PRE-LAB ANSWERS

Part 1:

1) Pseudo code for e^x approximating in a nested for loop shown in **DESIGN** (below).
2) Pseudo code for printing the output for e^x shown in **DESIGN** (below).

Part 2:

1) getopt() returns the next option character on the command line if there is one. If there isn't another character it returns -1. It may also return a ? if there is an unrecognized option and it stores into an external variable. But mainly it will return the next character and -1.
2) In this specific situation I opted to use neither bool or enum because it was simpler to just call my math functions inside of the switch statement I used in my getopt(). I believe my choice improves readability and requires less code and is therefore the best option for this particular program.
3) Pseudo code for main function shown in **DESIGN** (below).

DESIGN (pseudo code)

```
Includes
Defines (PI, OPTIONS)

Prototypes

int main(int argc, char **argv) {
        If (argc == 2 and length of argv[1] == 2) {
                while (more arguments) {
                        Switch {
                                case 'a':
                                        Call all my functions
                                case 's':
                                        Call Sin function
                                case 'c':
                                        Call Cos function
                                case 't':
                                        Call Tan function
                                case 'e':
                                        Call Exp function
                                default:
                                        print(error message)
                        }
                }
        }
        Else {
                print(error message)
        }
```

```
        Return 0;
}


Int Sin() {
        printf(column headers)
        variables (my_ans, lib_ans)
        For(double x = -2pi; x <= 2pi; x += pi/16) {
                my_ans = Horner approximation for x
                lib_ans = math.h's sin(x)
                printf(the outputs in the given format)
        }
        Return 0
}

Int Cos() {
        printf(column headers)
        variables (my_ans, lib_ans)
        For(double x = -2pi; x <= 2pi; x += pi/16) {
                my_ans = Horner approximation for x
                lib_ans = math.h's cos(x)
                printf(the outputs in the given format)
        }
        Return 0
}

Int Tan() {
        printf(column headers)
        variables (my_ans, lib_ans)
        For(double x = -pi/3; x <= pi/3; x += pi/16) {
                my_ans = Horner approximation for x
                lib_ans = math.h's tan(x)
                printf(the outputs in the given format)
        }
        Return 0
}

Int Exp() {
        printf(column headers)
        variables (my_ans, lib_ans, numerator, denominator, part, epsilon)
        For(double x = 0; x <= 9; x += 0.1) {
                my_ans, numerator, denominator, part = 1
                for (int i = 1; part > epsilon; i++) {
                        numerator *= x;
```

```
                denominator *= i;
                part = numerator / denominator;
                my_ans += part;
            }
            lib_ans = math.h's exp(x)
            printf(the outputs in the given format)
        }
        Return 0
}
```