Kuncheng Feng
CSC 416

RBG String GA Recreation Assignment

Task 1

Source Code

```
(setf *limit* 25)

(defmethod rbg()
      (nth (random 3) '(r b g))
)

(defmethod rbg-string()
      (generate *limit*)
)

(defmethod generate(number)
      (if (equal number 0)
            (list)
            (cons (rbg) (generate (- number 1)))
      )
)
```

Demo

```
[1]> (load "rbg.l")
;; Loading file rbg.l ...
;; Loaded file rbg.l
T
[2]> *limit*
25
[3]> (rbg)
B
[4]> (rbg)
G
[5]> (list (rbg) (rbg) (rbg) (rbg) (rbg))
(R G G R G)
[6]> (list (rbg) (rbg) (rbg) (rbg) (rbg))
(B G B B G)
[7]> ( rbg-string )
(R G R G R B R R G R G B R R R B G G B B G G B G R)
[8]> ( rbg-string )
(G B B G G R G B R B B G B R R B G R G R B R G B G)
[9]> ( rbg-string )
```

```
(G B G R G R B G B G B G B G B R G G B G R R G B B)
[10]> ( rbg-string )
(R R B R B R G B B G B G B G B R R R B G G G B G B)
```

Task 2: Mutation

Source Code

```
( defmethod mutation ( ( rbg-str list ) &aux position symbol )
      ( setf position ( random ( length rbg-str ) ) )
      ( setf symbol ( others '( r b g ) ( nth position rbg-str ) ) )
      ( change rbg-str ( pick symbol ) position )
)

(defmethod others(group unwanted)
      (remove unwanted group)
)

(defmethod pick(group)
      (nth (random (length group)) group)
)

(defmethod change(oldList newElement spot &aux newList)
      (setf newList (copy-list oldList))
      (setf (nth spot newList) newElement)
      newList
)
```

Demo

```
[1]> (load "rbg.l")
;; Loading file rbg.l ...
;; Loaded file rbg.l
T
[2]> (setf colors '(r g b r))
(R G B R)
[3]> (mutation colors)
(R G G R)
[4]> colors
(R G B R)
[5]> (mutation colors)
(B G B R)
[6]> colors
(R G B R)
[7]> (mutation colors)
(R G B B)
```

```
[8]> (setf s '(r b g g b r))
(R B G G B R)
[9]> (setf s (mutation s))
(R B G G B G)
[10]> (setf s (mutation s))
(R R G G B G)
[11]> (setf s (mutation s))
(R R G B B G)
[12]> (setf s (mutation s))
(B R G B B G)
[13]> (setf x (rbg-string))
(B B R G G B G B B B B G R R G R G G B R B R B B B)
[14]> (setf x (mutation x))
(B B R B G B G B B B B G R R G R G G B R B R B B B)
[15]> (setf x (mutation x))
(B B R B G B G B B B B G R R G R G G R R B R B B B)
[16]>
```

Task 3: Crossover

Source Code

```
( defmethod crossover ( ( m list ) ( f list ) &aux pos)
      ( setf pos ( + 1 ( random ( length m ) ) ) )
      ( append ( first-n m pos ) ( rest-n f pos ) )
)

(defmethod first-n((l list) pos &aux newPos)
      (setf newPos (- pos 1))
      (cond
            ((>= newPos (length l))
                  (first-n l newPos)
            )
            ((>= newPos 0)
                  (append (first-n l newPos) (list (nth newPos l)))
            )
            (t
                  (list)
            )
      )
)

(defmethod rest-n((l list) pos)
      (cond
            ((>= pos (length l))
                  (list)
            )
            (t
                  (append (list (nth pos l)) (rest-n l (+ pos 1)))
```

```
            )
        )
)
```

Demo

```
[1]> (load "rbg.l")
;; Loading file rbg.l ...
;; Loaded file rbg.l
T
[2]> (setf m '(a b c d e f g))
(A B C D E F G)
[3]> (setf f '(t u v w x y z))
(T U V W X Y Z)
[4]> (crossover m f)
(A B C D E Y Z)
[5]> (crossover m f)
(A B V W X Y Z)
[6]> (crossover m f)
(A B C D X Y Z)
[7]> (crossover m f)
(A U V W X Y Z)
[8]> m
(A B C D E F G)
[9]> f
(T U V W X Y Z)
[10]> (setf m (rbg-string))
(B B G B B B R G B B B G R B B R G B R G B G R B B)
[11]> (setf f (rbg-string))
(R G B B G G R B B B R G G B G B B B B G R R G R G)
[12]> (crossover m f)
(B B G B B B R G B B B G G B G B B B B G R R G R G)
[13]> (crossover m f)
(B G B B G G R B B B R G G B G B B B B G R R G R G)
[14]> (crossover m f)
(B B G B B B R G B B B G R B B R G B R G R R G R G)
[15]> m
(B B G B B B R G B B B G R B B R G B R G B G R B B)
[16]> f
(R G B B G G R B B B R G G B G B B B B G R R G R G)
[17]>
```

Task 4: Demo for Mutation and Crossover

Source Code

```
( defmethod mutation-demo (&aux s m)
      ( setf s ( rbg-string ) )
      ( dotimes ( i 10 )
            ( format t "s = ~A~%" s )
            ( setf m ( mutation s ) )
            ( format t "m = ~A~%~%" m )
      )
)

( defmethod crossover-demo (&aux m f x)
      ( setf m ( rbg-string ) )
      ( setf f ( rbg-string ) )
      ( dotimes ( i 10 )
            ( format t "m = ~A~%" m )
            ( setf x ( crossover m f ) )
            ( format t "x = ~A~%" x )
            ( format t "f = ~A~%~%" f )
      )
)
```

Demo

```
[]> (mutation-demo)
s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R B R B G G B B G G B G R G B B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R R R B G G B B G G B G R G B R G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R R R B G G B B G R B G R G B B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R R R B G G B B G G B G R G G B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (B B R R G R G B R R R B G G B B G G B G R G B B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R R R R G G B B G G B G R G B B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R G R B G G B B G G B G R G B B G)
```

```
s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R R R B B G B B G G B G R G B B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R R R R G R G B R R R B G G B B G G B G R G B B G)

s = (R B R R G R G B R R R B G G B B G G B G R G B B G)
m = (R B R R G R G B R R R B G R B B G G B G R G B B G)

NIL
[]> (crossover-demo)
m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R B G B G B G B G B R G R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B B R B R G B B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G R R B R B R G B B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R B G B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R B B B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R B G B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R G G B B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R B G B G B G B G B R G G B G R R G G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

m = (G G B G R G R B G B G B G B G B R G G B G R R G B)
x = (G G B G R G R B G B G B G B G B G B R R R B G G G B G)
f = (B R R B R B R G B B G B G B G B R R R B G G G B G)

NIL
[]>
```

Task 5: The Fitness Metric

Source Code

```
(defmethod fitness-r ((rbg-string list))
      (count 'r rbg-string)
)

(defmethod fitness-b ((rbg-string list))
      (count 'b rbg-string)
)

(defmethod fitness-g ((rbg-string list))
      (count 'g rbg-string)
)

( defmethod fitness-demo (&aux x fitness)
      ( setf x (rbg-string) )
      ( format t "x = ~A~%" x )
      ( format t "Directly applying the fitness metrics ...~%" )
      ( format t "fitness-r = ~A~%" ( fitness-r x ) )
      ( format t "fitness-b = ~A~%" ( fitness-b x ) )
      ( format t "fitness-g = ~A~%" ( fitness-g x ) )
      ( format t "Indirectly applying the fitness metrics ...~%" )
      ( setf fitness #'fitness-r )
      ( format t "fitness-r = ~A~%" ( funcall fitness x ) )
      ( setf fitness #'fitness-g )
      ( format t "fitness-g = ~A~%" ( funcall fitness x ) )
      ( setf fitness #'fitness-b )
      ( format t "fitness-b = ~A~%" ( funcall fitness x ) )
)
```

