Course: Programming Fundamental – ENSF 337
Lab #: Lab 1
Instructor: Mansouri Habibabadi
Student Name: Drew Hengehold
Lab Section: B01
Date submitted: Sept 20, 2022
UCID: 30151823

Lab1_Excersize_B

```
/*
* File Name: lab1exe_B.c
* Assignment: Lab 1 Exercise B
* Lab section: B01
* Completed by: Drew Hengehold
* Development Date: Sept 16, 2022
*/

#include <stdio.h>
int main()
{
    double num1 = -34.5;
    double num2 = 98.7;
    double sum; // sum of num1 and num2
    double sumSquared; // the square of num2 plus num2

     // 1) Add the two numbers and store the result in the
     variable 'sum'
    sum = num1 + num2;

     // 2) Compute the square of the sum and store the result
     in the variable 'sumSquared'
    sumSquared = sum*sum;

     // Use the variable 'sum' (computed above) for this
     computation

    printf( "The sum squared is: %lf \n", sumSquared);

     // 3) Now double the sum squared value and store the
     result in 'sumSquared'
    sumSquared *=2;

    printf( "The sum squared is now: %f \n", sumSquared);
    return 0;
}
```

```
1 error generated.
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % gcc  lab1exe_B.c
error: error reading 'lab1exe_B.c'
1 error generated.
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % ls
lab1exe_B.c             lab1exe_B.c.xcodeproj
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % cd lab1exe_B.c
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % ls
lab1exe_B.c
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % gcc  lab1exe_B.c
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % ls
a.out           lab1exe_B.c
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % ./a.out
The sum squared is: 4121.640000
The sum squared is now: 8243.280000
drewhengehold@Drews-MacBook-Pro lab1exe_B.c % clear

drewhengehold@Drews-MacBook-Pro lab1exe_B.c % ./a.out
The sum squared is: 4121.640000
The sum squared is now: 8243.280000
drewhengehold@Drews-MacBook-Pro lab1exe_B.c %
```

Above is the screenshot of the terminal output

Lab1 Excersize C

```
double z = 0;
double x = 2.5;
double y = -1.5;
int m = 18;
int n = 4;
```

***Brackets [] are used to specify which variable is being used**

A) `z = x + n * y – (x + n) * y;`

- Parenthesis has first precedent
  `(2.5[x] + 4[n]) = 6.5`
- Multiplication has second precedent, occurring from left to right.
  `4[n] * -1.5[y] = -6.0` then `6.5 * -1.5[y] = -9.75`
- Addition has the final precedent, again occurring from left to right.
- `2.5[x] + -6.0 = -3.5.` Then `-3.5 - -9.75 = 6.25.`

B) `z = m / n + m % n;`

- Division and modulus have same precedent so the equation will process left to right.
  `18[m] / 4[n] = 4` (decimal truncated) then `18[m] % 4[n] = 2` (the remainder)
- Then addition has the next precedent
  `4 + 2 = 6` (this is an integer)

C) `z = n / m + n % m;`

- Division and modulus have same precedent so the equation will process left to right.
  `n / m = 0.0` (decimal truncated) then `n % m = 4.0` (the remainder)
- Then addition has the next precedent
  `0.0 + 4.0 = 4.0` (this is a double)

D) `z = 5 * x – n / 5;`

- Multiplication and division have first precedent
  `5 * 2.5[x] = 12.5` then `4[n] / 5 = 0.0` (decimal truncated)
- Then addition has the next precedent
  `12.5 + 0.0 = 12.5` (this is a double)

E) `z = 1 – (1 – (1 – (1 – (1 – n))));`

- The parenthesis has first precedent, going from inner most parenthesis to outer most parenthesis.

