

Reta Gela  
14 September 2023

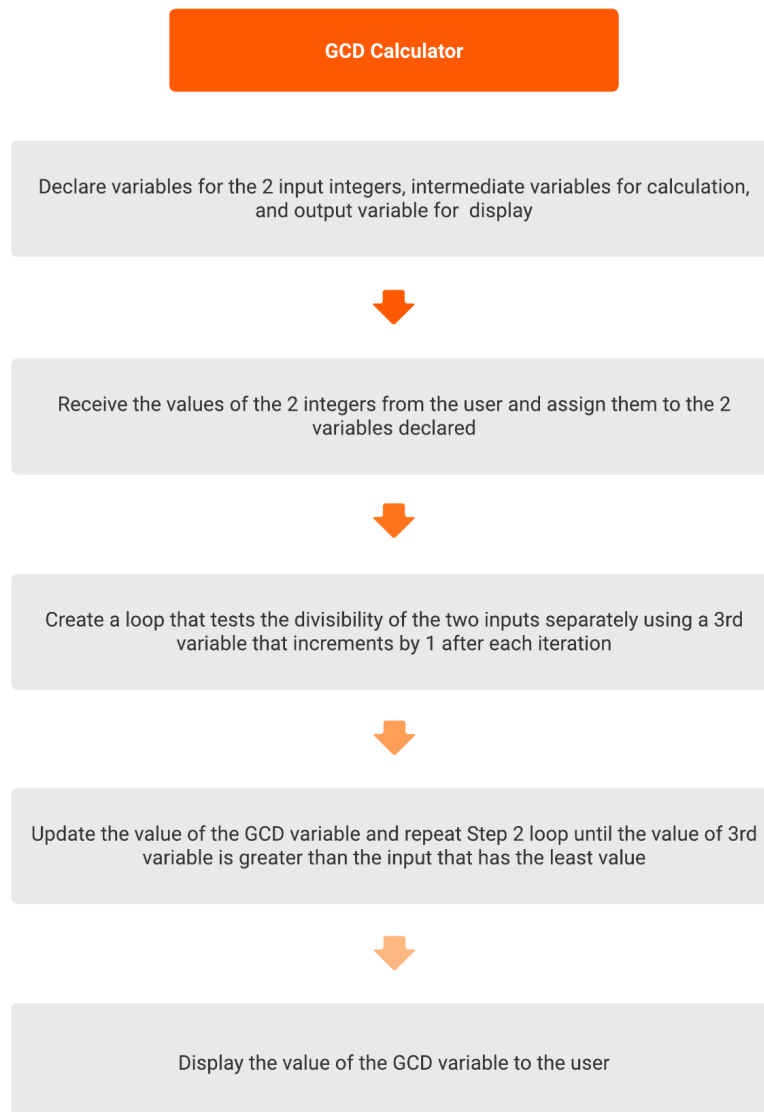
## ENEB 340 Lab 3 Report

### ***Procedure***

First, I created a 'lab3' directory in my home directory 'r374.' From there, I created source code files and labeled them based on each section of the instructions. For reference, I have included images of all 5 source code files with documentation for each program on page 11.

### **Lab3-1: GCD Calculator**

For this C program, I created a program that would take 2 input values from the user and calculate the greatest common divisor between them. I used the flowchart below to help me design the final code of the program.



*Figure 1: GCD Program Flowchart*

```
r374@raspberrypi:
File Edit Tabs Help
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 31
Enter the second number: 37
The GCD is 1
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 7
Enter the second number: 14
The GCD is 7
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 36
Enter the second number: 48
The GCD is 12
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 48
Enter the second number: 12
The GCD is 12
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 37
Enter the second number: 37
The GCD is 37
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 123
Enter the second number: 456
The GCD is 3
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
```