Demo

```
[1]> (load "rbg.l")
;; Loading file rbg.l ...
;; Loaded file rbg.l
T
[2]> (setf x (rbg-string))
(R B B R G R B R G G R G R G G B B B B R G R G B R)
[3]> x
(R B B R G R B R G G R G R G G B B B B R G R G B R)
[4]> (fitness-r x)
9
[5]> (fitness-b x)
8
[6]> (fitness-g x)
8
[7]> (setf fitness #'fitness-r)
#<STANDARD-GENERIC-FUNCTION FITNESS-R>
[8]> (funcall fitness x)
9
```

```
[9]> (setf fitness #'fitness-b)
#<STANDARD-GENERIC-FUNCTION FITNESS-B>
[10]> (funcall fitness x)
8
[11]> (setf fitness #'fitness-g)
#<STANDARD-GENERIC-FUNCTION FITNESS-G>
[12]> (funcall fitness x)
8

[13]> (fitness-demo)
x = (G G B G B B B B G R R G R G G B R B R B B B G R G)
Directly applying the fitness metrics ...
fitness-r = 6
fitness-b = 10
fitness-g = 9
Indirectly applying the fitness metrics ...
fitness-r = 6
fitness-g = 9
fitness-b = 10
NIL
```

Task 6: The Individual Class

Source Code

```
( defclass individual ()
      (
            ( rbg-string :accessor individual-rbg-string :initarg :rbg-string
)
            ( fitness :accessor individual-fitness :initarg :fitness )
            ( number :accessor individual-number :initarg :number )
      )
)

( defmethod random-individual (&aux rbg)
      ( setf rbg ( rbg-string ) )
      ( make-instance 'individual
            :rbg-string rbg
            :fitness ( funcall *fitness* rbg )
            :number 0
      )
)

( defmethod new-individual ( ( nr number ) ( notes list ) )
      ( make-instance 'individual
            :rbg-string notes
            :fitness ( funcall *fitness* notes )
            :number nr
```

```
      )
)

( defmethod display ( ( i individual ) )
      ( display-nnl i ) ( terpri )
)

( defmethod display-nnl ( ( i individual ) )
      ( prin1 ( individual-number i ) )
      ( princ ( filler ( individual-number i ) ) )
      ( prin1 ( individual-rbg-string i ) )
      ( princ " " )
      ( prin1 ( individual-fitness i ) )
      ( princ ( filler ( individual-fitness i ) ) )
)

( defmethod filler ( ( n number ) )
      ( cond
            ( ( < n 10 ) "     " )
            ( ( < n 100 ) "    " )
            ( ( < n 1000 ) "   " )
            ( ( < n 10000 ) "  " )
            ( ( < n 100000 ) " " )
      )
)

( defmethod fitness-b ( ( i individual ) )
      ( fitness-b ( individual-rbg-string i ) )
)

( defmethod fitness-r ( ( i individual ) )
      ( fitness-r ( individual-rbg-string i ) )
)

( defmethod fitness-g ( ( i individual ) )
      ( fitness-g ( individual-rbg-string i ) )
)

( defmethod individual-demo (&aux i0 i1 i2 i3 one two three)
      ( setf *fitness* #'fitness-r )
      ( setf i0 ( random-individual ) )
      ( display i0 )
      ( setf one ( rbg-string ) )
      ( setf i1 ( new-individual 1 one ) )
      ( display i1 )
      ( setf two ( rbg-string ) )
      ( setf i2 ( new-individual 2 two ) )
      ( display i2 )
      ( setf three ( rbg-string ) )
      ( setf i3 ( new-individual 3 three ) )
      ( display i3 )
      ( format t "Fitness of i0 = ~A~%" ( funcall *fitness* i0 ) )
      ( format t "Fitness of i1 = ~A~%" ( funcall *fitness* i1 ) )
      ( format t "Fitness of i2 = ~A~%" ( funcall *fitness* i2 ) )
      ( format t "Fitness of i3 = ~A~%" ( funcall *fitness* i3 ) )
      nil
```

```
)
```

Demo

```
[1]> (load "rbg.l")
;; Loading file rbg.l ...
;; Loaded file rbg.l
T
[2]> (setf rbg (rbg-string))
(B B B B R G R G B R R B G R R R G R B R B G G B B)
[3]> rbg
(B B B B R G R G B R R B G R R R G R B R B G G B B)
[4]> (setf *fitness* #'fitness-b)
#<STANDARD-GENERIC-FUNCTION FITNESS-B>
[5]> (setf rbg-i (new-individual 1 rbg))
#<INDIVIDUAL #x1AAD1F95>
[6]> (individual-number rbg-i)
1
[7]> (individual-rbg-string rbg-i)
(B B B B R G R G B R R B G R R R G R B R B G G B B)
[8]> (display rbg-i)
1     (B B B B R G R G B R R B G R R R G R B R B G G B B) 10
NIL
[9]> (funcall *fitness* rbg)
10
[10]> (setf r (random-individual))
#<INDIVIDUAL #x1AAB3279>
[11]> (display r)
0     (R G R R B G B G R G G R G B G B B G R G R G R B R) 6
NIL
[12]> (setf r (random-individual))
#<INDIVIDUAL #x1AAC3D55>
[13]> (display r)
0     (R G R G B R R R B G G B B G G B G R G B B G G R G) 7
NIL
[14]> (individual-demo)
0     (B R B B G B R R B G R G R B R G B G G B G R G R B) 8
1     (G B G B G B G B R G G B G R R G B B R R B R B R G) 7
2     (B B G B G B G B R R R B G G G B G B G B B B B B G) 3
3     (R G B B R G G R G B G R G G G B B B G R R B R G R) 8
Fitness of i0 = 8
Fitness of i1 = 7
Fitness of i2 = 3
Fitness of i3 = 8
NIL
```

