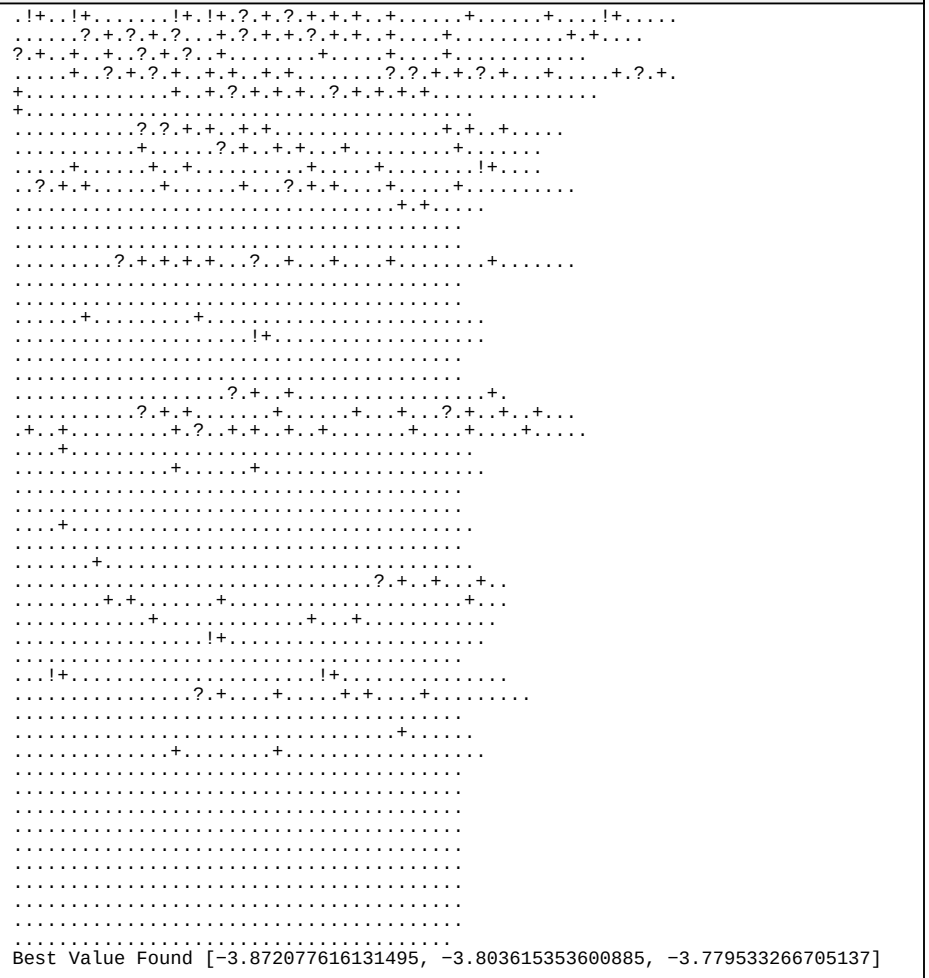
```
"""
Homework 2: The Fonseca Model
Last updated  Sunday, Sep  7 17:51:42 2014
@author: Rahul Krishna
"""

from __future__ import division
import sys,re,random,math,datetime,re,time
import numpy as np
import scipy as sp
sys.dont_write_bytecode = False