Figure 2: Test Case Outputs for GCD program

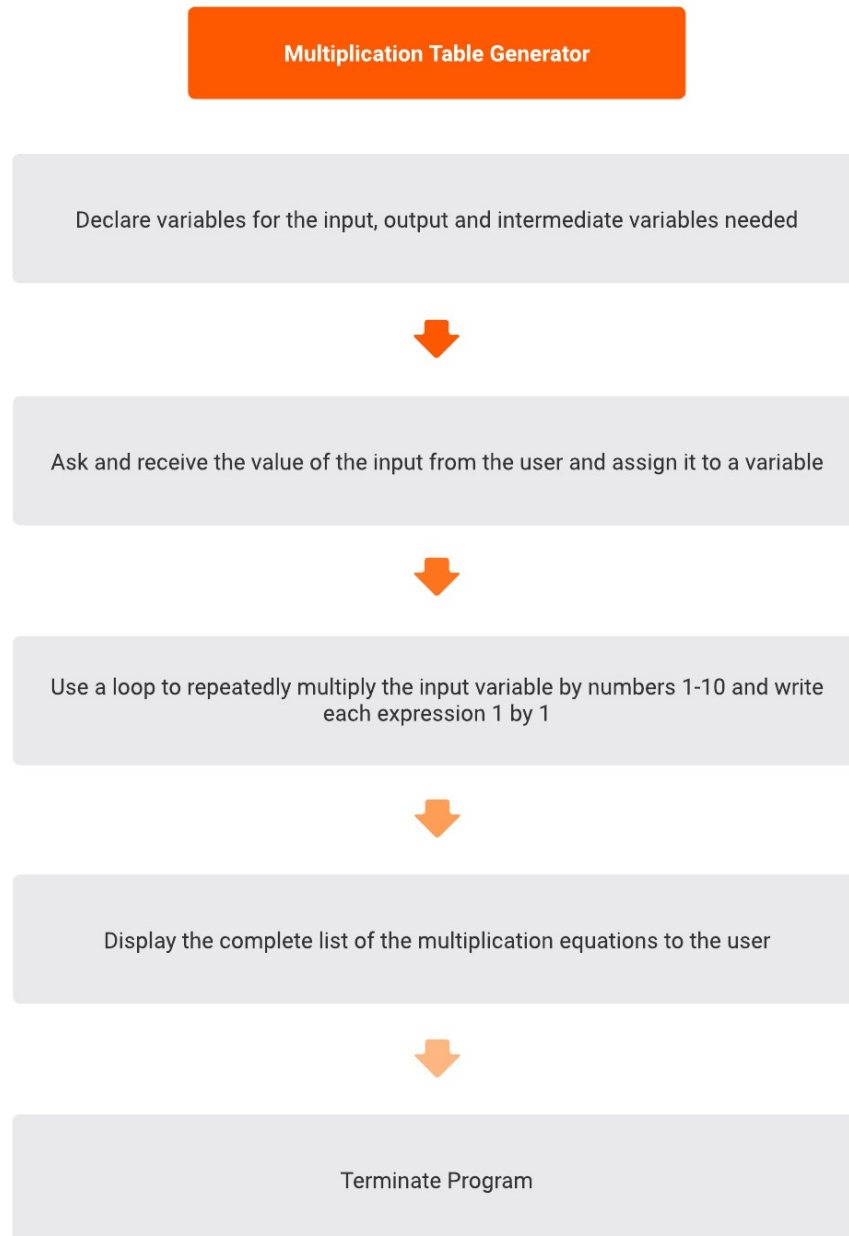
After writing and compiling the code, I executed the file and inserted the test cases given by the lab instructor. The results of these test cases are in Figure 1 and 2.

```
r374@raspberrypi:
File Edit Tabs Help
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 0
Enter the second number: 5
The GCD is 5
r374@raspberrypi:~/lab3 $ ./lab3-1.exe
WELCOME TO THE GCD CALCULATOR!
Enter the first number: 0
Enter the second number: 0
The GCD is 0
r374@raspberrypi:~/lab3 $ █
```

Figure 3: Test Case Outputs (Continued)

### Lab3-2: Multiplication Table

After successfully creating this program, I then began the second C problem: a multiplication table generator. This program was to take an integer from the user and create a 1 to 10 multiplication table using that integer. Again, I created a flow chart to help me visualize the C code needed to perform this task. The flow chart is in Figure 3.