Task 7: The Population Class

Source Code

```
( defconstant *population-size* 100 )
( defconstant *selection-size* 8 )

( setf *fitness* #'fitness-b )

( defclass population ()
      (
            ( individuals :accessor population-individuals :initarg
:individuals )
            ( generation :accessor population-generation :initform 0 )
      )
)

( defmethod size ( ( p population ) )
      ( length ( population-individuals p ) )
)

( defmethod display ( ( p population ) )
      ( terpri ) ( terpri )
      ( princ "Generation " )
      ( prin1 ( population-generation p ) )
      ( princ " population ..." )
      ( terpri ) ( terpri )
      ( dolist ( i ( population-individuals p ) )
            ( display i )
      )
      ( terpri )
)

( defmethod initial-population ( &aux individuals )
      ( setf individuals () )
      ( dotimes ( i *population-size* )
            ( push ( new-individual ( + i 1 ) ( rbg-string ) ) individuals )
      )
      ( make-instance 'population :individuals ( reverse individuals ) )
)

( defmethod average ( ( p population ) &aux ( sum 0 ) individuals)
      (setf individuals (population-individuals p))
      (dotimes (i *population-size*)
            (setf sum (+ sum (funcall *fitness* (nth i individuals))))
      )
      (/ sum (float *population-size*))
)

( setf *select-demo* nil )

( defmethod select-individual ( ( p population ) &aux i candidates rn )
      ( setf candidates ( select-individuals p ) )
      ( setf mfi ( most-fit-individual candidates ) )
```

```
      ( if *select-demo* ( select-demo-helper candidates mfi) )
      mfi
)

( defmethod select-individuals ( ( p population ) &aux individuals candidates
rn )
      ( setf individuals ( population-individuals p ) )
      ( setf candidates () )
      ( dotimes ( i *selection-size* )
            ( setf rn ( random *population-size* ) )
            ( push ( nth rn individuals ) candidates )
      )
      candidates
)

; Sometimes when the world suddenly changes color, the most-fit individual
could have a fitness of 0
( defmethod most-fit-individual ( ( l list ) &aux max-value max-individual
individual fitness)
      (setf max-value 0)
      (setf max-individual nil)
      (dotimes (i (length l))
            (setf individual (nth i l))
            (setf fitness (funcall *fitness* individual))
            (cond
                  ((>= fitness max-value)
                        (setf max-value fitness)
                        (setf max-individual individual)
                  )
            )
      )
      max-individual
)

( defmethod select-demo-helper ( ( l list ) ( i individual ) )
      ( princ "the sample of individuals ..." ) ( terpri )
      ( mapcar #'display l )
      ( terpri )
      ( princ "the most fit of the sample ... " ) ( terpri )
      ( display i )
      ( terpri )
      nil
)

( defmethod population-demo (&aux p)
      ( setf p ( initial-population ) )
      ( display p )
      ( format t "Average fitness = ~A~%~%" ( average p ) )
      ( setf *select-demo* t )
      ( format t "Sampling ...~%~%" )
      ( select-individual p ) ( terpri)
      ( format t "Sampling ...~%~%" )
      ( select-individual p ) ( terpri)
      ( format t "Sampling ...~%~%" )
      ( select-individual p ) ( terpri)
)
```

Demo

```
[13]> (population-demo)


Generation 0 population ...

1     (R B R B B B G B G G G B G G G G G R B R R B R G R) 8
2     (R B B B B B G G B G R G R R G G R B R G R B G B B) 10
3     (B B B G G G R R B B R G B B B B B G R R G R G R G) 10
4     (G R B R G R B B G G R G G B B G G B R R B G B G B) 9
5     (B R B R R R R R R B G B B B G B B G B G G G G B R) 10
6     (B B G R B R B B G B B B B R R R R G B R R G R R R) 10
7     (R R G B R R B R G B R G R G G R B B R G G G B B R) 8
8     (B B B B R B R G G G G R G B R G G R R G B G R G B) 8
9     (R G B G G G R R B B B B G B R R G R G R B G R G B) 8
10    (B R R G B R R R R R R R R G B G R B R G G B R B B) 7
11    (G G G G G G R B R G R R G B R G R R R G B R G B G) 4
12    (G G G B G B G G R B R G R G B R R B G R G G B G R) 6
13    (G R R B R B G R B G R G G G R G B R G G G B R R G) 5
14    (R R G G G G B G B G B R G R R G R G R B G B G G G) 5
15    (R B B R R G G R R G R R G G B B R G B B R B B B G) 9
16    (B B B R B B R G B B G G B G G G R B R G G G R G R R B) 9
17    (G G R B G G G G R R G R B G G G G G R B R R B G R B B) 6
18    (R B B B G B G R G B B G R R B R R R G G B G G G R B) 9
19    (R R B G B R B G G G B R G G G R R B R B R R R G G R B) 7
20    (B G G R G B B B G R R B B R B G R G G G B B R B G B G) 10
21    (R B G G G G B G B R R G B G G G G G R B B R G B R G B) 8
22    (B B B B G G G B B R R B G G R B G B G G G R R G R G B) 10
23    (B G R G B B R R R B B G R B R G B G G B B R G G G B R) 10
24    (B B G B R B B B R G R G B R B B R B G R B B R B R B) 13
25    (R R G R R R R B G B R R R R G R R B B R B R G G R B) 6
26    (G R B R G R G R R R R G B B R G G B R R G R B R R) 5
27    (B R R R B B B G G G R G G R R R B R G G G R R B B B) 8
28    (R R G G B G R G G G R B B B B R R B B R B B R G B) 10
29    (G B G G G R G G G B G R G G R R G R B R B B G R B) 6
30    (R B B G B R B B B G G B R R G R B R G B R R G G B R) 9
31    (R G B G B R R G G B R G G G B G B R R B B R R G G) 7
32    (B R G B G R B G G G G B G B R B R G G B R B G R B) 9
33    (G R R B B R R G R R G R R B G B B B R B B R R B B) 10
34    (G G R R B B G G B R R G B R G R R G B B R G R R R) 6
35    (B R B G R B B B R G G B R G B B G B G R B B G G G) 11
36    (R G B B R B R G B B B G R G R B R B R B B R B R R) 11
37    (R R B B G R G R B G G G G G R B R B R G B G R G B) 7
38    (B R G R R R G B B G R B G G B R B G G G G R B B R) 8
