```
;; This is my canvas. This is my art.
;;
;; Assignment information
;; ---
;; Class : Evolutionary Computing
;; Assignment : Make a final assignment that uses
evolutionary computation to its best potential.
;; Student : Nirman Dave
;;
;; Program information
;; ---
;; Name : Evol_4
;; File : core.clj
;; Version : 1.0
;;
;; Description : The idea is to take values of light
intensity given to a plant (x variable) and chlorophyll
generated (y variable) and to
;;                 evolve a formula that models the two
variables. Then use the same formula as an error
function to evolve a set of values
;;                 that correspond to light intensities.
This list of light intensitites can then be used by
green houses or farmers to automate
;;                 an efficient plant growth cycle using
changing light intensities.
;;
;;                 The following plant is considered for
the following conditions:
;;                 Plant considered: Soybean plant
;;                 Light intensities: Ranging from
90w/(m^2) to 220w/(m^2)
;;                 Plant environment temperature: Room
temperature maintained at 22*C
;;
;;                 The paper referenced for this data
collection is as cited below:
;;                 J. Elizabeth M. Ballantine and B. J.
Forde. "The Effect of Light Intensity and Temperature
on Plant Growth and Chloroplast
;;                 Ultrastructure in Soybean" American
Journal of Botany 57.10 (1970): 1150-1159. November
1970. Web. 27 April 2016.
;;
;; File use : This file is used to take the target data
from the research paper, and evolve a formula that
models the correlation.
;; Language : Clojure
;; Requirements : To be run using Gorilla-REPL (if not
using Gorilla-REPL comment the mentioned lines out)

;; Defining namespace and other requirements
(ns evol-4.core
  (:require [gorilla-plot.core :as plot])
  (:require [clojure.zip :as zip]))

;; Target data aquired from the Ballantine and Forde's
research paper (1970, p.1157)
;; The data is represented in the fashion : [light
intensity(w/(m^2)) chlorophyll growth(mg/g)]
(def target-data
  (list [90 4.0]
        [100 3.842857143]
        [110    3.685714286]
        [120    3.528571429]
        [130    3.371428571]
        [140    3.214285714]
        [150    3.057142857]
        [160    2.9]
        [170    2.742857143]
        [180    2.585714286]
        [190    2.428571429]
        [200    2.271428571]
        [210    2.114285714]
        [220    1.8]]))

;; Lists the potential functions that the AI can use
during evolution
(def function-table (zipmap '(+ - * pd)
                            '(2 2 2 2)))

(defn pd
  "Also known as protected division. Avoids dividing by
zero."
  [num denom]
  (if (zero? denom)
    0
    (/ num denom)))

(defn absolute
  "A funtion that returns the absolute value of a
number n."
  [number]
  (max number (- number)))

(defn todo-function
  "A function that returns a randomly selected function
from the function-table."
  []
  (rand-nth (keys function-table)))

(defn todo-terminal
  "Returns a random number from the todo-code."
  []
  (rand-nth (list 'x (- (rand 10) 5))))

(defn todo-code
  "Returns a randomized code that will be evaled and
mutated."
  [number]
  (if (or (zero? number)
          (zero? (rand-int 2)))
    (todo-terminal)
    (let [f (todo-function)]
      (cons f (repeatedly (get function-table f)
                          #(todo-code (dec
number)))))))

(defn misprint
  "Basically just an error function. Looks how far the
computer generated
  result is from the actual result."
  [correlation]
  (let [value-function (eval (list 'fn '[x]
correlation))]
    (reduce + (map (fn [[x y]]
                    (absolute
                          (- (value-function x) y)))
                      target-data))))

(defn todo-codePopulation
  "Returns the size of the todo-code."
  [c]
  (if (seq? c)
    (count (flatten c))
    1))

(defn at-index
  "Defines the index to tweak."
