Note: could add error-checking to ensure that the values given to us for xEvalPos and xEvalNeg are correct – test if f(xEvalPos) > 0, and test if f(xEvalNeg) < 0. If they are reversed, you can swap them; if both are positive or both are negative, throw an exception (bisection is not guaranteed to work with those starting points)

\* Algorithm:

\* Given f(x), xEvalPos, xEvalNeg, precision

\* keepGoing <-- true

\* while (keepGoing)

\* // calculate a better approximation

\* guess <-- (xEvalPos + xEvalNeg) / 2

\* fAtGuess <-- f(guess)

\* if (fAtGuess = 0)

\* keepGoing <-- false

\* else if (fAtGuess > 0)

\* xEvalPos <-- guess

\* else

\* xEvalNeg <-- guess

\*

\* // are we close enough?

\* error <-- | xEvalPos - xEvalNeg |

\* if error <= precision

\* keepGoing <-- false

\* return guess

Could change error from amount of error to relative error; could also test (against another desired precision passed in) whether f(x) is close enough to 0 (currently only testing if x1 and x2 are close enough together), and then could add loop counter to make sure we don’t have an infinite loop.