39    (G G G R B G B B R B B R B R G G G B G R G G G B B) 9
40    (G R B R G G B B B B B G R B R G G G R B G G B B G) 10
41    (R R B G G R G B R G G B R G B B G G G G R B R B R) 7
42    (R R G R G B B B G R R R G R R G B B R B G G R R G) 6
43    (R G R G R B G G B G R B R G R B G G B R R G B G B) 7
44    (R G R B B B G B G G G R G G B B G B B G B G R B R) 10
45    (R G R G G G R G B G R R R B B G R B R G G B G B R) 6
46    (B B R B G B R B B G G R B G B G B G B R G B R G B) 12
```

```
47    (R B R G R B R R R R G G R G B G B R B R B B B B G) 9
48    (G B R B B B B B R G G G R R G B B G B B B G G G G) 11
49    (R G R R R R G G R B R R R B B R R G G R G G R G R) 3
50    (R R B B G G G G R R R R R B G G G B R B R R R R R) 5
51    (G G B G B B R G G B G R B B G G R R G B G R G G B) 8
52    (B R R G R R B G G G G R R B R G B B B B G R B B B) 10
53    (G R R B B G G R R G R G R B R G B B R B B B R B R) 9
54    (G R R R G B G R B G R R B R G G B B G B G R R R B) 7
55    (R B B G G B R R G R G B G B G R B B R G R G G G R) 7
56    (G R G G G R B B R G G G B G B R B R G G B R B G R) 7
57    (G G B G G R G B B G G B B G G B B G B G R G R B R) 9
58    (R B B B B B G B G R G G G B G G G R R B B R R B R) 10
59    (G G B R G B B G G G R R B B R G B G B G G B G B G) 9
60    (G G G G G R R R R B B B R B B G B R R G G B R B G) 8
61    (B R R G R R B B B R R G G R R B B B R B R G R R G) 8
62    (R R B R R G R G G B B G B G R G R G G G B R B G G) 6
63    (R B B R G G B B G B R R R R B B G B B B R G R G B) 11
64    (B B G R R R R R R G B B B G B G R B R R G R G R R) 7
65    (G R B R B B R R B G R R G B B B B R R B B R R G G) 10
66    (B R G G R R B B R B G R R R R B B G R R G R B G G) 7
67    (G R G R B G G B R G R G G R G B R R R R B B B R G) 6
68    (G G B R G G B B R R G R R B R R B R G G R B G R G) 6
69    (R G R R R B G B R B G G R R B R B R R G G R R B R) 6
70    (R G G R R R R G B R G G G G G B B R R R B B R R B) 6
71    (R R B R R G G R B G G B B G B G G G R G B R R B G B) 8
72    (R R B B B R G R G R G G R R G R B B G B G R R B B) 8
73    (B R G G B R R B R B G B G G B G R G R R G B B B G) 9
74    (R B B G G B G G G G R R B R B B G G B R R B G R R) 8
75    (R G B R B B G R R G B B B G R G R B R R G G G R B) 8
76    (G B R B R B G G B G G B R R B B B R R R B G R G B) 10
77    (B R G R R G B G R B B R R R R R G B R G B B G R B) 8
78    (B R R R B R R B B G R G G R R R B B R R B B R R B R) 9
79    (G G R B B B G R R B R G B R R B G R R B R G B G R) 8
80    (R G R R R G B R G G G G B G G R B G R B R G B B B) 7
81    (G R G B B B R B G B R B B R R G G G G G G G R G R R) 7
82    (R R R G G B G R B R G G B R B R R R G B B B R B B) 9
83    (G B B R B R G B B R R G B G G B G R B B B B B R R) 12
84    (R G R R R B G B B R R B G G G G G R B G B R B B G) 8
85    (G G G R G B R G B B G G R B G B B R R G R B G G G) 7
86    (B B R B R G G R G G G B G R G R B G B B B G G R B) 9
87    (B R R R G G R B R G G R R R B B R G R R B B G G B) 7
88    (B B B B G B B G B R R B R B B B B G B B R R R B G) 15
89    (G B G R R B G G R B R B G R B B R R B G R R B G R) 8
90    (G R G B R B B B G B B R R G G G R B G G B G G R G) 8
91    (R R R B G B G G G R R G R R B B G R G R G B G G B) 6
92    (R R B B R B B G R G R B G R B B B R R R R B G R R) 9
93    (G G R B B B R B R R R R G R R G G G B B R B B G B) 9
94    (R R R G G R B B R R B B R R G G B G B R G R B R B) 8
95    (B R R B B G R B B B R R R B G B B B R B R G B G G) 12
96    (B R G G R B G B G G B G R G B R R B R B B R R B R) 9
97    (B G G G G G R R G B B G B R R R G R R B G G B R R) 6
98    (B G G R G R B B B B G R G R R G B R G G G G G G G) 6
99    (G G R R G R G R R G G R B B G R B R G R G B B R B) 6
100   (G G B B R B R B G B R B B G G R G R B B B B R R G) 11

Average fitness = 8.19
```

```
Sampling ...

the sample of individuals ...
30    (R B B G B R B B G G B R R G R B R G B R R G G B R) 9
52    (B R R G R R B G G G G R R B R G B B B B G R B B B) 10
71    (R R B R R G G R B G G B B G B G G R G B R R B G B) 8
84    (R G R R R B G B B R R B G G G G G R B G B R B B G) 8
99    (G G R R G R G R R G G R B B G R B R G R G B B R B) 6
50    (R R B B G G G G R R R R R B G G G B R B R R R R R) 5
52    (B R R G R R B G G G G R R B R G B B B B G R B B B) 10
94    (R R R G G R B B R R B B R R G G B G B R G R B R B) 8

the most fit of the sample ...
52    (B R R G R R B G G G G R R B R G B B B B G R B B B) 10


Sampling ...

the sample of individuals ...
2     (R B B B B B G G B G R G R R G G R B R G R B G B B) 10
23    (B G R G B B R R B B G R B R G B G B B R G G G B R) 10
55    (R B B G G B R R G R G B G B G R B B R G R G G G R) 7
41    (R R B G G R G B R G G B R G B B G G G G R B R B R) 7
82    (R R R G G B G R B R G G B R B R R R G B B B R B B) 9
21    (R B G G G B G B R R G B G G G G R B B R G B R G B) 8
16    (B B B R B R G B B G G B G G R B R G G G R G R R B) 9
45    (R G R G G G R G B G R R R B B G R B R G G B G B R) 6

the most fit of the sample ...
2     (R B B B B B G G B G R G R R G G R B R G R B G B B) 10


Sampling ...

the sample of individuals ...
90    (G R G B R B B B G B B R R G G G R B G G B G G R G) 8
11    (G G G G G G R B R G R R G B R G R R R G B R G B G) 4
82    (R R R G G B G R B R G G B R B R R R G B B B R B B) 9
84    (R G R R R B G B B R R B G G G G G R B G B R B B G) 8
16    (B B B R B R G B B G G B G G R B R G G G R G R R B) 9
60    (G G G G G R R R R B B B R B B G B R R G G B R B G) 8
94    (R R R G G R B B R R B B R R G G B G B R G R B R B) 8
37    (R R B B G R G R B G G G G G G R B R B R G B G R G B) 7

the most fit of the sample ...
82    (R R R G G B G R B R G G B R B R R R G B B B R B B) 9


NIL
```