# Define some aliases.
rand=random.uniform
randi=random.randint
e=math.e
random.seed()

class simulatedAnnealing:

  def __init__(self):
    pass

  def energy(self,x,emax,emin):
    f1, f2=(1-e**np.sum([(x[z]-1/(np.sqrt(z+1))) for z in xrange(3)])),\
    (1-e**np.sum([(x[z]+1/(np.sqrt(z+1))) for z in xrange(0,3)]))
    ener=f1-f2
    eNorm= (ener-emin)/(emax-emin)
#   print e_norm
    return eNorm

  def neighbour(self,x,xmax,xmin):
    def __new(x,z):
        return xmin+(xmax-xmin)*rand(0,1) if rand(0,1)<0.5 else x[z]
    x_new=[__new(x,z) for z in xrange(3)]
    return x_new

  def do_a_randJump(self, e, en, t, k):
      p=math.e**(-(e-en)/(t*k))<rand(0,1)
#      print p
      return p

  def baselining(self):
    emax=-1;emin=1;
    for x in xrange(int(1e3)):
      x_tmp=[randi(-4,4) for z in xrange(3)]
      ener=(1-e**np.sum([(x_tmp[z]-1/(np.sqrt(z+1))) for z in xrange(3)]))-\
      (1-e**np.sum([(x_tmp[z]+1/(np.sqrt(z+1))) for z in xrange(3)]))

      if ener≥emax:
        emax=ener
      elif ener≤emin:
        emin=ener
    return emax,emin

f=open('log_sa_fonseca.txt','w')
def say(self,x):
  self.f.write(str(x));
  sys.stdout.flush()

# Initial temperature

class main:

  k=1
  kmax=2000

  xmax=4;
  xmin=-4;

  sa=simulatedAnnealing()
  emax, emin = sa.baselining()
  print emax, emin
```



```
  # Initial state and energy
  sb=s=[rand(-4,4) for z in xrange(3)]
  eb=e=sa.energy(s,emax,emin)
  print e

  print 'Initial Best', sb

  for k in xrange(1,kmax):
    #print k
    sn=sa.neighbour(s,xmax,xmin)
    en=sa.energy(sn,emax,emin)
    t=k/kmax

    if en<eb:
      eb, sb=en, sn; sa.say('!')

    if en<e:
      s, e = sn, en; sa.say('+')

    elif sa.do_a_randJump(en,e,k,1e-5): # The cooling factor needs to be reallyl
ow for some reason!!
      s, e=sn, en; sa.say('?')

    sa.say('.')
    if k%40≡0: sa.say('\n')# sa.say(format(sb,'0.2f'))

  sa.say('\n'),sa.say('Best Value Found '), sa.say(sb)

# Print Energy and best value.
  print sb
if __name__≡'main':
  main()
```



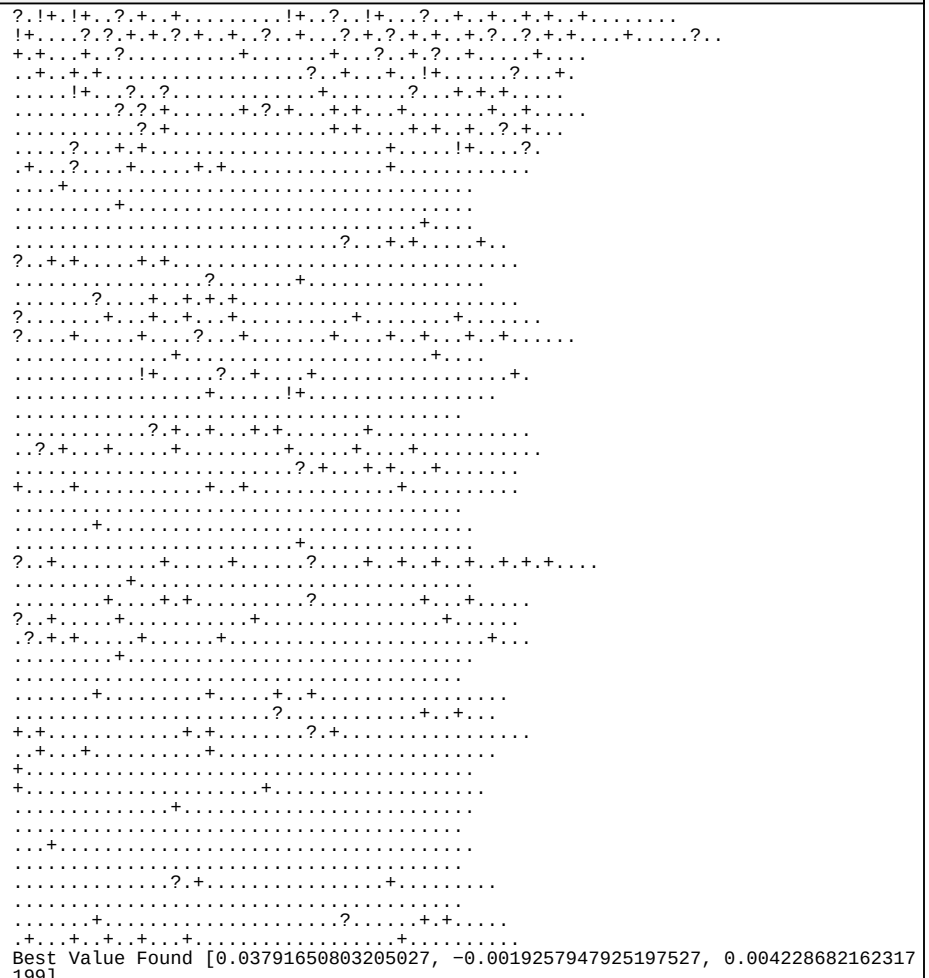
```
Best Value Found [−3.872077616131495, −3.803615353600885, −3.779533266705137]
```

```
# -*- coding: utf-8 -*-

"""
Homework 2: THe Kursawe Model
Last updated  Sunday, Sep  7 17:51:42 2014
@author: Rahul Krishna
"""

from __future__ import division
import sys,re,random,math,datetime,re,time
import numpy as np
import scipy as sp
sys.dont_write_bytecode = False