*Figure 4: Multiplication Program Flowchart*

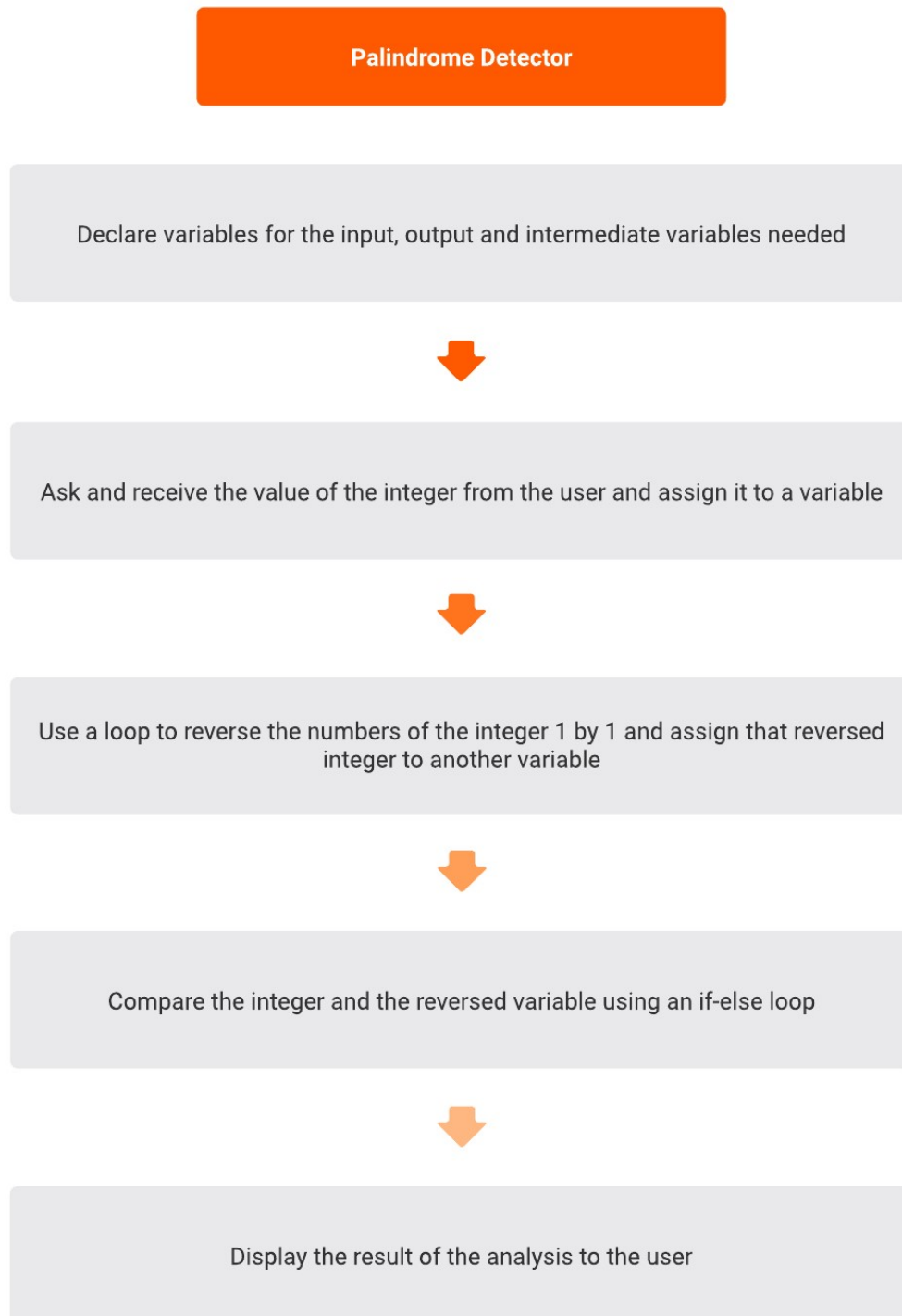
Using the flowchart, I constructed each portion of the C code chronologically and compiled. After a few adjustments, the program compiled and executed successfully. Using the test cases given by the instructor, I tested the functionality of the program (see Figure 5).

```
r374@raspberrypi: ~/lab3
File Edit Tabs Help
r374@raspberrypi:~/lab3 $ gcc lab3-2.c -o lab3-2.exe
r374@raspberrypi:~/lab3 $ ./lab3-2.exe
MULTIPLICATION TABLE GENERATOR
Please enter a positive integer: 0
0 * 1 = 0
0 * 2 = 0
0 * 3 = 0
0 * 4 = 0
0 * 5 = 0
0 * 6 = 0
0 * 7 = 0
0 * 8 = 0
0 * 9 = 0
0 * 10 = 0
r374@raspberrypi:~/lab3 $ ./lab3-2.exe
MULTIPLICATION TABLE GENERATOR
Please enter a positive integer: 1
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
r374@raspberrypi:~/lab3 $ ./lab3-2.exe
MULTIPLICATION TABLE GENERATOR
Please enter a positive integer: 17
17 * 1 = 17
17 * 2 = 34
17 * 3 = 51
17 * 4 = 68
17 * 5 = 85
17 * 6 = 102
17 * 7 = 119
17 * 8 = 136
17 * 9 = 153
17 * 10 = 170
r374@raspberrypi:~/lab3 $ ./lab3-2.exe
MULTIPLICATION TABLE GENERATOR
Please enter a positive integer: 1000
1000 * 1 = 1000
1000 * 2 = 2000
1000 * 3 = 3000
1000 * 4 = 4000
1000 * 5 = 5000
1000 * 6 = 6000
1000 * 7 = 7000
1000 * 8 = 8000
1000 * 9 = 9000
1000 * 10 = 10000
r374@raspberrypi:~/lab3 $
```