Task 8: Incorporating Mutation

Source Code:

```
( defmethod mutate ( ( i individual ) &aux mutation )
      ( setf mutation ( mutation ( individual-rbg-string i ) ) )
      ( make-instance 'individual
            :number ( individual-number i )
            :rbg-string mutation
            :fitness ( funcall *fitness* mutation )
      )
)

( defconstant *pc-m* 50 )

( defmethod maybe-mutate ( ( i individual ) )
      ( if ( <= ( + 1 ( random 100 ) ) *pc-m* )
            ( mutate i )
            i
      )
)

( defmethod mutate-demo ()
      ( setf i ( random-individual ) )
      ( display i )
      ( dotimes ( x 20 )
            ( setf i ( mutate i ) )
            ( display i )
      )
)

( defmethod maybe-mutate-demo ()
      ( setf i ( random-individual ) )
      ( display i )
      ( dotimes ( x 20 )
            ( setf n ( maybe-mutate i ) )
            ( display-nnl n )
            ( if ( not ( equal n i ) ) ( princ " *" ) )
            ( terpri )
            ( setf i n )
      )
)
```

Demo:

```
[]> (mutate-demo)
0     (G R G B G B B G R G R G R B R R G R G B R R R B G) 6
0     (G R G B G B B B R G R G R B R R G R G B R R R B G) 7
0     (G R G B G B B B R G R G R B R R G R G B R R G B G) 7
0     (G R G B R B B B R G R G R B R R G R G B R R G B G) 7
0     (G R G B R B B B R G R G R B B R G R G B R R G B G) 8
0     (G R G B R B R B R G R G R B B R G R G B R R G B G) 7
```

```
0     (G R G B R B B B R G R G R B B R G R G B R R G B G) 8
0     (G R G B R B B B R R R G R B B R G R G B R R G B G) 8
0     (G R G B R B B B R R R G R B B R G R G B R R G R G) 7
0     (G R G B R B B B R R R G R B B R G B G B R R G R G) 8
0     (G R G B R B B B R R R G R B B R G B G B R R B R G) 9
0     (R R G B R B B B R R R G R B B R G B G B R R B R G) 9
0     (R R G B R B B B R R R R R B B R G B G B R R B R G) 9
0     (R R G B R B B B R G R R R B B R G B G B R R B R G) 9
0     (R R G B R B B B R G R R G B B R G B G B R R B R G) 9
0     (R B G B R B B B R G R R G B B R G B G B R R B R G) 10
0     (R B G B R B B B R G R R G R B R G B G B R R B R G) 9
0     (R B G B R B B B R G R R G R B R B B G B R R B R G) 10
0     (R B G B R B B B R G R R G R B R B G G B R R B R G) 9
0     (R B R B R B B B R G R R G R B R B G G B R R B R G) 9
0     (R B R B R B B B R G R R G R B R B G G B R R B R G) 8
NIL
[]> (maybe-mutate-demo)
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G B G B G B G B R G G B G R R G B B R R B R B R G) 9
0     (G R G B G B G B R G G B G R R G B B R R B R B R G) 8       *
0     (G R G B G B G B R G G B G R G G B B R R B R B R G) 8       *
0     (G R G B G B G B R G G B G R G G B B R R B G B R G) 8       *
0     (G R G B G B G B R G G B G R G G B G R R B G B R G) 7       *
0     (G R G B G B G B R R G B G R G G B G R R B G B R G) 7       *
0     (G R G B G B G B R R G G G R G G B G R R B G B R G) 6       *
0     (G R G B G B G B R R G G B R G G B G R R B G B R G) 7       *
0     (G R G B G B G B R R G G B R G G B G R R B G B R G) 7
0     (G R G B G B G R R R G G B R G G B G R R B G B R G) 6       *
0     (G R G B G B G R R R G G B R G G B G R R B B B R G) 7       *
0     (G R G B G B G R R R G G B R G G B G R R B B B R G) 6       *
0     (G R G B G B G R R R G G B R G G B G R R B B B R G) 6
0     (G R G B G B G R R R G G B R G G B G R R B B B R G) 6
NIL
```

Task 9: Copy

Source Code

```
( setf *copy-demo* nil )

( defconstant *pc-c* 40 )

( defmethod perform-copies ( ( cp population ) ( np population))
      ( dotimes ( i ( nr-copies ) )
            ( perform-one-copy cp np )
      )
)

( defmethod nr-copies ()
      ( * ( / *pc-c* 100 ) *population-size* )
)

( defmethod perform-one-copy ( ( cp population ) ( np population ) &aux x m mm
new-i )
      ( setf m ( select-individual cp ) )
      ( if *copy-demo* ( format t "Selected individual = ~%" ) )
      ( if *copy-demo* ( display m ) )
      ( setf mm ( maybe-mutate m ) )
      ( if *copy-demo* ( format t "Possibly muted individual = ~&" ) )
      ( if *copy-demo* ( display mm ) )
      ( setf ( individual-number mm ) ( + 1 ( size np ) ) )
      ( if *copy-demo* ( format t "Renumbered individual = ~&" ) )
      ( if *copy-demo* ( display mm ) )
      ( setf new-i ( new-individual ( + 1 ( size np ) ) (
individual-rbg-string mm ) ) )
      ( setf
            ( population-individuals np )
            ( append ( population-individuals np ) ( list new-i ) )
      )
      nil
)

( defmethod empty-population ( ( cp population ) &aux np )
      ( setf np ( make-instance 'population ) )
      ( setf ( population-individuals np ) () )
      ( setf ( population-generation np ) ( + 1 ( population-generation cp ) ) )
)
      np
)

( defmethod perform-copies-demo ( &aux cp np )
      ( setf cp ( initial-population ) )
      ( setf np ( empty-population cp ) )
      ( format t
"------------------------------------------------------------------------~%
~%")
      ( display np )
      ( format t
```

```
"~%~%--------------------------------------------------------------------------
--~%")
      ( setf *select-demo* t )
      ( setf *copy-demo* t )
      ( dotimes ( i 10 )
            ( perform-one-copy cp np )
            ( format t
"-------------------------------------------------------------------------~%
~%")
            ( display np )
            ( format t
"~%~%--------------------------------------------------------------------------
--~%")
      )
      ( setf *select-demo* nil )
      ( setf *copy-demo* nil )
      nil
)
```

Demo

```
…

------------------------------------------------------------------------
the sample of individuals ...
42    (R B R G R G G R G G G B R R G B B G B B B B R G R) 8
4     (R G G R B B R R G G G B B G B B B G R B B R G B G) 10
11    (B B B B G B G B G R R G R B R B R B R R G B R B G) 11
82    (B R B B B G G B B R B G R G B R R R R R B R G B R) 10
46    (B B G B B G R B R B B B R B G R R R G B R B G G R) 11
15    (G G B G G G G R R B G R R R R B R R R B B R R G G) 5
54    (R B G R B G B B B R B G B G R G R G G B B G B R G) 10
84    (R B R R B B B G B R B R R R G B B B G B B B B G G) 13

the most fit of the sample ...
84    (R B R R B B B G B R B R R R G B B B G B B B B G G) 13

Selected individual =
84    (R B R R B B B G B R B R R R G B B B G B B B B G G) 13
Possibly muted individual =
84    (R B R R B B B G B R B R R R G B B B G B B B B G G) 13
Renumbered individual =
10    (R B R R B B B G B R B R R R G B B B G B B B B G G) 13
------------------------------------------------------------------------




Generation 1 population ...
1     (G G G B B R B B B R R G B R B B B G G B B B R B G) 13
2     (B G B B G G R B B B B B B B G B G G B B B B R B G) 16
3     (B G B B R G R B B B B B B B B G B G G B B B B R R G) 15
4     (R G G R B B R R G G G B B G B B B G R B B R G B G) 10
5     (B B R R B R R R R B G B B R B B B R R R B B R R B) 12
```

```
6     (B R G B G B B R G R R B B G B B G R G G B R R B B) 11
7     (B B B B R R R R B B B B R G G G R R B G B B R B R) 12
8     (B R B G G B R B G B R B G B G B R G B B G G R G G) 10
9     (G G G B B R R R R R B B B B B B G B B R B G G G) 12
10    (R B R R B B B G B R B R R R G B B B G B B B B G G) 13



------------------------------------------------------------------------
nil
```

Task 10: Crossover

Source Code:

```
( setf *crossover-demo* nil )

( defconstant *pc-x* 60 )

( defmethod perform-crossovers ( ( cp population ) ( np population ) )
       ( dotimes ( i ( nr-crossovers ) )
              ( perform-one-crossover cp np )
       )
)

( defmethod nr-crossovers ()
       ( * ( / *pc-x* 100 ) *population-size* )
)

( defmethod perform-one-crossover ( ( cp population ) ( np population ) )
       ( let ( x m mm mother father new-i )
              ( setf mother ( select-individual cp ) )
              ( setf father ( select-individual cp ) )
              ( if *crossover-demo* ( format t "Selected mother = ~%" ) )
              ( if *crossover-demo* ( display mother ) )
              ( if *crossover-demo* ( format t "Selected father = ~&" ) )
              ( if *crossover-demo* ( display father ) )
              ( setf m ( crossover mother father ) )
              ( if *crossover-demo* ( format t "the crossover = ~&" ) )
              ( if *crossover-demo* ( display m ) )
              ( setf mm ( maybe-mutate m ) )
              ( if *crossover-demo* ( format t "the possibly mutated individual = ~&" ) )
              ( if *crossover-demo* ( display mm ) )
              ( setf ( individual-number mm ) ( + 1 ( size np ) ) )
              ( if *crossover-demo* ( format t "the renumbered individual = ~&" ) )
              ( if *crossover-demo* ( display mm ) )
              ( setf new-i ( new-individual ( + 1 ( size np ) ) ( individual-rbg-string mm
) ))
              ( setf
                     ( population-individuals np )
                     ( append ( population-individuals np ) ( list new-i ) )
              )
       )
       nil
)
```

```
; The inner crossover function called here takes the rbg-string,
; which is a previously established function
( defmethod crossover ( ( mother individual ) ( father individual ) &aux mi fi x i )
       ( setf mi (individual-rbg-string mother ) )
       ( setf fi (individual-rbg-string father ) )
       ( setf x ( crossover mi fi ) )
       ( setf i ( new-individual 0 x ) )
       i
)
```