  [tree point-index]
  (let [index (mod (absolute point-index) (todo-
codePopulation tree))
        zipper (zip/seq-zip tree)]
    (loop [z zipper i index]
      (if (zero? i)
        (zip/node z)
        (if (seq? (zip/node z))
          (recur (zip/next (zip/next z)) (dec i))
          (recur (zip/next z) (dec i)))))))

(defn insert-at-index
  "Inserts an item at a given index."
  [tree point-index new-subtree]
  (let [index (mod (absolute point-index) (todo-
codePopulation tree))
        zipper (zip/seq-zip tree)]
    (loop [z zipper i index]
      (if (zero? i)
        (zip/root (zip/replace z new-subtree))
        (if (seq? (zip/node z))
          (recur (zip/next (zip/next z)) (dec i))
          (recur (zip/next z) (dec i)))))))


(defn mutate
  "First mutation algorithm that inserts a code at
point i."
  [i]
  (insert-at-index i
                   (rand-int (todo-codePopulation i))
                   (todo-code 2)))

(defn crossover
  "Crossover takes two codes and creates a new one that
is the combination of both."
  [i j]
  (insert-at-index i
                   (rand-int (todo-codePopulation i))
                   (at-index j (rand-int (todo-
codePopulation j)))))

(defn sort-by-misprint
  "Sorts the collection of items by misprint level."
  [population]
  (vec (map second
            (sort (fn [[err1 ind1] [err2 ind2]] (< err1
err2))
                  (map #(vector (misprint %) %)
population)))))

(defn select-x
  "Selects a particular code from the population size
```

```
and returns that."
  [population tournament-size]
  (let [size (count population)]
    (nth population
         (apply min (repeatedly tournament-size #(rand-
int size))))))

(defn goabc
  "Takes an input t and returns it as is. This is to
avoid converting
  a computer generated code into a string."
  [t]
  t)

(defn model-correlation
  "Uses evolutionary computing to model the correlation
between light intensity and chlorophyll growth in the
  Soybean plant, by creating a formula that returns
value y for value x."
  [popsize]
  (println "======================")
  (println "")
  (println "Evolving a formula that models the
correlation between the light intensity and
chlorophyll")
  (loop [generation 0
         population (sort-by-misprint (repeatedly
popsize #(todo-code 2)))]
    (let [best (first population)
          best-error (misprint best)]
      (println "======================")
      (println "Generation:" generation)
      (println "Best error:" best-error)
      (println "Best program:" best)
      (println "     Median error:" (misprint (nth
population
                                                   (int
(/ popsize 2)))))
      (println "     Average program size:"
               (float (/ (reduce + (map count (map
flatten population)))
                          (count population))))
      (if (< best-error 0.47)          ; Over a period of
different trials it was observed that 0.46 was the best
error reachable. Which is acceptable amount of
uncertainty when talking about light intensity values
that range in hundreds.
        (do (println "Success:" best)
          (goabc best))
        (recur
          (inc generation)
          (sort-by-misprint
            (concat
              (repeatedly (* 0.1875 popsize) #(mutate
(select-x population 7)))
              (repeatedly (* 0.75 popsize) #(crossover
(select-x population 7)

(select-x population 7)))
              (repeatedly (* 0.0625 popsize) #(select-x
population 7))))))))))
```

```
nil
```

```
#'evol-4.core/target-data

#'evol-4.core/function-table

#'evol-4.core/pd

#'evol-4.core/absolute

#'evol-4.core/todo-function

#'evol-4.core/todo-terminal

#'evol-4.core/todo-code

#'evol-4.core/misprint

#'evol-4.core/todo-codePopulation

#'evol-4.core/at-index

#'evol-4.core/insert-at-index

#'evol-4.core/mutate

#'evol-4.core/crossover

#'evol-4.core/sort-by-misprint