# Define some aliases.
rand=random.uniform
randi=random.randint
e=math.e
sin=math.sin
sqrt=math.sqrt
random.seed()

class simulatedAnnealing:

  def __init__(self):
    pass

  def energy(self,x,emax,emin):
    a=0.8; b=3; xsize=3
    f1=np.sum([-10*e**(-0.2*sqrt(x[z]**2+x[z+1]**2)) for z in xrange(xsize-1)])
    f2=np.sum([abs(x[z])**a+5*sin(x[z]**b)])
    ener=f1+f2
    eMron= (ener-emin)/(emax-emin)
    return eMron

  def neighbour(self,x,xmax,xmin):
    def __new(x,z):
        return xmin+(xmax-xmin)*rand(0,1) if rand(0,1)<0.33 else x[z]
    x_new=[__new(x,z) for z in xrange(3)]
    return x_new

  def do_a_randJump(self, e, en, t, k):
      p=math.e**(-(e-en)/(t*k))<rand(0,1)
#      print p
      return p

  def baselining(self):
    emax=-1;emin=1; a=0.8; b=3; xsize=3;
    for x in xrange(int(1e3)):
      x_tmp=[randi(-5,5) for z in xrange(3)]
      ener=np.sum([-10*e**(-0.2*sqrt(x_tmp[z]**2+x_tmp[z+1]**2)) for z in xrange
(xsize-1)])+\
      np.sum([abs(x_tmp[z])**a+5*sin(x_tmp[z]**b)])
      if ener>=emax:
        emax=ener
      elif ener<=emin:
        emin=ener
    return emax,emin

  f=open('log_sa_kursawe.txt','w')
  def say(self,x):
    self.f.write(str(x));
    sys.stdout.flush()

# Initial temperature

class main:

  k=1
  kmax=2000

  xmax=4;
  xmin=-4;
```



```
  sa=simulatedAnnealing()
  emax, emin = sa.baselining()
  print emax, emin
  # Initial state and energy
  sb=s=[rand(-4,4) for z in xrange(3)]
  eb=e=sa.energy(s,emax,emin)
  #print e

  print 'Initial Best', sb

  for k in xrange(1,kmax):
    #print k
    sn=sa.neighbour(s,xmax,xmin)
    en=sa.energy(sn,emax,emin)
    t=k/kmax

    if en<eb:
      eb, sb=en, sn; sa.say('!')

    if en<e:
      s, e = sn, en; sa.say('+')

    elif sa.do_a_randJump(en,e,k,1e-2): # The cooling factor needs to be reallyl
ow for some reason!!
      s, e=sn, en; sa.say('?')

