CSCI 2202 Introduction Programming with Applications
Lab 8: Functions, Monte Carlo Simulation - 2 **due: 23 Feb 2020**

- In this lab you will estimate probabilities using a Monte Carlo simulation.
- Implement the bisection algorithm for finding the roots (zeros) of a function.

(1) (a) Write and test a Boolean function: `hasComplexRoots(a, b, c)`, that returns `True` if the quadratic $ax^2+bx+c=0$ has em complex roots and `False` otherwise.
Note: The roots of the quadratic above are given by $x = \dfrac{-b \pm \sqrt{b^2-4ac}}{2a}$

(b) Using the function `hasComplexRoots(a, b, c)`, to compute the *probability* $P_1(n)$ (the Monte Carlo estimate) that the quadratic has complex roots, assuming that the coefficients are drawn from a `uniform` random distribution on $[0, 1]$, (using `random.random()`). Run the simulation for $n = 1000$.
Repeat the experiment to compute $P_2(n)$, where the coefficients are chosen from a normal distribution (`random.normalvariate(0, 1)`) with mean `mu` $= 0$ and std. deviation `sigma` $= 1$.

(2)
- Implement a function `bisection`, that can operate on an arbitrary function. Your program should accept: a function (f - to find the roots/zeros of); two endpoints ($x0, x1$); a tolerance limit (ϵ) and the max. number of iterations `m` as input parameters. The function `bisection` should output the final approximation to the root (zero) and the number of iterations it took to find the root. The program should make sure that the initial interval is acceptable for this method (*i.e. (i)* it should be greater than the tolerance (ϵ) and *(ii)* must contain a zero of f (*i.e.* $f(x0) \cdot f(x1) < 0$). Recall that the max. iterations n to achieve an error bound of:
$\dfrac{x_1 - x_0}{2^{n+1}}$ which worked out (see class notes) to be: $n \geq \dfrac{\ln(x_1 - x_0) - \ln(2 \cdot \epsilon)}{\ln 2}$
Set this value as your max. iterations m .

- The *bisection method* for finding roots was covered in class. The algorithm is in the notes on-line.

- **Note1:** Define Python functions for function evaluations in the problems:
Note2: You can return two values from a python function using
`return rootApprox, iterCount` as the final line in the function. Make sure to pick up both values in the calling code as:
`xr, numIter = bisection(a, b, tol, f)` for example.

- Use the `bisection(f, x0, x1, tol, m)` to compute *(i)* the positive root of $f(x) = x^2 - x - 1$. The positive root of this equation is known to lie in the interval [1,2]. Use a tolerance of f 10^{-8}. *(ii)* Repeat this exercise for the negative root:. Search in $[-1, 0]$. (The solution to *(i)* is the *Golden Ratio*: Φ the limit of

the ratio of consecutive Fibonacci numbers. The solution to *(ii)* is known to be $-1/\Phi$).

- Use your program to compute the roots of $f_1(x) = x^3 - 3x - 1$ on [0, 1], $f_2(x) = x^3 - 2\sin(x)$ on [0.5, 2], $f_3 = \tan(x) - 2x$ on [0, 2].
  Start by plotting the functions to find a reasonable interval to bracket the root and print x, $\mid f(x) \mid$ and the number of iterations required for convergence.