#'evol-4.core/select-x

#'evol-4.core/goabc

#'evol-4.core/model-correlation
```

```
(model-correlation 1000)
```

```
======================

Evolving a formula that models the correlation between the
light intensity and chlorophyll
======================
Generation: 0
Best error: 7.832623992347864
Best program: (- (pd 1.0763915754993985 x) -2.9802190895543825)
     Median error: 1535.5260594994188
     Average program size: 2.613
======================
Generation: 1
Best error: 7.807655305848459
Best program: (+ (pd 2.9029583828504677 x) 3.0461089207246683)
     Median error: 39.22953669752752
     Average program size: 2.3176823
======================
Generation: 2
Best error: 7.779412833684618
Best program: (- (pd 3.4123905591510404 x) -2.9802190895543825)
     Median error: 11.728472292899035
     Average program size: 1.5264735
======================
Generation: 3
Best error: 7.773496248545199
Best program: (+ (pd (- 2.902759688112111 -3.6166641972039484)
x) 3.0461089207246683)
     Median error: 8.246399880834883
     Average program size: 1.5064934
======================
Generation: 4
Best error: 7.708638469291649
Best program: (+ (pd (- 2.902759688112111 -3.6166641972039484)
x) 2.9029583828504677)
     Median error: 7.8571428569999995
     Average program size: 2.2197802
======================
Generation: 5
Best error: 7.693242075182052
Best program: (+ (pd (- (pd 2.379842228284179
2.379842228284179) -3.6166641972039484) x) (- (pd
2.5786698147744103 x) -2.9802190895543825))
     Median error: 7.952657514863464
     Average program size: 4.092907
======================
Generation: 6
Best error: 7.630908445976267
Best program: (+ (pd (- 2.902759688112111 -3.6166641972039484)
x) (- (pd 3.4123905591510404 x) -2.9802190895543825))
     Median error: 10.491381929068204
     Average program size: 6.3346653
======================
Generation: 7
Best error: 7.545122716703659
Best program: (- (pd (* -3.764392689956464 -4.448512311283496)
x) -2.9802190895543825)
     Median error: 8.256409339485202
     Average program size: 8.337663
======================
Generation: 8
Best error: 7.515652147874956
Best program: (+ (pd (+ (pd 2.902759688112111 x)
3.0461089207246683) x) (- (pd (* -3.764392689956464
-4.448512311283496) x) -2.9802190895543825))
     Median error: 7.8571428569999995
     Average program size: 11.395604
======================
Generation: 9
Best error: 7.48137712704547
Best program: (- (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) -3.6166641972039484)
-0.7358343578310045) x) 3.0461089207246683))
-2.9802190895543825) -3.6166641972039484)))
-2.9802190895543825)
     Median error: 7.803992949204514
     Average program size: 13.903097
======================
Generation: 10
Best error: 7.422271135195165
Best program: (- (pd (- (+ 3.0463331004392185 (-
3.0463331004392185 -2.9802190895543825)) -3.6166641972039484)
```

```
(pd x 3.0463331004392185)) -2.9802190895543825)
     Median error: 7.789799279248252
     Average program size: 16.026974
======================
Generation: 11
Best error: 7.411915800835818
Best program: (+ (pd (- (- (* -3.764392689956464
-4.722881432804133) -2.9802190895543825) -3.6166641972039484)
x) 2.949648592478421)
     Median error: 7.720236677602966
     Average program size: 18.898102
======================
Generation: 12
Best error: 7.276759798790491
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (pd (- -2.9802190895543825 x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) -2.9802190895543825) -2.9802190895543825)
x) (- (pd (* -3.764392689956464 -4.448512311283496) x)
-2.9802190895543825))
     Median error: 7.6800118193417894
     Average program size: 23.148851
======================
Generation: 13
Best error: 7.140639936737069
Best program: (- (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825)
-3.6166641972039484) (* -3.764392689956464 (-