    sa.say('.')
    if k%40==0: sa.say('\n')# sa.say(format(sb,'0.2f'))

  sa.say('\n'),sa.say('Best Value Found '), sa.say(sb)

# Print Energy and best value.
  print sb
if __name__=='main':
  main()
```

Sep 09, 14 5:52 **csc710sbse: hw2:Rahul Krishna** Page 1/1

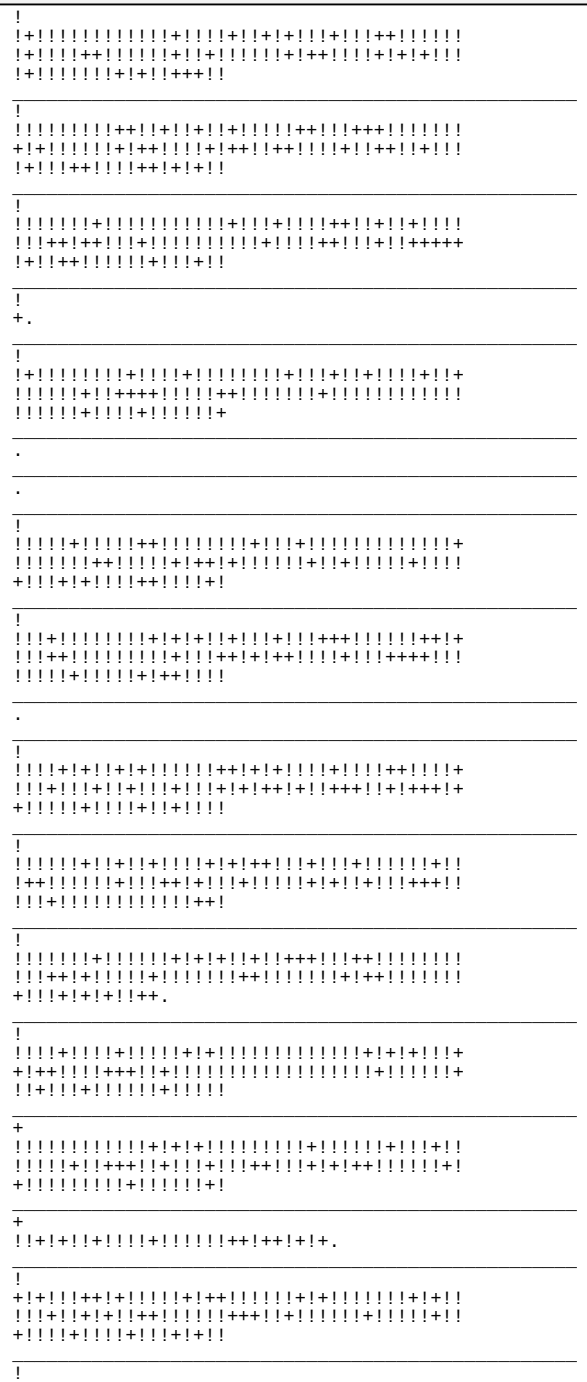


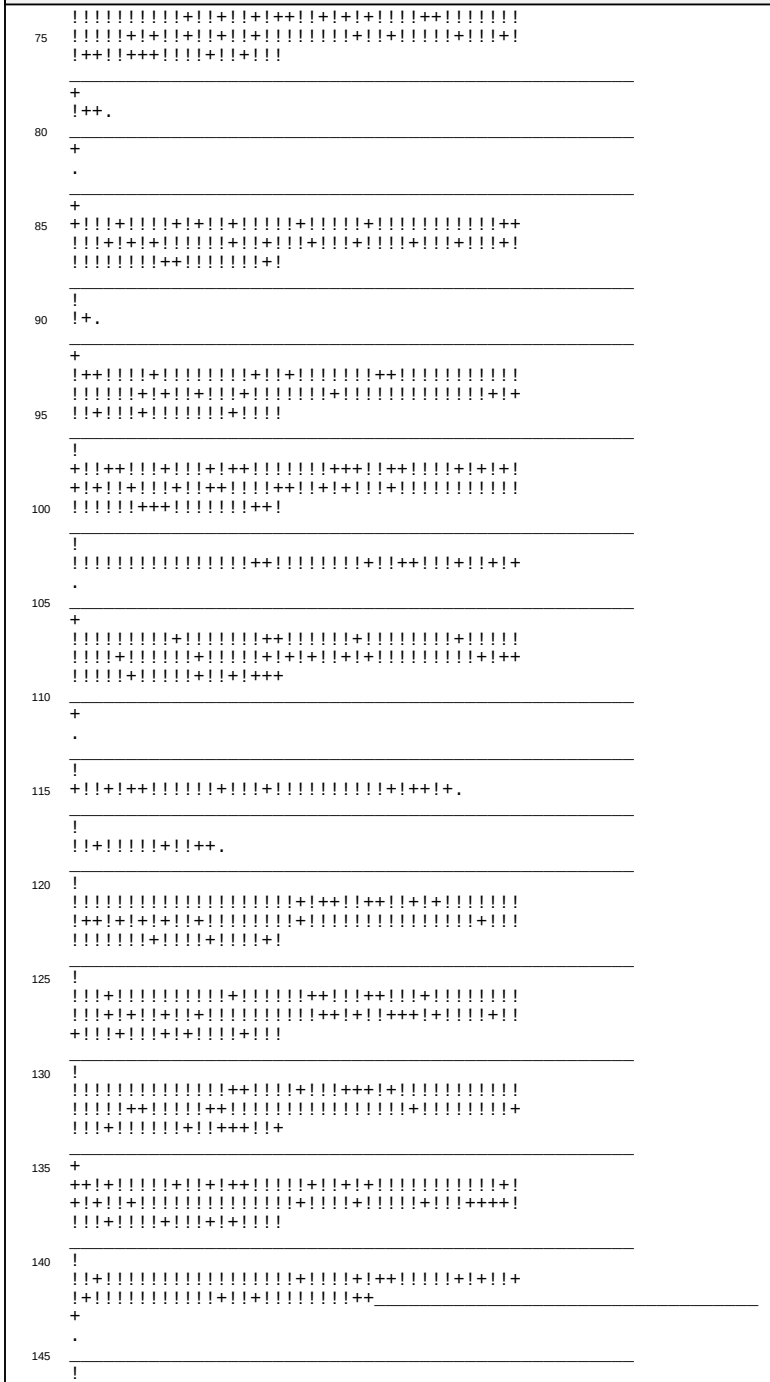
```
Best Value Found [0.03791650803205027, −0.0019257947925197527, 0.004228682162317
199]
```

```
# -*- coding: utf-8 -*-
"""
MaxWalkSat
Created on Mon Sep 08 02:15:42 2014

@author: Rahul

The Algorithm:

FOR i = 1 to max-tries DO
  solution = random assignment
  FOR j =1 to max-changes DO
    IF  score(solution) > threshold
       THEN  RETURN solution
    FI
    c = random part of solution
    IF    p < random()
    THEN  change a random setting in c
    ELSE  change setting in c that maximizes score(solution)
    FI
RETURN failure, best solution found

"""

## Standard imports

from __future__ import division
import sys,re,random,math,datetime,re,time
import numpy as np
import scipy as sp
sys.dont_write_bytecode = False

## Define some aliases.
rand=random.uniform
randi=random.randint
e=math.e
sin=math.sin
sqrt=math.sqrt
random.seed()

maxTries=100
maxChanges=100

## Create a class that defines all definitions in MaxWalkSat
class mWalkSat:

    def __init__(self):
        pass

## All we really need is a scoring function, which in our case would be the
## energy.


    def score(self,x,emax,emin):
        f1, f2=(1-e**np.sum([(x[z]-1/(np.sqrt(z+1))) for z in xrange(3)])),\
        (1-e**np.sum([(x[z]+1/(np.sqrt(z+1))) for z in xrange(0,3)]))