Figure 5: Test Case Outputs for Multiplication Program

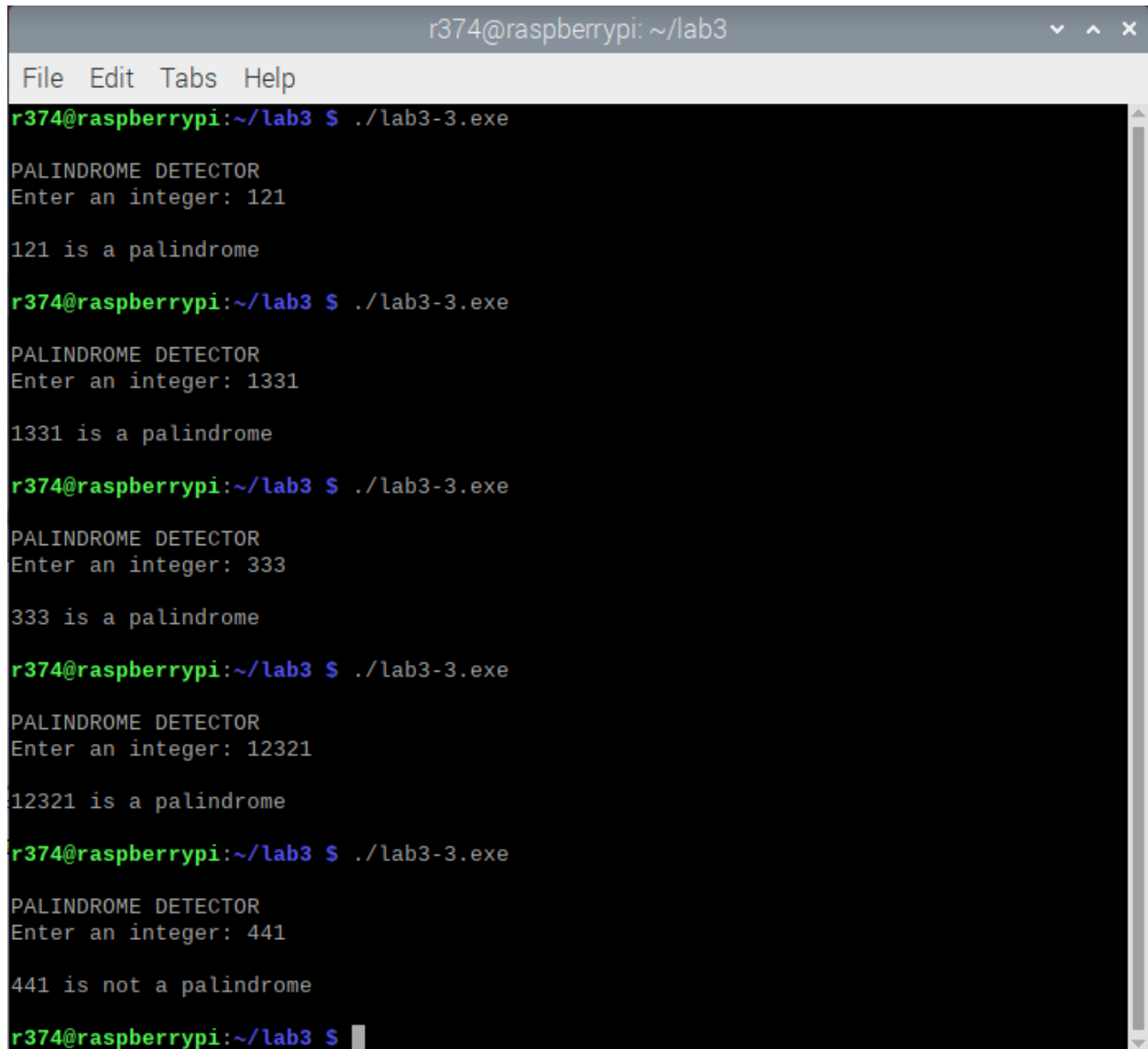
### Lab3-3: Palindrome

In the third problem, I was tasked with creating a program that would determine whether the integer given by the user was a palindrome or not. Just like before, I created a pseudocode and flowchart to help construct the basic elements of the code (see Figure 6).