(1 – 4[n]) = -3 then (1 - -3) = 4 then
(1 – 4) = -3 then (1 - -3) = 4

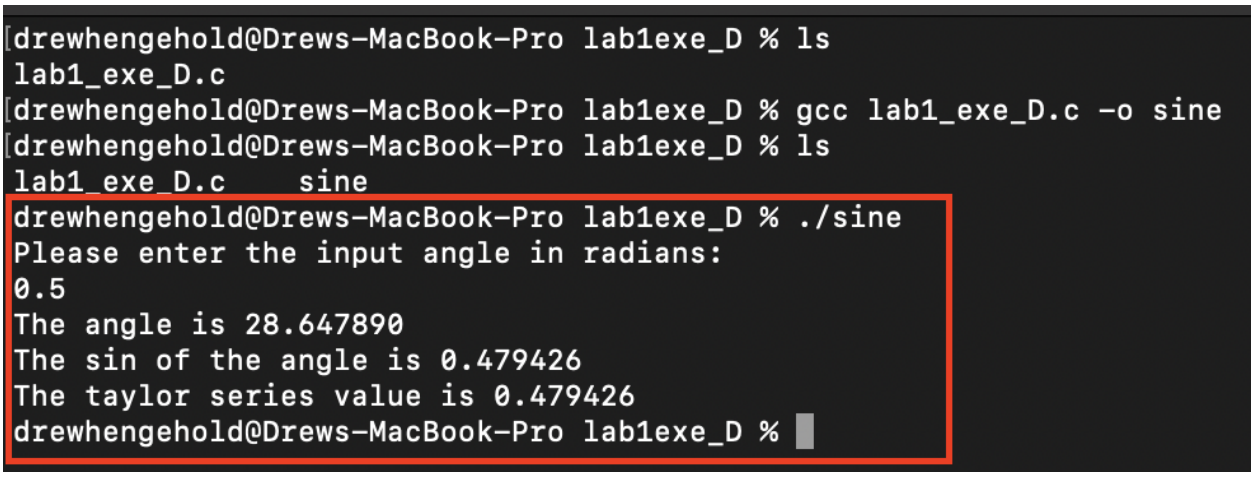
- Then addition has the next precedent
  1 – 4 = -3 (this is an integer)

F) z = sqrt(sqrt((double)n));

- The parenthesis has first precedent
  ((double)4[n]) = 4.0 **This casts 4 (an integer) into 4.0 (a double)**
- The second set of parenthesis then has precedent
  sqrt(4.0) = 2.0 **the sqrt is the math function for square root of the double 4.0 = 2.0**
- The final sqrt math function then proceeds
  Sqrt(2.0) = 1.4142 (this is a double)

Lab1 Excersize D

```
/*
* File Name: lab1exe_D.c
* Assignment: Lab 1 Exercise D
* Lab section: B01
* Completed by: Drew Hengehold
* Development Date: Sept 16, 2022
*/

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

int main() {

    double angle_radian, angle_degree, angle_sin;
    printf("Please enter the input angle in radians:\n");
    scanf("%lf", &angle_radian);

    angle_degree = angle_radian*180/M_PI;
    printf("The angle is %lf\n", angle_degree);

    angle_sin = sin(angle_radian);
    printf("The sin of the angle is %lf\n", angle_sin);

    angle_sin = angle_radian -
(((pow(angle_radian,3))/(3*2*1)))+((pow(angle_radian,5))/(5*4*
3*2*1))-((pow(angle_radian,7))/(7*6*5*4*3*2*1));
    printf("The taylor series value is %lf\n", angle_sin);
    return 0;
}
```