        ener=f1-f2
        eNorm= (ener-emin)/(emax-emin)
       #print e_norm
        return eNorm

    def baselining(self):
        emax=-1;emin=1;
        for x in xrange(int(1e3)):
          x_tmp=[randi(-4,4) for z in xrange(3)]
          ener=(1-e**np.sum([(x_tmp[z]-1/(np.sqrt(z+1))) for z in xrange(3)]))-\
          (1-e**np.sum([(x_tmp[z]+1/(np.sqrt(z+1))) for z in xrange(3)]))
          if ener≥emax:
            emax=ener
          elif ener≤emin:
            emin=ener
        return emax, emin
```



```
    def neighbour(self,x,xmax,xmin):
        return xmin+(xmax-xmin)*rand(0,1)

f=open('log_mwalksat.txt','w')
def say(self,x):
    self.f.write(str(x));
    sys.stdout.flush()

class main:
    # Create an instance of the maxWalkSAT class
    mwSAT=mWalkSat()
    score=mwSAT.score # Create an alias for the score function (Not required)
    neighbour=mwSAT.neighbour
    say=mwSAT.say
    # Do a baselining study on the score function
    emax, emin= mwSAT.baselining()
    # First define the limits of the independent the variables
    xmax, xmin=4, -4
    for i in xrange(maxTries):
        # Lets create a random assignment, I'll use list comprehesions here.
        x=xn=xb=[rand(-4,4) for z in xrange(3)]
        # Create a threshold for energy, let's say thresh=0.1% of emax (which is
 1) for starters
        thresh=1e-7
        for j in xrange(maxChanges):
            # Let's check if energy has gone below the threshold.
            # If so, look no further.
            if score(xn,emax,emin)<thresh:
                say('.')
