Pre-lab Part 1:

EPSILON = 10^-9

1) Function Exp( input value) {

```
 *Note:The 0th term of the series will always be 1. Since x^0/0! Is always 1.
 As a result, the computation starts with the first term*
variables : result = 1, number_of_terms = 1,  error_term = 1
While step_term is greater than EPSILON term {
  error_term = previous_error_term x (input value/number_of_terms)
  result  = previous_result + error_term
  Increment number_of_terms by 1
}
return result
}
```

2) Function exp_simulation() {

```
 Print “x (tab space)  Exp (tab space)  Library (tab space)  Difference”
 print  “-   -----    ---------     -----------”
 For loop iterating over range of x from 1 to 10 over interval of 0.1: {
   user_defined_value = Exp(x)
   library_value = exp(x)
   Delta = user_defined_value - library_value
   Print x (tab space)+ user_defined value (tab space) + library_value (tab space) +delta
}
```

Pre-lab part 2:

1) The function getopt() returns the next argument flag in the argument list formed when the user inputs the flags. If getopt() iterates through all flags, then it returns -1 which signals the end of the loop.
2) An enum is a better option for this for multiple reasons.  In terms of functionality, it allows the different testing options to be chosen in a mutually exclusive way so only one of the inputted flags is processed. Each testing option can then be thought of as an independent state of the program that must be executed. In terms of reading the code, having user defined types makes it elegant and easier to understand.
3) Code for main function:

***Values for defining type of test using user defined variable “test_type”**

```
Typedef enum {SIN, COS, TAN, EXP, ALL} Test_type;
```

***In main function:**

```
Int main(num_arguments, argument_list_pointer) {
  While input flags don’t equal -1 {
     Variable Test_type selected
      Switch between types of flags {
         If input flag is ‘s’:
                Test_type = SIN
              If input flag is ‘c’:
                Test_type = COS
             If input flag is ‘t’;
                Test_type = TAN
             If input flag is ‘e’:
                Test_type = EXP
             If input flag is ‘a’:
                Test_type = ALL

            }
        Switch between types of test(Test_type) {
          If input flag is SIN:
            Run sin_test()
         If input flag is COS:
            Run cos_test()
        If input flag is TAN:
           Run tan_test()
        If input flag is EXP:
           Run exp_test()
        If input flag is ALL:
           Run sin_test(), cos_test(), tan_test(), exp_test()
        }
   }
  Return 0
}
```

**Program functionality:**
This program implements the functions sin, cos, tan and exp. The approximation are made using fixed term(14 term) Pade approximants for sin, cos and tan. For the exp function, a taylor expansion is used that iterates over enough taylor terms until the x^n/n! term reaches 10^-9. For the sake of ease, a factorial identity was used so that the solution term was computed merely by the addition of each taylor term. The taylor term is determined by its multiplication by the x/n term in every iteration until the taylor term(x^n/n!) reaches 10^-9

***Note:**The approximants provided in the asgn2.pdf lab manual document by the instructor were used in this lab. All approximations are centered around x=0.

They are the following:
**For sin:**

$$\sin(x) \approx \frac{x((x^2(52785432-479249x^2)-1640635920)x^2+11511339840)}{((18361x^2+3177720)x^2+277920720)x^2+11511339840}.$$

Simulated from -2pi to 2pi on intervals on pi/16

---

**For cos**:

$$\cos(x) \approx \frac{(x^2(1075032-14615x^2)-18471600)x^2+39251520}{((127x^2+16632)x^2+1154160)x^2+39251520}.$$

Simulated from -2pi to 2pi on intervals on pi/16

---

**For tan:**

$$\tan(x) \approx \frac{x(x^8-990x^6+135135x^4-4729725x^2+34459425)}{45(x^8-308x^6+21021x^4-360360x^2+765765)}.$$

Simulated from -pi/2 + 0.001 to pi/2-0.001 on intervals on pi/16 since function has undefined value at -pi/2 and pi/2

---

**For exp:**

$$\frac{x^n}{n!} = \frac{x^{n-1}}{(n-1)!} \times \frac{x}{n}.$$

Simulated from 0 to 10 on intervals on 0.1

***Note: The base term for the multiplication in this function was 1 since the first term would be always be 0 in the e^x taylor series.***

---

Main program functionality: Each respective function takes in an input number and returns the output value processed by the function. Each function is simulated by a respective helper function(chosen by inputting flags). The helper function are of the form “func_type_test()” where func_type is either sin,cos, tan or exp. The function follow the following format:

```
Void func_type_test() {
  print “x” (tab space) “function type” (tab space) Library (tab space) Difference
  print  “-                    -----------------                    ---------                    -------------”
  Variables: result_value, library, diff_value
  For loop iterating over respective test domain of each function over respective interval {
    result_value = function_programmer_approx(domain value)
    library = math_library_value(domain value)
    difference_between_both = result_value = library
    print domain value (tab space) result_value (tab space) library (tab space) diff_value
  }
  return nothing
}
```