Demo:

```
…

the sample of individuals ...
20    (B B G R R G R B B R G R R G B B G B G G G B R R R) 8
63    (R G R R B R R B R G B G G G B G R R R G B R B B G) 7
9     (R B B R G B G R G B G B B R G B B B G G B G R G B) 11
7     (G G R R G B R G R R G B R R B B R G R B R B R R G) 6
18    (G B B B B G B R R G R B G R R R R R B B G R G R B) 9
29    (R R G R R R B B G G G R B R G R R R R G R B B R G) 5
62    (B G G G G G R G B R G R R G R R R G B R R G G G G R) 3
7     (G G R R G B R G R R G B R R B B R G R B R B R R G) 6

the most fit of the sample ...
9     (R B B R G B G R G B G B B R G B B B G G B G R G B) 11

the sample of individuals ...
73    (B R B R R B G G R B R R G B G R G G G B R R B G B B G) 9
31    (G B B G R R B B B B G R G B B R R R B G R B B G G) 11
70    (G R G B G G G B R R G G B B G B G B B B R R B R R G) 9
58    (B B B G B G R R B G G B G B G B G B B G B B R G R) 12
60    (R B R R G B B R B R B G R B G G B R R B G B R R R) 9
74    (R R G B R R B G B B B B G B R G G G B B R G G R R) 9
94    (G B B G R B R B G B R R G G B B G G G R B R B G B) 10
72    (B R G G B R B G R R R G R R B G B G R R R R R G B) 6

the most fit of the sample ...
58    (B B B G B G R R B G G B G B G B G B B G B B R G R) 12

Selected mother =
9     (R B B R G B G R G B G B B R G B B B G G B G R G B) 11
Selected father =
58    (B B B G B G R R B G G B G B G B G B B G B B R G R) 12
the crossover =
0     (R B B R G B R R B G G B G B G B G B B G B B R G R) 11
the possibly mutated individual =
0     (R B B R G B R R B G G G G B G B G B B G B B R G R) 10
the renumbered individual =
10    (R B B R G B R R B G G G G B G B G B B G B B R G R) 10
--------------------------------------------------------------------------



Generation 1 population ...
```

```
1     (B G R R G B R R G B R G B R B B R G B B R G R G B) 9
2     (B B B G G B G R B G B G B G B R B R R B B B R R B) 13
3     (B R B B G G B R R G R B G B R B B R R R B G B B R) 11
4     (B R R B B B R B B R B B B G R R B G G R R G G G B) 11
5     (R G B B B G G G B G G G B G B B G G R B B B R R B) 11
6     (R G B R B B R R R G R B B B B G B B R B B B R R B) 13
7     (R G G B R G B R B G B B R R G B G B G R R B B G B) 10
8     (R B B R G B G R G B G B B R G B B B G G B G R R G) 10
9     (B B B B B B R B B R B B B G R R B G G R R G B B B) 15
10    (R B B R G B R R B G G G G B G B G B B G B B R G R) 10



--------------------------------------------------------------------------
nil
```

Task 11:

Source Code:

```
;; THE NEXT GENERATION METHOD FOR THE GA
( defmethod next-generation ( ( cp population ) &aux np )
      ( setf np ( empty-population cp ) )
      ( perform-copies cp np )
      ( perform-crossovers cp np )
      np
)

;; THE GA!
( defconstant *nr-generations* 25 )

( defmethod ga ( &aux p )
      ( format t "THE WORLD IS BLUE ~%~%" )
      ( setf *fitness* #'fitness-b )
      ( setf p ( initial-population ) )
      ( terpri )
      ( summarize p )
      ( dotimes ( i *nr-generations* )
            ( setf p ( next-generation p ) )
            ( check-average p )
      )
      ( terpri )
      ( summarize p )
      ( format t "THE WORLD IS RED ~%~%" )
      ( setf *fitness* #'fitness-r )
      ( dotimes ( i *nr-generations* )
            ( setf p ( next-generation p ) )
            ( check-average p )
```

```
      )
      ( terpri )
      ( summarize p )
      ( format t "THE WORLD IS GREEN ~%~%" )
      ( setf *fitness* #'fitness-g )
      ( dotimes ( i *nr-generations* )
            ( setf p ( next-generation p ) )
            ( check-average p )
      )
      ( terpri )
      ( summarize p )
)

;; METHODS TO PROVIDE INFORMATION ON "PROGRESS"
( defmethod summarize ( ( p population ) )
      ( display p )
      ( check-average p )
      ( terpri )
)

( defmethod check-average ( ( p population ) )
      ( format t "average fitness of population ~A = ~A~%"
            ( population-generation p )
            ( average p )
      )
)
```