*Figure 6: Palindrome Program Flowchart*

After making several adjustments to the program, it finally compiled and executed the operation. I used the test cases given by the lab instructor and received the following results in Figure 7).



```
r374@raspberrypi: ~/lab3
File Edit Tabs Help
r374@raspberrypi:~/lab3 $ ./lab3-3.exe
PALINDROME DETECTOR
Enter an integer: 121

121 is a palindrome

r374@raspberrypi:~/lab3 $ ./lab3-3.exe
PALINDROME DETECTOR
Enter an integer: 1331

1331 is a palindrome

r374@raspberrypi:~/lab3 $ ./lab3-3.exe
PALINDROME DETECTOR
Enter an integer: 333

333 is a palindrome

r374@raspberrypi:~/lab3 $ ./lab3-3.exe
PALINDROME DETECTOR
Enter an integer: 12321

12321 is a palindrome

r374@raspberrypi:~/lab3 $ ./lab3-3.exe
PALINDROME DETECTOR
Enter an integer: 441

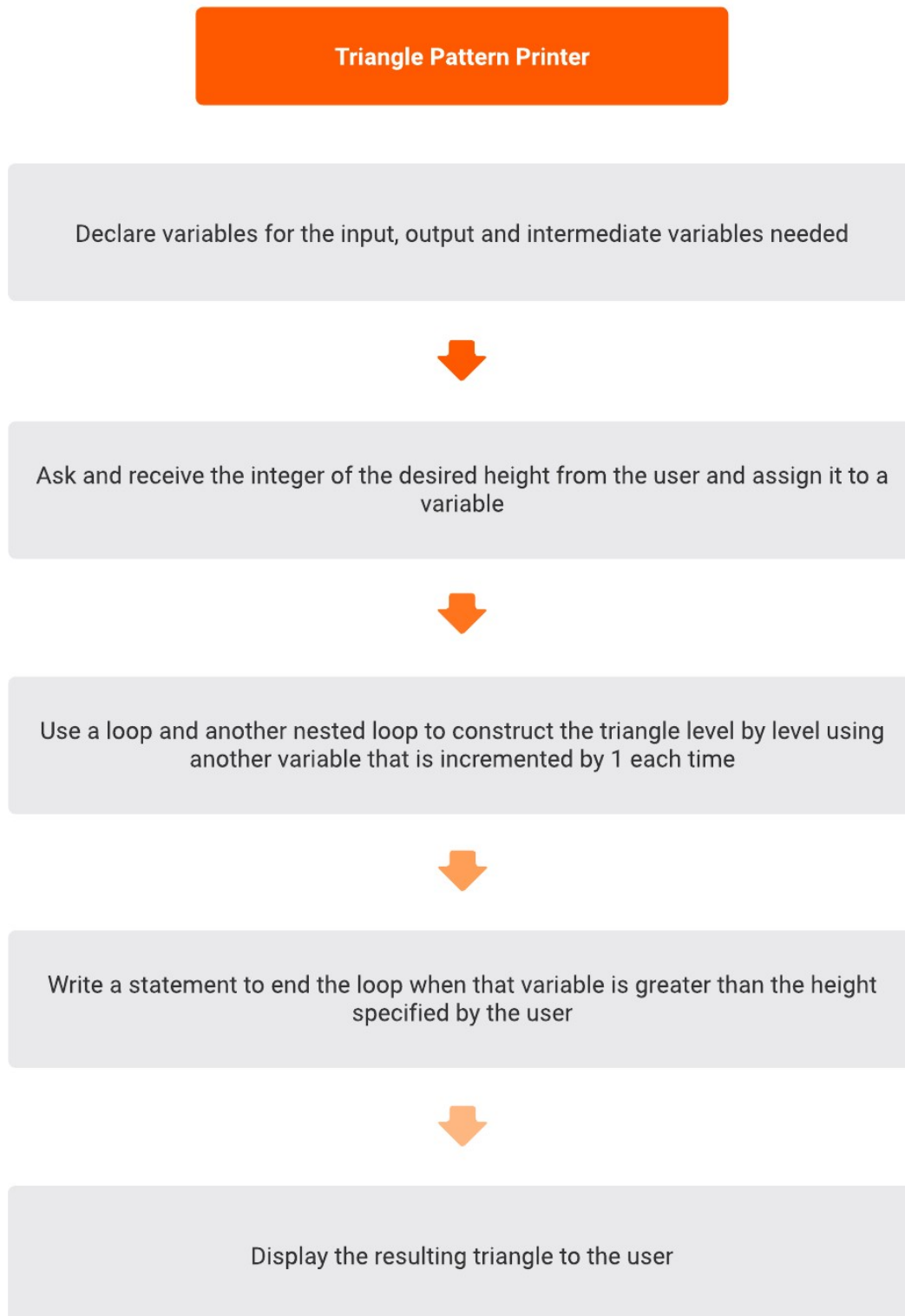
441 is not a palindrome

r374@raspberrypi:~/lab3 $
```

Figure 7: Test Case Outputs for the Palindrome Program

#### Lab3-4: Triangle Pattern Printer

After that program, I then created another flow chart for a program that would create an isosceles triangle of asterisks (\*) with a height of the integer received from the user. The flowchart is in Figure 8.