                break
            else:
                randIndx=randi(0,2) # Choose a random part of solution x
                if rand(0,1)<0.25: # Probablity p=0.33
                    y=xn[randIndx]
                    xn[randIndx]=neighbour(y,xmax,xmin)
                    say('+')
                    #print 'Random change on', randIndx
                else:
                    # xTmp is a temporary variable
                    xTmp= xn; xTmp[randIndx]=rand(-4,4)
                    xBest=score(xTmp,emax,emin);
                    # Step from xmin to xmax, take 10 steps
                    Step=np.linspace(xmin,xmax,10)
                    say('!')
                    for i in xrange(np.size(Step)):
                        xNew=xn; xNew[randIndx]=Step[i];
                        if score(xNew,emax,emin)<xBest:
                            xBest=score(xNew,emax,emin)
                            xn=xNew

            if j%40≡0: say('\n')
        say('\n')
        for z in xrange(50): say('_')
        say('\n')
        if score(xn,emax,emin)<score(xb,emax,emin):
            xb=xn
    say('Best solution found: '), say(xb)
```

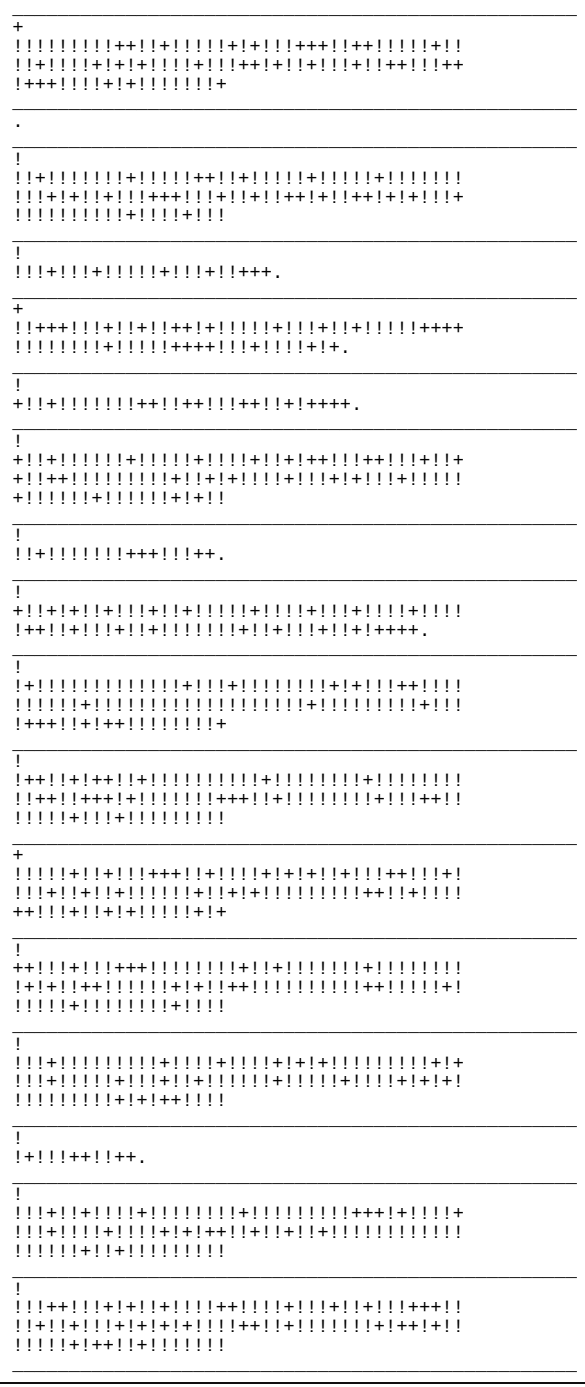


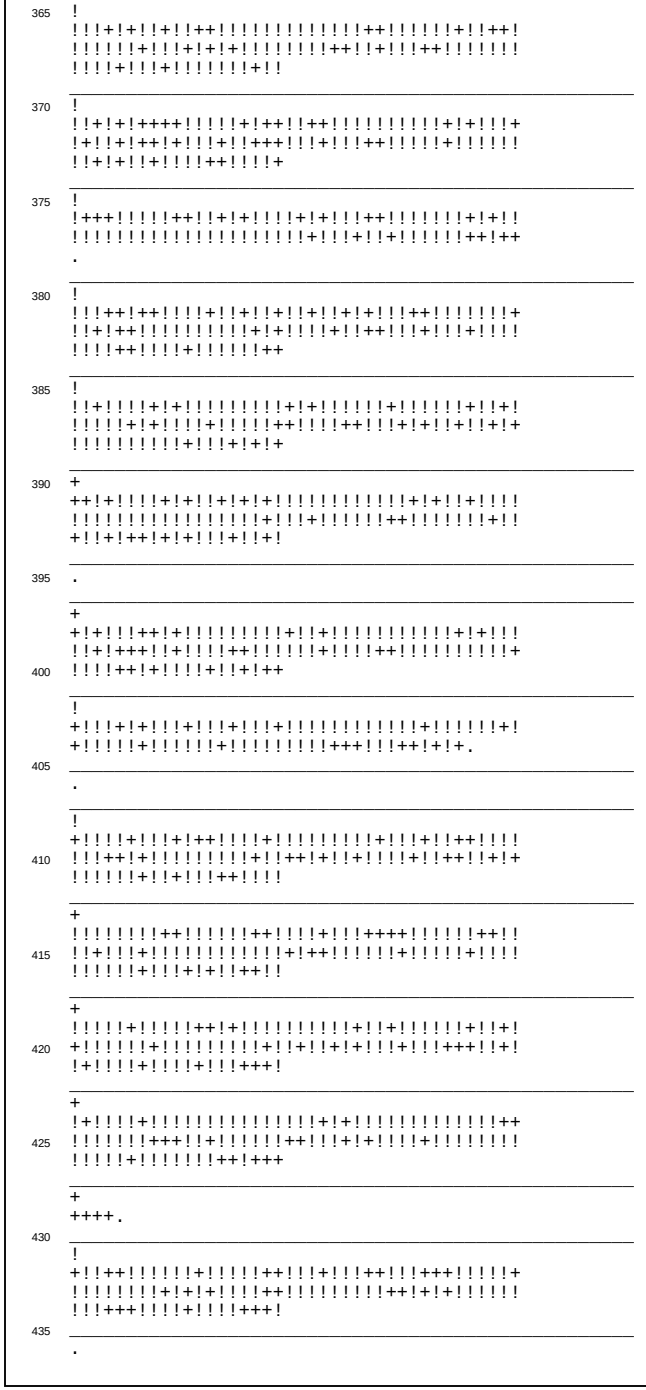


```
+!!+++!+++!+++!++!+!+++!++++++.
_____________________________________________________________
+
.
_____________________________________________________________
.
_____________________________________________________________
+
!+!++++++++!+++++++++++++!+!+.
_____________________________________________________________
+
+++++!+++++!++++!++!++++++++++++++++++++
++!++++++!++++++!+++++!+++++!!++!+!++++!
++.
_____________________________________________________________
+
+++.
_____________________________________________________________
+
++++!++++!+++++++.
_____________________________________________________________
.
_____________________________________________________________
+
!+!!!+++!+++!++!!!!++++++!++++++!++!++++
++++!!+!+++!++++++++++!!++.
_____________________________________________________________
+
++!+!!!++!!++!++++++!!++++++++++++++!+!++
++++++!++++!+++++!+!+!+!+++++!+++.
_____________________________________________________________
!
++++++!++!+++++.
_____________________________________________________________
!
!+++!++!!++!+++++++++!+++.
_____________________________________________________________
+
+++++!++++++!++++++!++++++!++.
_____________________________________________________________
+
++++++++++++++++++!++.
_____________________________________________________________
.
_____________________________________________________________
.