Demo:

```
[1]> (load "rbg.l")
;; Loading file rbg.l ...
;; Loaded file rbg.l
T
[2]> (ga)
THE WORLD IS BLUE




Generation 0 population ...
1     (G R R G R G G B R B R B B B G R G G B R R R G B R) 7
2     (G B B R G B G B B R G R R R G R R G G B G G B G R) 7
3     (R B B G B G B G R B G B B B G B R R G B B G B B B) 14
4     (G R G B R G R G R G G R G B G G B G B R R R R G G) 4
5     (G R B G G R B B R B R B R R G R R R G B B B B R G) 9
6     (G G R B G G G R B R R B G R B G R B R R G R R B B G) 7
7     (R G R G G B R G B B G G G G B B G B B G G G R R G) 7
8     (G G G R R R R R R R G G R G G B G B G B B R R R R) 4
9     (R G R B B G B R G G R G B R R G G B B R G B R B B) 9
10    (B R R G R R R R G R G B R R B B R R R B B B R B R) 8
11    (R R G G G B G R G G B R G R B G G R B R B B R G G) 6
12    (R R R R B B R B G R R G R R R G R R B R G G R B B) 6
13    (B G R B R G B B B B B B R G R R G G G B B G R B R) 11
```

```
14    (G B R G G R R G G R B B B G R R G R B R R R R B R) 6
15    (G B B G B G B G R R G R R G B R B R G R B B G R R) 8
16    (G B G B B G R G R G R G B G R B R G G G B R G B B) 8
17    (B G R B R G R R R R B B R G R G G G R R B R B B G) 7
18    (R G G B R G R G B G R B R R B R G B G R R G R G R) 5
19    (B B B G G B R G G G R B G R R G G G B G B B G R B G) 9
20    (R R B R B B R B B B G R R G G G B B G G B G G G B) 10
21    (R G B R B G R B G R G G G B B G B G R B R R R R R) 7
22    (R B G B R G G B R B G B G G G G G G B R R B B G G) 8
23    (G B G B R R R R G B G B B R G B G R B G B R G R B) 9
24    (G B R B G R G G G G B R B R B R G B G G B G B R R) 8
25    (R B G R B B R R B B G G G R G R B R G G B R B R R) 8
26    (G G G G R R B B G R B B B G G G G R B G R G G G B) 7
27    (R R R B G G B G B G R G B G R G G G B G G R G R B) 6
28    (B G G G R R R B G B G G R B B G B G B G R B G R B) 9
29    (R G R R R R R B R R G R B B G R R G R G B B G B G) 6
30    (G B G G G G G B R G B G G G B R G G R B B G G B B G B) 9
31    (B R G B B B B G R B G B R R R G B R G R B B R R B G) 10
32    (B R G R B R R R G R G G B R B R R R G R R B B G B) 7
33    (G G G G R G B G B B R G G G G G R R G R B R R G R R) 4
34    (B B G G R B B R R B B B R R B R G G B G R R G R B) 10
35    (G G B R B R R B G B G B R R G B R B G B G R R R G) 8
36    (G G B B R R B R G G R R R G R G R R G G G R R G B) 4
37    (B R R B G G R G R B R R G R B R R B R R R B G R B) 7
38    (G G R B R R R G G B B B G G R G B R G G B B R G B) 8
39    (G B G B G R R G R R B B G R R G B G B G R G B R G) 7
40    (B G B B B R G B B B B G B R B R B G G G R B R R R) 12
41    (R G R B G R G G B G G B G R B R B R B B B R B G B) 10
42    (B R G B R G B B G G G G G B G R G G G G B G R B G G) 7
43    (B B B B B G R B B R G B G G G G G B G R G R G G G) 9
44    (B R G R G R R G B G G R B B G R G B R R G G R B G) 6
45    (B B B G R R G B B B R R B R B G G R B G B B R G B) 12
46    (G G R B B R B B R B G G B R B B B B R B G R G R B) 12
47    (G R G B R B G B G R R R G R B G G G R R R B R B R) 6
48    (B B B G G R B B G R R G G B R G R G B R B B B B G) 11
49    (R R R G G R B B R G G B G G G G B R B B G R R G G G) 6
50    (G R B B R R B B G R B R R G B R G R R G B R G R G) 7
51    (R R G R R G B G G R R G B B G R G G R B B B B B R) 8
52    (G B R B B B B R B R R G G G R B B R R R G B R R R) 9
53    (B G R G R B G R B G R R B R R B G R G B R G G B R) 7
54    (R R R G B R G R R R B B B G B B R G B B R G G R G) 8
55    (G R G R R R B G R R R G G G R B R B G R B B B G G) 6
56    (G R R R B B G B G G R R G R B R R R B R B B R G R) 7
57    (R G G R G R R B G G G R B R B G R G R B R B R B R) 6
58    (B B B B B B R R G G R G B B R G B G B R G G G B B) 12
59    (R B G G R R G R B G B B G R R G B B R R G B B R R) 8
60    (R B G B B R B B G G R R G B B G R R G B G G R B G) 9
61    (B G B R R B G G R R B R G R G R R B G B R B G G G) 7
62    (B B G R R R B B G G G R R G G R B B R G R B R G R) 7
63    (G G R G R G G G R B R B G R G B G R B R R G B B B) 7
64    (G G G G G B B R R G G G G B R B B R R G G G G R R) 5
65    (B B R B R B G R R R B B G B B B G G B R G B B G R) 12
66    (R R B R B G B R B G G R G B B R B G B R B G B R G) 10
67    (R B R G B B G B G G B G G B R G R G B R G B B B B) 11
68    (R G G R R R R R B R G B B B R B R B G R G R R B B) 8
69    (G B B B B R B G G G B G G G R B G G G G R G G G B) 8
```

```
70    (G R R G B B B G G B B R R B G R G B B B B G G B G) 11
71    (B G G R G R B G B G R G R R B B G G B B B G B B G) 10
72    (B G B G R B B G G R G R G R G B R G B B B G G B B) 10
73    (G G B R G B G B G B R R R G R R B B G B B G R G G) 8
74    (G R B B B R G G R R G G R G R G B R R G G B G B G) 6
75    (G B G B R G G G R B R G R R R R B G G B B G G R B) 7
76    (R B G G R G B R G B G G R R G B G B G G R R G B R) 6
77    (G G R B G B R G B R R R G G G G G R R G G B R G R) 4
78    (R G G R G G B R R B R B R R G B B B R R R G R R B) 7
79    (B R G G B B B B B B R R R B R G G B B G G R G G G) 10
80    (B R R R R G B R B R G B G B B G G G B R G R B G B) 9
81    (B R R G B R R G B R B R R G B B R G G R G G G G B) 7
82    (B B G R B B G R R B R R R R R R G B R G R G G B R) 7
83    (B B B B G G B G G G B B R R R R G R R R R G G B G) 8
84    (B B B G B R B R R B G B G R R G G R G B B B R R R) 10
85    (B R G B G B G B R B G B B G B G G G B R B G B R G B) 12
86    (B G R G R B R G B G R G B B R R B G R G R B R R R) 7
87    (B R G R B R B R R G G R R B R B R G G B R B B R B) 9
88    (B B B R B G R G R R G G G R B B B G R R R R R G G) 7
89    (G G B G R G R R G R G G G G B R G B R B R B B B B) 8
90    (G G R B R G B G B G R G R B B B B B G B G R B G G) 10
91    (R G R B B B B G G R B B G R G G B G G B G B G R R) 9
92    (B B R G B G B B B G B R R G G R G R G B R B B R B) 11
93    (G R G G B R B G R R R B R R R G G B B G G B R R R) 6
94    (G B G B B B B R G G G R B B B G R G R B R B G G G) 10
95    (G B R G G R G G R G R B G R B G G B G R R G G B B) 6
96    (G G G B G B R R B B B G G R R B G B R R R B R R R) 8
97    (G B B G R G G R G B R G R B G R R G G R G R G B G) 5
98    (R R R B R R B B R G B R R B B B G B G B B G B B B) 13
99    (R B R G B B B R G B G B G R R R G G G R G R B G R) 7
100   (R B B G G R G B B R B R G G B G R G G G G G R G G) 6

average fitness of population 0 = 8.01

average fitness of population 1 = 10.71
average fitness of population 2 = 12.85
average fitness of population 3 = 14.47
average fitness of population 4 = 15.72
average fitness of population 5 = 16.89
average fitness of population 6 = 18.13
average fitness of population 7 = 19.08
average fitness of population 8 = 20.39
average fitness of population 9 = 21.79
average fitness of population 10 = 22.44
average fitness of population 11 = 22.99
average fitness of population 12 = 23.53
average fitness of population 13 = 23.72
average fitness of population 14 = 23.99
average fitness of population 15 = 24.36
average fitness of population 16 = 24.48
average fitness of population 17 = 24.54
average fitness of population 18 = 24.57
average fitness of population 19 = 24.51
average fitness of population 20 = 24.49
average fitness of population 21 = 24.5
average fitness of population 22 = 24.45
```

```
average fitness of population 23 = 24.54
average fitness of population 24 = 24.44
average fitness of population 25 = 24.43



Generation 25 population ...
1     (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
2     (B B B B B B B B B B B B B B B B B B B B B R B B B) 24
3     (B B B B B B B B B B B B B B B B B B B B G B B B B) 24
4     (B B B B B B B B B B B B B R B B B B B B B B B B B) 24
5     (B B B B B B B B B B B B B B B B B G B B B B B B B) 24
6     (B B B B B B B G B B B B B B B B B B B B B B B B B) 24
7     (B B R B B B B B B B B B B B B B B B B B B B B B B) 24
8     (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
9     (B B B B B B B B B B B B G B B B B B B B B B B B B) 24
10    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
11    (B B B B B B B B G B B B B B B B B B B B B B B B B) 24
12    (B B B B B G B B B B B B B B B B B B B B B B B B B) 24
13    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
14    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
15    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
16    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
17    (B B B B B B B B B B B B B B B G B B B B B B B B) 24
18    (B B B B B B B B B B B B B B B B B B B B B B B B R) 24
19    (B B B B B B B B B B B B B B B B B B B G B B B B) 24
20    (B B B B B B B B B B R B B B B B B B B B B B B B) 24
