What is the significance of the values n and y in the example? Is this some modification of the nth root algorithm?

Edit:

From the book (page 21-25) it seems y is the current guess and x is the number we are trying to find the fifth root of.

What level of decimal precision will be required by a guess in order to be considered a correct answer?

Anything that is within $1E^{-6}$ should be good enough.

I modified my code to use percentage and now it's not playing nice with lengthy decimals, particularly with root 5 of 2.

Inspecting the following variables should return true to make my program work:
guess is 1.1486983550
x is 2
(/ (^ guess 5) x) is 1.0000000000
(= (/ (^ guess 5) x) 1.0000000000) is #f

any suggestions?

Subject: Re: Problem 4
Posted by lusth on Fri, 26 Aug 2016 11:17:33 GMT
View Forum Message <> Reply to Message

2.0000 cannot be represented exactly in floating point, so, as a general rule, never use equality to compare calculated real numbers. Stop when things are "close enough".

Subject: Re: Problem 4
Posted by padietl on Fri, 16 Sep 2016 18:43:12 GMT
View Forum Message <> Reply to Message

Should we accommodate negative numbers?

Subject: Re: Problem 4
Posted by lusth on Fri, 16 Sep 2016 18:48:30 GMT
View Forum Message <> Reply to Message

If you want to, but I'll only test non-negative numbers.