-4.801265780782318 3.8746105294591935))) x) (pd (pd x
1.7169758653117073) x))) -2.9802190895543825)
-3.6166641972039484))) -2.9802190895543825)
     Median error: 7.5554731803455235
     Average program size: 26.177822
======================
Generation: 14
Best error: 7.140639936737069
Best program: (- (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825)
-3.6166641972039484) (* -3.764392689956464 (-
-4.801265780782318 3.8746105294591935))) x) (pd (pd x
1.7169758653117073) x))) -2.9802190895543825)
-3.6166641972039484))) -2.9802190895543825)
     Median error: 7.497469083919025
     Average program size: 30.897102
======================
Generation: 15
Best error: 6.7768341838032
Best program: (- (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- -3.6166641972039484 (*
-3.764392689956464 (- -4.801265780782318 3.8746105294591935)))
x) (pd (pd x 1.7169758653117073) x))) -2.9802190895543825)
-3.6166641972039484))) -2.9802190895543825)
     Median error: 7.462913107095668
     Average program size: 37.646355
======================
Generation: 16
Best error: 6.031028543109783
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (pd (- -2.9802190895543825 x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (-
2.902759688112111 -3.6166641972039484) x) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) -4.981132699549359)
-2.7300237357709998) x) (pd (pd x 1.7169758653117073) x)))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x) (- (pd (* -3.764392689956464
-4.448512311283496) x) -2.9802190895543825))
     Median error: 7.415297945004487
     Average program size: 50.641357
======================
Generation: 17
Best error: 4.6481441109234956
Best program: (+ (pd (- (- (+ 3.0463331004392185 (- (pd (-
-2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 (pd x 1.7169758653117073))
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) -2.9802190895543825) -3.764392689956464)
(* -3.764392689956464 -4.448512311283496)) (- (pd (*
-3.764392689956464 -4.448512311283496) x) -2.9802190895543825))
     Median error: 7.369115117545904
     Average program size: 59.413586
======================
Generation: 18
Best error: 4.121507052853044
Best program: (+ (pd (- (- (+ (pd -4.448512311283496
-2.7609835396098905) (- (pd (- -2.9802190895543825 x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) -2.9802190895543825) -3.764392689956464)
(- (+ 3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484)) (- (pd (*
-3.764392689956464 -4.448512311283496) x) -2.9802190895543825))
     Median error: 7.318575963633083
     Average program size: 65.6014
======================
Generation: 19
Best error: 3.95114071688875
Best program: (+ (pd (- (- (+ 3.0463331004392185 (- (pd (- (pd
-3.6860492534245424 x) x) (- (+ 3.0463331004392185 (- (pd (*
-2.5082900072093848 -3.8547215698893678) (* x x))
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) -2.9802190895543825) -2.9802190895543825)
(- (+ 3.0463331004392185 (- 3.0463331004392185
-2.9802190895543825)) -3.6166641972039484)) (- (pd (*
-3.764392689956464 -4.448512311283496) x) -2.9802190895543825))
     Median error: 7.181840943486653
     Average program size: 83.51449
======================
Generation: 20
Best error: 2.2633964125703656
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (pd (- -2.9802190895543825 x) (- (+
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (-
2.902759688112111 -3.6166641972039484) x) x) (- (+
3.0463331004392185 (pd (- (- (+ (pd (- (+ x x)
-3.6166641972039484) x) (- (pd (- -2.9802190895543825 x) (- (+
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (- (pd
3.02991640155763 x) -3.6166641972039484) x) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) -4.981132699549359)
-2.7300237357709998) x) (pd (pd x 1.7169758653117073) x)))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd (pd