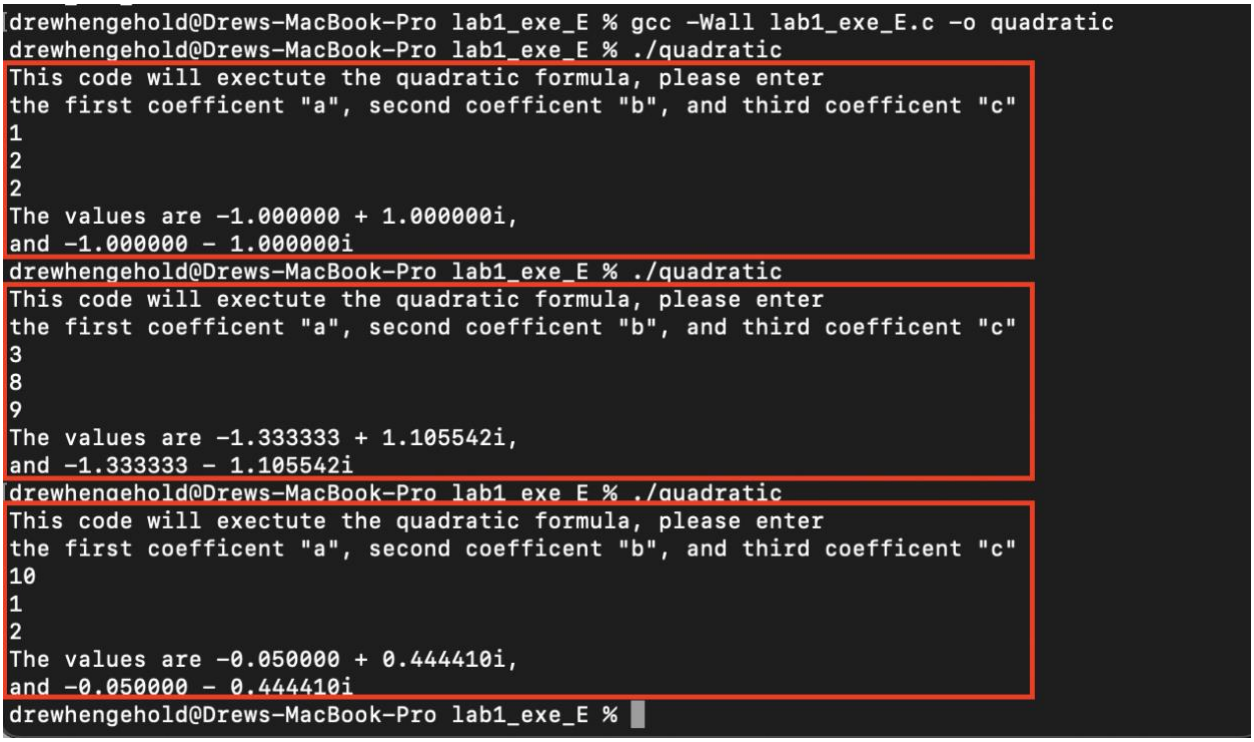
The above is Terminal Output of program

Test Output Letter	Angle Calculated	Sin of the input	Taylor Series approximation
A (0)	0.0	0.0	0.0
B (0.5)	28.648	0.479	0.479
C (1.0)	57.296	0.841	0.841
D (2.5)	143.239	0.598	0.588

The above is a table of the terminal outputs A – D

Lab1 Excersize E

```
/*
* File Name: lab1exe_E.c
* Assignment: Lab 1 Exercise E
* Lab section: B01
* Completed by: Drew Hengehold
* Development Date: Sept 16, 2022
*/

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

int main() {
    double a, b, c;

    printf("This code will exectute the quadratic formula,
please enter\nthe first coefficent \"a\", second coefficent
\"b\", and third coefficent \"c\"\n");
    scanf("%lf%lf%lf", &a, &b, &c);

    if(0 > (pow(b,2)-4*a*c)){
        printf("The values are %lf + %lfi,\nand %lf - %lfi\n",
(b*-1)/(2*a), sqrt(fabs((pow(b,2)-4*a*c)))/(2*a), (b*-
1)/(2*a), sqrt(fabs((pow(b,2)-4*a*c)))/(2*a));
    }
    else
    {
        printf("The first value is %lf\n The second value is
%lf\n", ((b*-1)+sqrt(pow(b,2)-4*a*c))/(2*a), ((b*-1)-
sqrt(pow(b,2)-4*a*c))/(2*a));
    }

    return 0;
}
```



Above is the screenshot of the 3 outcomes of using different coeffects in terminal

Coefficient A	Coefficient B	Coefficient C	Root 1	Root 2
1	2	2	-1.0 + 1.0i	-1.0 - 1.0i
3	8	9	-1.33 + 1.11i	-1.33 - 1.11i
10	1	2	-0.05 + 0.44i	-0.05 - 0.44i

Above is the table of the values performed in the screenshot and their outputs