_____________________________________________________________
.
_____________________________________________________________
.
_____________________________________________________________
+
++!+!+++!!!+++++!!+!++!++++.
_____________________________________________________________
+
.
_____________________________________________________________
+
.
_____________________________________________________________
.
_____________________________________________________________
!
!!+++.
_____________________________________________________________
!
!+++++++!++!+++!!!+!+++++!++++!++!++++++
+.
_____________________________________________________________
!
+!+.
_____________________________________________________________
+
```



```
+++++++!+++++.
_____________________________________________________________
.
_____________________________________________________________
+
+++!++!+++++++!++++++++++++!++.
_____________________________________________________________
!
++++++.
_____________________________________________________________
!
!!+!++.
_____________________________________________________________
!
++++!!+!!+!+++++!+!!+!++++++++!++++++!+.
_____________________________________________________________
!
+++!++.
_____________________________________________________________
!
+++.
_____________________________________________________________
+
+++++!++++++!+++++!+.
_____________________________________________________________
+
++++++!+!++++!++!+++!+!!++++++++!+++!++++
++!+++++++!+++!++!++.
_____________________________________________________________
.
_____________________________________________________________
!
++.
_____________________________________________________________
!
+++!+!!++++++++++++++!+!!++++++++!++!!++!
++++!++!++++++++.
_____________________________________________________________
!
!++!++!!!+!++!++++!++++++++!++++!++!!++++
+++++++!++!+++!+!+++++!+++!!+!!!!++++!++
++++++.
_____________________________________________________________
+
+++++++++.
_____________________________________________________________
+
++!++!+!!+++++++.
_____________________________________________________________
+
++++.
_____________________________________________________________
+
!++!!+!!+!!!+++++!!+!++!!+!+++.
_____________________________________________________________
+
+.
_____________________________________________________________
.
_____________________________________________________________
Best solution found: [-0.8597488796167694, -2.6629032031236797, -2.0252384321946
755]_____________________________________________________
!
!!!!!+!!+!+!!!!!!+!!++!!!!!!!!!++!!!!!!+
+++!!!+!!+!+!!!!!!!!!+!!+!!+!!!++!!+!!++
!!!!!!!!!+!!+!!!+!!
_____________________________________________________________
!
+!++!+++!!++!++!!+!!+!!+!+!!!+!!!!!!!!!!
!!!!+!++!!!!!!!!++!!+!!!+!!!!!++!!!!!+!+
!!!+!!!++!+!+!!!!!!
_____________________________________________________________
.
```





```
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  9 07:24:05 2014

@author: rahul
"""

Effect of changing the probablity of random change
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
The output file has 3 indicators:
  - '+' indicates a random change on a variable.
  - '!' indicates a local search on a variable.
  - Finally, a '.' indicatates that a set threshold has been reached.

The probability of a local search is (1-p) and the probability of a random
jump is p. Since, we are not sure if the system has to randomly change state
of remain in its current state, we can logically make use of the principle of
maximum indifference (or maximum entropy). In accordance to this, we can set
p=(1-p)=0.5. This maximized the expanse of search.

Here's the sample of search at p=0.5:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
_____________________________________________________________________
!
+++!+!+!!!!!++!!++!++!!!!++!+!!+!++!+!!+
!+!+!!!!!++!+!!++!!+!+!!+!!!!+!++++!+++!
!!!!+!!!++!++!!+!!+
_____________________________________________________________________

At p=0.75 (P[local search]=0.25)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
_____________________________________________________________________
!
+++!!!++++++++++!!++++++!+++!+++++++!!+
++!++!++!++!+++!+!++++++!+++!+!+!+!+!!++
!!!+++++.
_____________________________________________________________________

At p=0.25 (P[local search]=0.75)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
_____________________________________________________________________
!
!!!!!!!+!!!!!!!!!!!+!!!+!!!!++!!+!!+!!!!
!!!++!++!!!+!!!!!!!!!!+!!!!++!!!+!!+++++
!+!!++!!!!!!+!!!+!!
_____________________________________________________________________

Notice how at p=0.5 the '+' and '!' are almost equally distributed. At, p=0.75
the '+' is far more than '!', suggesting that local search is performed less
often. And at p=0.25 the number of local searches is very high.
```