Purpose of the lab:
Write your own arcsin, arccos, arctan, log functions to compare the result in <math.h>. We need to check the comparison results and analyze it to the writeup.pdf at the end.

Command-line option:

-a run all tests compare with <math.h>
-s only runs arcsin and compares with <math.h>'s arcsin
-c only runs arccos and compares with <math.h>'s arccos
-t only runs arctan and compares with <math.h>'s arctan
-l -t only runs log and compares with <math.h>'s log

Files:
Mathlib.c: define arson,arcos, arctan, exp, and log functions
Mathlib-test.c: the main function, create command-line options and print out results comparison between math.h and mathlib.c functions.

Pesudocode:(x represents input)
Arcsin:
ew terms, then we get:

$$\arcsin(x) = x + (\frac{1}{2})\frac{x^3}{3} + (\frac{1 \times 3}{2 \times 4})\frac{x^5}{5} + (\frac{1 \times 3 \times 5}{2 \times 4 \times 6})\frac{x^7}{7} + O(x^9).$$

Calculate the numerator and denominator for each term, and add it up:
Numerator = x, denominator = 1, sum = x, term = x, power = 3, count = 0

When power is smaller than EPSILON, power +=2{
Count = count + 1
int n = 1
    int d = 2
    for(int i = 1; i < count; i++){
        n *= (n + 2)
        d *= (n + 2)
    }this part find (1x3x5x7…../ 2x4x6x8……) for each term

numerator *= x * x * n
    denominator = power * d
    term = numerator / denominator
    sum += numerator / denominator
Return sum
}

Arccos:
arcsin(M_pi/2 - arcsin(x))

Arctan:
Variable = squareroot of (x square + 1)
arcsin (x/variable)

Log:
Create Exp function first:

```
double Exp(double x) {
    double term = 1, sum = 1;
    for (int k = 1; Abs(term) > EPSILON; k += 1) {
        term *= x / k;
        sum += term;
    }
    return sum;
}
```

a = 1
b = Exp(a)
while (Abs(b - x) > EPSILON) {
a = a + (x - b) / b
b = Exp(a)
}
return a

Update:
Rewrote arcsin formula (a * (pow(2*i-1, 2) * pow(x,1) / (2*i)*(2*1+1)))
And then add up all terms until last term > Epsilon

However, when ¾ and -¾, the difference got bigger, so I created inverse method:
Create for loop when find sin(y)= x, so sin(y) - x not equal to 0 keep looping:
Split -pi and pi, find the middle, if middle not equal to sin(y), then resize to left part or right part, and find the new middle and continue the same thing
Until when a > Epsilon, the loop stop