second); // %lf for double correct by using %d since our variable is an integer.

    printf("\n");
    variable_swap(first, second);

    printf("\n");
    math_swap(first, second);

    return 0;
}
```

## Problem

Unintended Results (Logic Errors): lab04-2\_4

## Analysis

This program calculates values of resistances, voltages, or current using Ohm's Law. This program demonstrated some important uses for programming and how it can be incorporated to many other fields and practical uses. These is why it is important to verify that our program outputs are effective and correct.

## Design

This is another program where the design was a really simple error. Making sure that we are using the correct data type for our program is necessary to obtain the desire outputs. This error happens to be a data type error when we declared the variables for our program.

## Testing

When I first tested this program, the result gave me some insight of where and what the error was. And how much difference a data type error can make in our final result.

## Comments

This was an interesting problem that required some observation of each line of code, and analyzing how the computer might interpret each part and process of our program.

## Screen Shots

SS #1

```
jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ gcc lab04-2_4.c -o lab04-2_4

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_4
selection:
1 for voltage
2 for resistance
3 for current
1
Enter floating point numbers for input...
Please enter a resistance value: 12.5
Please enter a current value: 68.9
Your voltage is: 861.250000 Volts

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$
```

## SS #2

```
int main(int argc, char *argv[])
{
    int selection = 0;
    double v, i, r; // double instead of integer

    printf("selection:\n1 for voltage\n2 for resistance\n3 for current\n");

    scanf("%d", &selection);

    if (selection > 3 || selection < 1)
    {
        printf("Invalid number\n");
        return -1;
    }
}
```

## Problem

Unintended Results (Logic Errors): lab04-2\_5

## Analysis

This program takes in an integer from the user and checks to see if it is a whole number. Additionally, it will tell the user if the number is positive, negative, or zero. This is a program that seems to be somewhat complicated but the solutions for our logic error are simple.

## Design

For this problem, the design of the code was almost correct, the only issues were a missing = symbol at an if statement and an error in a printf statement.

## Testing

During the testing process decided to test the program as it was given to get a better understanding of the results and what needed to be fixed. After analyzing the code, I noticed some simple mistakes mentioned before and after fixing them the program provided a correct output.

## Comments

This program helps us understand the importance of utilizing the right symbols and to make sure that we double check each step of our program in the manner that the computer might interprets the instructions given.

## Screen Shots

### SS #1

```
jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ gcc lab04-2_5.c -o lab04-2_5

jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-2_5
Please type a number between -10000 and 10000: 654
654 is positive and 654 is non-negative and 654 is non-zero and 654 is a whole number.
```

## SS #2

```
int is_zero(int number)
{
    if (number == 0) //add another =
    {
        printf("%d is zero and ", number); // change n to number
        return 1;
    }

    printf("%d is non-zero and ", number);

    return 0;
}
```

## Problem

Putting It All Together: lab04\_3

## Analysis

This program will play a simple Guessing Game with the computer. This is probably one of the most interesting programs we have worked with so far. This program contained many errors all over including some syntax and logic errors.

## Design

There were several issues with the design of the program given, ranging from inaccurate comment lines, absent header folders, and common symbol and typing errors.

## Testing

Since this problem contained many and different kind of errors the best and easiest way to correct them was by analyzing, correcting, and compiling each error we fixed in order. Eventually all the syntax errors were fixed and then we only had to verify each step of our program process to identify the logic errors and correct them.

## Comments

As I stated before, this was one of the most interesting programs we had to work on so far because of the compilation of errors provided and the really interesting game that we are able to play once we correctly fix the program.

## Screen Shots

SS #1

```
jesus@ASUS_GA503 /cygdrive/c/fall2022/se185/lab04
$ ./lab04-3
Do you want to play a game? Enter 'y' to play, anything else not to play. :(
-> y
y
You are guessing a number. The options are 1 through 100.
What is your guess on what number I will select?
-> 50
You guessed too low. Enter another guess.
-> 75
You guessed too low. Enter another guess.
-> 90
You guessed too high. Enter another guess.
-> 85
The number was 85!
You guessed the number correctly!
Do you want to play again? ('y' for yes)
-> c
c
Thanks for playing!
```

## SS #2

(Full program provided in zipfile)

```
int main(int argc, char *argv[])
{
    char prompt = '-';
    int played = 0, computer_guess = 0;

    prompt = ask_to_play(played);
    played = 1;

    while (prompt == 'y')    /* This line does not contain an error */
    {
        computer_guess = select_random_number();
        run_game(computer_guess);
        prompt = ask_to_play(played); // missing e in played
    }

    printf("\n\nThanks for playing!\n");

    return 0;
}

/**
 * Asks the player if they want to play the Guessing Game.
 *
 * @param played_before - Whether the player has played a round of the game before or not.
 * @return - Whether the player wants to play again or not.
 */
char ask_to_play(int played_before)
{
    char yes_or_no;

    if (!played_before)    /* This line does not contain an error */
    {
        printf("Do you want to play a game? "
               "Enter 'y' to play, anything else not to play. :(\n -> ");
        scanf(" %c", &yes_or_no); // & missing
    } else
    {
        scanf(" %c", &yes_or_no);
    }

    printf("%c", yes_or_no);

    return yes_or_no;
}
```

## Questions:

For each program in Part 1 and 2 that you fixed, answer the following:

1. What changes did you have to make to fix the program? Please list the line number with the changes that you made.
2. What kind of issue caused this problem?
  - a. Example: missing semicolon, wrong variable types, missing brackets, etc.

**Changes and explanations also provided in zipfile.**

- [lab04-1 1.c](#) :

Line 28: Missing semicolon

Line 31: Missing quotation marks

Line 32: Missing semicolon

Line 38: Added an open brace

Line 40: Printf function misspelled, missing n

- [lab04-1 2.c](#)

Line 17: changed data type to double

Line 33: Variable needed to be declared

- [lab04-1 3.c](#)

Line 13: Included stdio.h header folder

Line 14: Included stdlib.h header folder

- [lab04-1 4.c](#)

Line 28-32: Removed or replaced symbols that can not be used in declaration

Line 38: Removed the 0

Line 45, 47: Replaced some symbols

- [lab04-1 5.c](#)

Line 19: Deleted extra main function

Line 44: Deleted unnecessary printf statement

Line 45: Deleted extra main function

- [lab04-2 1.c](#)

Line 34: Changed == to just =

Line 39: Added an extra =

Line 44: Added an extra =

- [lab04-2 2.c](#)

Line 56, 59, 62, 65, 68, 71, 74, 77: Changed data type to int (deleting it also works)

- [lab04-2 3.c](#)

Line 37: Changed format specifier to %d since variable is an integer



- [lab04-2 4.c](#)

Line 37: Changed data type to double

- [lab04-2 5.c](#)

Line 110: Added another = symbol

Line 112: Changed n to number

*For Part 3: What is the purpose of the “-Wall” flag? Do you have to fix all of the messages that it gives you?*

As stated in the lab the flag will tell the compiler to warn us of a potential issue in your code. These are issues that do not cause compile errors but may cause issues when you run your code.

The flag basically provides us an extra help to identify some of the problems regarding our programs. Even though is a good practice to use the flag, I wanted to avoid using it to practice and to learn to recognize some of the errors that can occur while programming.