*Figure 8: Triangle Printer Flowchart*

After completing the code and debugging, I used some test cases provided to assess the validity of the program's outputs (see Figure 9).

```
r374@raspberrypi: ~/lab3
File Edit Tabs Help
r374@raspberrypi:~/lab3 $ ./lab3-4.exe
Enter the height of the triangle: 5

  *
 ***
*****
*****
*****

r374@raspberrypi:~/lab3 $ ./lab3-4.exe
Enter the height of the triangle: 8

  *
 ***
*****
*****
*****
*****
*****
*****
*****
*****

r374@raspberrypi:~/lab3 $
```

Figure 9: Test Case Outputs for Pattern Program

### Lab3-5: Euclidean Distance

In the final program problem, I created a program that calculated the distance between 2 coordinate points given by the user. The flowchart of the code I created is in Figure 10 to the right.

I used the flowchart to break down the function of the code into manageable pieces. I compiled and executed my code in the terminal. Using the test cases provided in Canvas, I tested the output of the program several times and made necessary fixes until it worked properly.

The outputs of the test cases are in Figure 11.

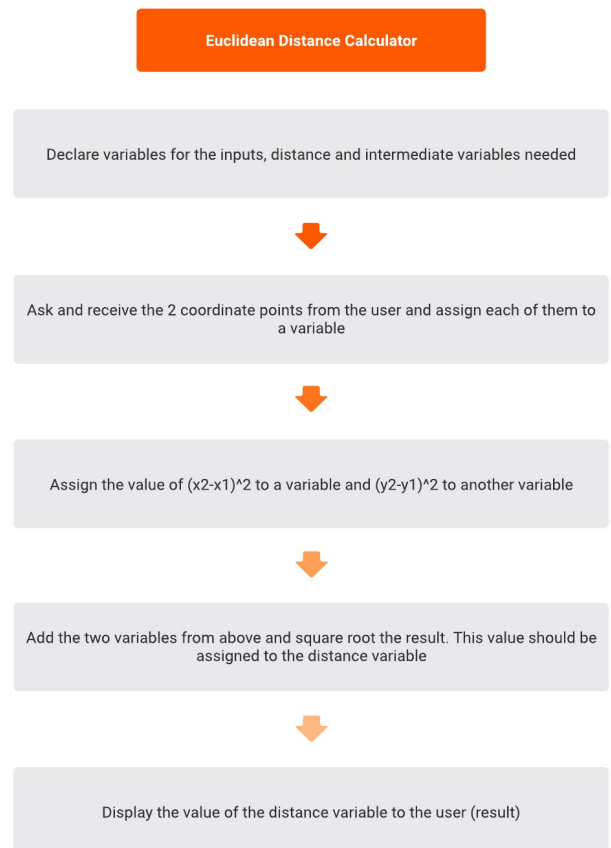
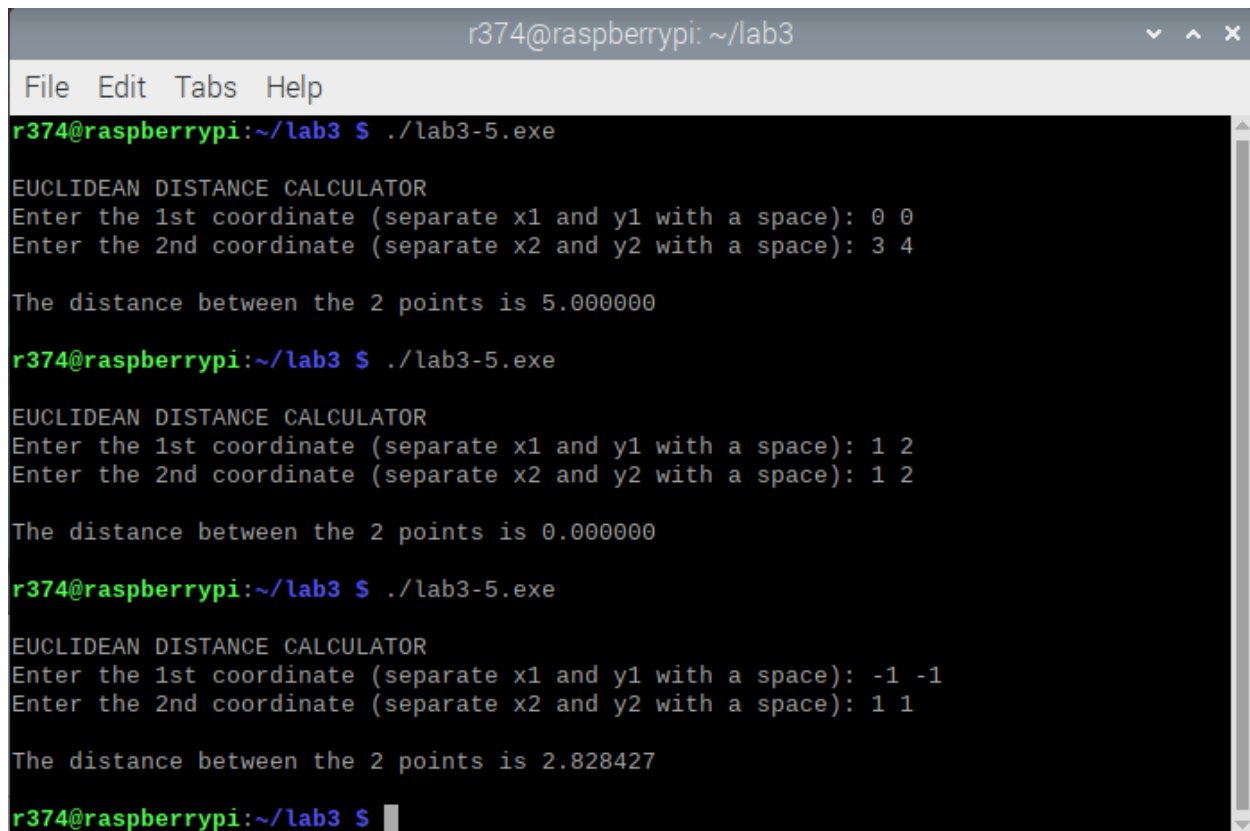


