This lab considers the Bisection method and the Illinois algorithm as well as Newton’s method of finding roots to equations.

1. First, consider the problem of solving the equation \( \tan(x) - \lambda x = 0 \) where \( \lambda \) is some non-zero real number. Show, by looking at the graphs of \( y = \lambda x \) and \( y = \tan(x) \), that there is at least one root in each open interval \((-\pi/2 + n\pi, \pi/2 + n\pi)\) for \( n = \ldots -3, -2, -1, 0, 1, 2, 3, \ldots\).

2. Get copies of the sample program `bisection.c`, and the makefile (Makefile) in one of your subdirectories. Compile and link the program by typing:
   
   `make bisection < cr >`

   Run the program with a variety of choices for \( \lambda \). You can type the value of \( \lambda \) you want on the command line, for example
   
   `bisection -1.56 < cr >`

   Look carefully at the procedure `bisect()` which does most of the work. Under what conditions is its return value nonsense? The program actually works reasonably well for small choices of \( \lambda \) (e.g. -1.56, 2.03, etc.) but can fail badly for large values of lambda (e.g. 10000). Why does this happen?

3. Get copies of the sample program `illinois.c`, and compile and link it. This program finds an approximate root of a cubic equation where you enter the coefficients on the command line. For example, if you want to find an approximate root of \( 6.78 - 1.23x + 1.09x^2 + 0.99x^3 \) you could type
   
   `illinois 6.78 -1.23 1.09 0.99 < cr >`

   and all three methods (Bisection method, Illinois algorithm, and Newton’s method) would be applied to find an approximate root of \(-2.595906\). Try this with several cubic polynomials. In general, Newton’s method converges more quickly as long as the seed is not too far from the root and the derivative is bounded away from zero near the root.

**Assignment** Write a program which numerically finds at least one root of the equation

\[
b - a \times x + \sin(x) = 0.0
\]

to eight places (i.e. 0.00000001) of accuracy where \( a > 0.0 \) and \( b > 0.0 \). There will always be at least one root, why? Your program should let \( a \) and \( b \) be typed on the command line

`my_program a b < cr >`

Your program should use and compare the Bisection method and Illinois algorithm to Newton’s method. For all methods, it should display the root \( r \) found to eight places of accuracy and display the value of \( f(r) \) (which, hopefully, is close to zero) to ten places of accuracy, where

\[
f(x) = b - a \times x + \sin(x)
\]