3.0463331004392185 (+ x x)) (+ (pd (- (- (- (pd
3.0463331004392185 (+ x x)) -2.9802190895543825)
-4.981132699549359) -2.7300237357709998) x) (pd (pd x
1.7169758653117073) x))) -2.9802190895543825)
-3.6166641972039484))))) -2.9802190895543825) x) (- (pd (*
-3.764392689956464 -4.448512311283496) x) -2.9802190895543825))
     Median error: 6.756181704001534
     Average program size: 102.87013
======================
Generation: 21
Best error: 2.143221788406124
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (pd (- -2.9802190895543825 x) (- (+
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (-
2.902759688112111 -3.6166641972039484) x) x) (- (+
3.0463331004392185 (pd (- (- (+ (pd (- (+ x x)
-3.6166641972039484) x) (- (pd (- -2.9802190895543825 x) (- (+
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (- (pd
3.02991640155763 x) -3.6166641972039484) x) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ x (pd (pd x 1.7169758653117073) x)))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd (pd
3.0463331004392185 (+ x x)) (+ (pd (- (- (- (pd
3.0463331004392185 (+ x x)) -2.9802190895543825)
-4.981132699549359) -2.7300237357709998) x) (pd (pd x
1.7169758653117073) x))) -2.9802190895543825)
-3.6166641972039484))))) -2.9802190895543825) x) (- (pd (*
-3.764392689956464 -4.448512311283496) x) -2.9802190895543825))
     Median error: 5.763090782297978
     Average program size: 116.31968
======================
Generation: 22
Best error: 0.8701637894989671
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (- (pd (- (- (+ (pd (- (pd
3.02991640155763 -3.764392689956464) 3.0463331004392185) x) (-
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) -2.9802190895543825)) -4.223357958669716)
-2.9802190895543825) (+ (pd (- (- (- (pd 3.0463331004392185 (-
(- (+ (pd (- 3.331358159869156 -3.6166641972039484) x) (-
2.949648592478421 -2.9802190895543825)) -2.9802190895543825)
3.8746105294591935)) -2.9802190895543825) -3.6166641972039484)
(pd (- (pd -3.1870218912857586 x) -4.983887570744702) (-
-4.801265780782318 x))) x) (pd (pd 3.0463331004392185
1.7169758653117073) x))) -2.9802190895543825)
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (- (- (+ x
x) x) -3.6166641972039484) 3.4123905591510404) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) -4.981132699549359)
-2.7300237357709998) x) -2.9802190895543825))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x) (- (pd (* -3.764392689956464
-4.448512311283496) x) -2.9802190895543825))
     Median error: 4.912511816507255
     Average program size: 134.43457
======================
Generation: 23
Best error: 0.8829589769412762
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (- (pd (- (- (+ (pd (- (pd
3.02991640155763 -3.764392689956464) 3.0463331004392185) x) (-
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) -2.9802190895543825)) -4.223357958669716)
-2.9802190895543825) (+ (pd (- (- (- (pd 3.0463331004392185 (-
(- (+ (pd (- 3.331358159869156 -3.6166641972039484) x) (-
2.949648592478421 -2.9802190895543825)) -2.9802190895543825)
```

```
3.8746105294591935)) -2.9802190895543825) -3.6166641972039484)
(pd (- (pd -3.1870218912857586 x) -4.983887570744702) (-
-4.801265780782318 x))) x) (pd (pd 3.0463331004392185
1.7169758653117073) x))) -2.9802190895543825)
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd (- (pd 3.02991640155763 x) x) x) (- (pd (- (pd (- (- (+ x
x) x) -3.6166641972039484) 3.4123905591510404) x) (- (+
3.0463331004392185 (- (pd (pd 3.02991640155763 x) x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) -4.981132699549359)
-2.7300237357709998) x) -2.9802190895543825))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x) (- (pd (* -3.764392689956464
-4.448512311283496) x) -2.9802190895543825))
     Median error: 4.364886983335296
     Average program size: 156.38962
======================
Generation: 24
Best error: 0.31263304733676645