Figure 10: Distance Program Flowchart



A screenshot of a terminal window titled 'r374@raspberrypi: ~/lab3'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows three separate runs of a program named './lab3-5.exe'. Each run displays the text 'EUCLIDEAN DISTANCE CALCULATOR' followed by prompts for two coordinates. The first run uses coordinates (0, 0) and (3, 4), resulting in a distance of 5.000000. The second run uses coordinates (1, 2) and (1, 2), resulting in a distance of 0.000000. The third run uses coordinates (-1, -1) and (1, 1), resulting in a distance of 2.828427. The prompt 'r374@raspberrypi:~/lab3 \$' is shown at the end of each run, with a cursor following the last one.

```
r374@raspberrypi: ~/lab3
File Edit Tabs Help
r374@raspberrypi:~/lab3 $ ./lab3-5.exe
EUCLIDEAN DISTANCE CALCULATOR
Enter the 1st coordinate (separate x1 and y1 with a space): 0 0
Enter the 2nd coordinate (separate x2 and y2 with a space): 3 4

The distance between the 2 points is 5.000000

r374@raspberrypi:~/lab3 $ ./lab3-5.exe
EUCLIDEAN DISTANCE CALCULATOR
Enter the 1st coordinate (separate x1 and y1 with a space): 1 2
Enter the 2nd coordinate (separate x2 and y2 with a space): 1 2

The distance between the 2 points is 0.000000

r374@raspberrypi:~/lab3 $ ./lab3-5.exe
EUCLIDEAN DISTANCE CALCULATOR
Enter the 1st coordinate (separate x1 and y1 with a space): -1 -1
Enter the 2nd coordinate (separate x2 and y2 with a space): 1 1

The distance between the 2 points is 2.828427

r374@raspberrypi:~/lab3 $
```

Figure 11: Test Case Outputs for Distance Calculator Program

## Conclusion

After completing all of the programming assignments for this lab, I have learned a lot of things about the intricacies of program development. Using flowcharts and pseudocode helped me visualize the structures needed to create the programs. Using some of the coding tips provided in the lecture slides helped me identify bugs in the code a lot faster. In addition, I learned that when using the '<math.h>' library in a code, the "gcc" compilation command must include a "-lm" at the end of it in order for the terminal to recognize math functions like 'sqrt.'

