Curtin University
DEPARTMENT OF MATHEMATICS & STATISTICS

ACCELERATED MATHEMATICS UNITS MATH1017 AND MATH1021

Lab Sheet 2

The aim today is to write some code that returns the real roots (or zeros) of a quadratic, if it has any, and `fail` (indicating no real roots), otherwise.

Ingredients: `proc`, `if`.

You will probably need to do this in stages, perhaps something like the following.

1. Start with a line:

   ```
   a := 1; b := 5; c := 6;
   ```

   These represent the coefficients of your quadratic.

2. Enter an assignment statement where `discr` representing the discriminant is assigned an expression in `a`, `b` and `c`.

3. Enter two assignment statements for the roots `root1` and `root2`.
   This works because the above coefficients are of a polynomial with two real roots (really zeros).

4. Nestle some of your code lines so far in an `if` statement to deal with the different cases: 2 distinct real roots, 2 equal real roots, no real roots.
   Test all your current code by changing your values for `a`, `b` and `c` on your start line and re-executing all your code.

5. You might like to add a `print` message to say what case you have. Strings are done as character strings between pairs of double quotes. Try: `print("Hello\n");`
   The `\n` is a newline character.

6. Now nestle all the lines of your current code in a `proc`, except that your start line will disappear, and your `proc` line will have the arguments `a`, `b` and `c`, i.e. you will have something like:

   ```
   proc(a, b, c)
    local discr, root1, root2;
    .
    .
   end;
   ```

   and you will need to have some `return` lines. One such line might look like:

   ```
   return [root1, root2]
   ```

   A sequence of objects in square brackets is a *list*. You might like to do: `?list`
   to find out about them. If you stick `quadrts :=` in front of `proc` on your `proc` line, you should be able to do

   ```
   quadrts(1, 5, 6);
   ```

and have it print: `Quadratic has two real roots`
and have it return `[-2, -3]`, or perhaps you'd prefer it returned a *set*.

7. **Further refinements.** Change your `proc` line so that it takes a *polynomial*. You will then need to include `a, b, c` in the `local` line, and use `coeff` or `coeffs` to determine `a, b, c` from your *polynomial*, which will be `p` if you changed your `proc` line to: `proc(p)`

8. Save the text of your `proc` as a textfile on your USB stick. Everyone has one of those with them these days, don't they? If you don't, perhaps save firstly to the `C:` drive and then email it to yourself, and load it onto a USB stick at home, and have it with you next time.