Best program: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (- (pd (- (- (+ (pd (- (pd
3.02991640155763 -3.764392689956464) 3.0463331004392185) x) (-
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) -2.9802190895543825)) -4.223357958669716)
-2.9802190895543825) (+ (pd (- (- (- (pd 3.0463331004392185 (-
(- (+ (pd (- 3.331358159869156 -3.6166641972039484) x) (-
2.949648592478421 -2.9802190895543825)) -2.9802190895543825)
3.8746105294591935)) -2.9802190895543825) -3.6166641972039484)
(pd (- (pd -3.1870218912857586 x) -4.983887570744702) (-
-4.801265780782318 x))) x) (pd (pd 3.0463331004392185
1.7169758653117073) x))) -2.9802190895543825)
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd 1.1900815625130452 x) (- (pd (- (pd (- (- (+ x x) x)
-3.6166641972039484) 3.4123905591510404) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) 1.7169758653117073)
-2.7300237357709998) x) -2.9802190895543825))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x) (- (pd (* -3.764392689956464
-4.448512311283496) x) -2.9802190895543825))
     Median error: 3.1195478447162968
     Average program size: 178.07092
Success: (+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (- (pd (- (- (+ (pd (- (pd
3.02991640155763 -3.764392689956464) 3.0463331004392185) x) (-
(pd (- -2.9802190895543825 x) (- (+ 3.0463331004392185 (- (pd
2.852242671068268 x) -3.6860492534245424))
-3.6166641972039484)) -2.9802190895543825)) -4.223357958669716)
-2.9802190895543825) (+ (pd (- (- (- (pd 3.0463331004392185 (-
(- (+ (pd (- 3.331358159869156 -3.6166641972039484) x) (-
2.949648592478421 -2.9802190895543825)) -2.9802190895543825)
```

```
3.8746105294591935)) -2.9802190895543825) -3.6166641972039484)
(pd (- (pd -3.1870218912857586 x) -4.983887570744702) (-
-4.801265780782318 x))) x) (pd (pd 3.0463331004392185
1.7169758653117073) x))) -2.9802190895543825)
-2.9802190895543825)) (pd (- (- (- (pd 3.0463331004392185 (-
-4.801265780782318 3.8746105294591935)) -2.9802190895543825) (+
(pd 1.1900815625130452 x) (- (pd (- (pd (- (- (+ x x) x)
-3.6166641972039484) 3.4123905591510404) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (- (pd
3.0463331004392185 (+ (pd (- (- (- (pd 3.0463331004392185 (+ x
x)) -2.9802190895543825) 1.7169758653117073)
-2.7300237357709998) x) -2.9802190895543825))
-2.9802190895543825) -3.6166641972039484)))))
-2.9802190895543825) x) (- (pd (* -3.764392689956464
-4.448512311283496) x) -2.9802190895543825))
```

```
(+ (pd (- (- (+ (pd (- (pd 3.02991640155763 x)
-3.6166641972039484) x) (- (- (pd (- (- (+ (pd (- (pd
3.02991640155763 -3.764392689956464) 3.0463331004392185)
x) (- (pd (- -2.9802190895543825 x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825)) -4.223357958669716)
-2.9802190895543825) (+ (pd (- (- (- (pd
3.0463331004392185 (- (- (+ (pd (- 3.331358159869156
-3.6166641972039484) x) (- 2.949648592478421
-2.9802190895543825)) -2.9802190895543825)
3.8746105294591935)) -2.9802190895543825)
-3.6166641972039484) (pd (- (pd -3.1870218912857586 x)
-4.983887570744702) (- -4.801265780782318 x))) x) (pd (pd
3.0463331004392185 1.7169758653117073) x)))
-2.9802190895543825) -2.9802190895543825)) (pd (- (- (-
(pd 3.0463331004392185 (- -4.801265780782318
3.8746105294591935)) -2.9802190895543825) (+ (pd
1.1900815625130452 x) (- (pd (- (pd (- (- (+ x x) x)
-3.6166641972039484) 3.4123905591510404) x) (- (+
3.0463331004392185 (- (pd 2.852242671068268 x)
-3.6860492534245424)) -3.6166641972039484))
-2.9802190895543825))) (* -3.764392689956464
-4.448512311283496)) (pd 3.1171280524240395 (pd x (- (-
(pd 3.0463331004392185 (+ (pd (- (- (- (pd
3.0463331004392185 (+ x x)) -2.9802190895543825)
1.7169758653117073) -2.7300237357709998) x)
-2.9802190895543825)) -2.9802190895543825)
-3.6166641972039484))))) -2.9802190895543825) x) (- (pd
(* -3.764392689956464 -4.448512311283496) x)
-2.9802190895543825))
```