In the future, I would like to further develop my knowledge of the C programming language with fewer but tougher program assignments in this lab class. Focusing on five different ones like in this lab, did not exactly help me internalize many of the techniques I learned.

## Images of the Source Code Files

### Lab3-1.c (GCD Calculator)

```
/*      Name: Reta Gela
      Created: September 14, 2023
      Summary: A C program that finds the GCD of 2 numbers inputted by the user*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main (void)
{
    // declaring variables
    int n, m, q1, q2, GCD, i, j;
    // receives 2 input values from user
    printf("\nWELCOME TO THE GCD CALCULATOR!\n");
    printf("\nEnter the first number: ");
    scanf("%d", &n);
    printf("Enter the second number: ");
    scanf("%d", &m);
    // calculates the GCD of the 2 inputs
    }
    if(n > m) {
        j = m;
    }
    else {
        j = n;
    }
    for(i = 1; i < j; i++) {
        q1 = n % i;
        q2 = m % i;
        if(n == 0 && m == 0) {
            GCD = 0;
            break;
        }
        if(q1 == 0 && q2 == 0) {
            GCD = i;
        }
    }
    // displays the GCD to the user
    printf("\nThe GCD is %d\n\n", GCD);
    return 0;
}
```

### Lab3-2.c (Multiplication Table Generator)

```
/*      Name:      Reta Gela
      Created:     September 14, 2023
      Summary:     A C program that displays the multiplication table of an
                  integer entered by the user*/

#include <stdio.h>

int main (void)
{
    //      declare variable n
    int n;
    //      receives input value from user
    printf("MULTIPLICATION TABLE GENERATOR\n");
    printf("Please enter a positive integer: ");
    scanf("%d", &n);
    //      calculates and displays multiplication table
    for (int x = 1; x <= 10; ++x) {
        printf("%d * %d = %d \n", n, x, n * x);
    }
    return 0;
}
```

### Lab3-3.c (Palindrome Detector)

```
/*      Name: Reta Gela
      Created: September 14, 2023
      Summary: A C program that identifies if the number entered by the
              user is a palindrome or not */

#include <stdio.h>

int main() {
    //declaring variables
    int n, rev = 0, p, y;

    //receiving the test number from user
    printf("\nPALINDROME DETECTOR\n");
    printf("Enter an integer: ");
    scanf("%d", &n);
    p = n;
    // reversed integer is stored in "rev" variable
    while (n != 0) {
        y = n % 10;
        rev = rev * 10 + y;
        n /= 10;
    }
    // prints out whether number is a palindrome or not based on the
    // reversed integer (rev) equaling the original number (x)
    if (p == rev) {
        printf("\n%d is a palindrome\n\n", p);
    }
    else {
        printf("\n%d is not a palindrome\n\n", p);
    }
    return 0;
}
```

### Lab3-4.c (Triangle Pattern Generator)

```
/*      Name: Reta Gela
      Created: September 14, 2023
      Summary: A C program that prints an equilateral triangle with
               asterisks (*) of a height specified by the user */

#include <stdio.h>

int main (void)
{
    // declaring the variables
    int x, y, i, h;
    //receives the height value from user and assigns to 'h'
    printf("Enter the height of the triangle: ");
    scanf("%d",&h);
    //      formulates the triangle level by level until height is reached
    printf("\n");
    for(i=1;i<=h;i++) {
        for(x = 1;x <= h-i;x++) {
            printf(" ");
        }
        for(y = 1;y <= ((2*i)-1);y++) {
            printf("*");
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

### Lab3-5.c (Euclidean Distance Calculator)

```
/*      Name: Reta Gela
      Created: September 14, 2023
      Summary: A C program that calculates the Euclidean distance between
              2 points denoted by their respective coordinates (x1, y1)
              and (x2, y2) which are entered by the user*/

#include <stdio.h>
#include <math.h>

int main (void)
{
    // declaring the variables
    int x1, x2, y1, y2;
    float xp, yp, d;

    //receiving the coordinate points from the user
    printf("\nEUCLIDEAN DISTANCE CALCULATOR\n");
    printf("Enter the 1st coordinate (separate x1 and y1 with a space): ");
    scanf("%d %d", &x1, &y1);
    printf("Enter the 2nd coordinate (separate x2 and y2 with a space): ");
    scanf("%d %d", &x2, &y2);

    //calculating the distance
    xp = (x2 - x1)*(x2 - x1);
    yp = (y2 - y1)*(y2 - y1);
    d = sqrt((xp + yp));

    //displaying the calculated distance
    printf("\nThe distance between the 2 points is %f\n\n", d);
    return 0;
}
```