21    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
22    (B B R B B B B B B B B B B B B B B B B B B B B B B) 24
23    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
24    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
25    (B B B B B B B B B B B B B B B G B B B B B B B B) 24
26    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
27    (B B B B B B B B R B B B B B B B B B B B R B B) 23
28    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
29    (B B B B B B B B B B B B B B B B B B B G B B B B) 24
30    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
31    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
32    (B B B B B B B B B R B B B B B B B B B B B B B B) 24
33    (B B B B B B B B B B B B B B B B B B B B B R B B) 24
34    (B B B B B B B B B B B B B B B B B B B B B B R B) 24
35    (B B B B B B B B B B G B B B B B B B B B B B B B) 24
36    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
37    (B B B G B B B B B B B B B B B B B B B B B B B B) 24
38    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
39    (B R B B B B B B B B B B B B B B B B B B B B B B) 24
40    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
41    (B B B B B B B B B B B B B B B B B B B B R B B B) 24
42    (B B B B R B B B B B B B B B B B B B B B B B B B) 24
43    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
44    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
45    (B B B B B B B B B B B B B B R B B B B B B B B B) 24
46    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
47    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
48    (B B B B B B B B B B G B B B B B B B B B B B B B) 24
49    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
```

```
50    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
51    (B B B B B B B B B B B B B B B B B B B B B B B B R) 24
52    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
53    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
54    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
55    (B B B B B B R B B B B B B B B B B B B B B B B B B) 24
56    (B B B B B B B G B B B B B B B B B B B B B B B B B) 24
57    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
58    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
59    (B B B B B R B B B B B B B B B B B B B B B B B B B) 24
60    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
61    (B B B B B B B R B B B B B B B B B B B B B B B B B) 24
62    (G B B B B B B B B B B B B B B B B B B B B B B B B) 24
63    (B B B B B B B B B B B B B B B B B G B B B B B B B) 24
64    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
65    (B B B B B B B B B B B B B B B B B B B G B B B B B) 24
66    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
67    (B B B B B B B B B R B B B B B B B B B B B B B B B) 24
68    (B B B B B B B B B B B B B B B B B R B B B B B B B) 24
69    (B B B B B B B B B G B B B B B B B B B B B B B B B) 24
70    (B B B B B B B B B B B B B B B B B R B B B B B B B) 24
71    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
72    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
73    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
74    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
75    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
76    (G B B B B B B B B B B B B B B B B B B B B B B B B) 24
77    (B B B B B B B B B B B B B B B B B B B B B R B B B) 24
78    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
79    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
80    (B B B B B B B G B B B B B B B B B B B B B B B B B) 24
81    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
82    (B B B R B B B B B B B B B B B B B B B B B B B B B) 24
83    (B B B B B B B B B B B B B B B B B B B B R B B B B) 24
84    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
85    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
86    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
87    (B B B B B B R B B B B B B B B B B B B B B B B B B) 24
88    (B B B B B B B B B B B B B B B B B B B B B B B B R) 24
89    (G B B B B B B B B B B B B B B B B B B B B B B B B) 24
90    (B B B B B G B B B B B B B B B B B B B B B B B B B) 24
91    (B B B B B B B B B G B B B B B B B B B B B B B B B) 24
92    (B B B G B B B B B B B B B B B B B B B B B B B B B) 24
93    (B B B B B B B B B B B G B B B B B B B B B B B B B) 24
94    (B B B B B B B B B R B B B B B B B B B B B B B B B) 24
95    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
96    (B B B B B B B B B B B B B B B B B B B B B B B B B) 25
97    (B B B B B B B B B B B B G B B B B B B B B B B B B) 24
98    (B B B B B B B B B B B B B B B G B B B B B B B B B) 24
99    (B B B B B B B B B B B B B B B B B R B B B B B B B) 24
100   (B B B B B B B B B B B B B B B B B B B B B B B B G) 24

average fitness of population 25 = 24.43

THE WORLD IS RED
```

```
average fitness of population 26 = 1.27
average fitness of population 27 = 2.36
average fitness of population 28 = 3.79
average fitness of population 29 = 5.45
average fitness of population 30 = 7.55
average fitness of population 31 = 9.4
average fitness of population 32 = 10.61
average fitness of population 33 = 11.75
average fitness of population 34 = 13.4
average fitness of population 35 = 15.07
average fitness of population 36 = 16.72
average fitness of population 37 = 17.62
average fitness of population 38 = 18.75
average fitness of population 39 = 19.79
average fitness of population 40 = 20.71
average fitness of population 41 = 21.86
average fitness of population 42 = 22.47
average fitness of population 43 = 23.06
average fitness of population 44 = 23.72
average fitness of population 45 = 24.33
average fitness of population 46 = 24.47
average fitness of population 47 = 24.41
average fitness of population 48 = 24.46
average fitness of population 49 = 24.56
average fitness of population 50 = 24.52



Generation 50 population ...
1     (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
2     (R R R R R R G R R R R R R R R R R R R R R R R R R) 24
3     (R R R R R R R R R R R R R R R B R R R R R R R R R) 24
4     (R R R R R R R R R B R R R R R R R R R R R R R R R) 24
5     (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
6     (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
7     (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
8     (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
9     (R R R R R R R R R R R R R R R R R R R R R G R R R) 24
10    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
11    (G R R R R R R R R R R R R R R R R R R R R R R R R) 24
12    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
13    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
14    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
15    (R R R R R R R R B R R R R R R R R R R R R R R R R) 24
16    (R R R R R R R R R R R B R R R R R R R R R R R R R) 24
17    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
18    (R R R R R R R R R R R R R R R R R R R R B R R R) 24
19    (R R R R R R R R R R R R R R R R R R R R R R R R B) 24
20    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
21    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
22    (R R R R R R R R R R R R R R R B R R R R R R R R) 24
23    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
24    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
25    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
26    (R R R R R R R R R R R R R R R R R R R B R R R) 24
27    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
```

```
28    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
29    (R R R R R R R R R R R R R R R R R R R R R R R G R) 24
30    (R R R R R R R R R R R R B R R R R R R R R R R R R) 24
31    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
32    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
33    (R R R R R G R R R R R R R R R R R R R R R R R R R) 24
34    (R R R R R R R R R R R R R R R R R R R G R R R R) 24
35    (R R R R R R R R R R R R R R R G R R R R R R R R) 24
36    (R R R R R R R R R R R R R R R R G R R R R R R R) 24
37    (R R R R R R R R G R R R R R R R R R R R R R R R) 24
38    (R R R R R R R R R R R R R R R R R R R B R R R R) 24
39    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
40    (R R R R R R R R R R R B R R R R R R R R R R R R R) 24
41    (R R R R R R R R R B R R R R R R R R R R R R R R R) 24
42    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
43    (R R R R R R R R R R R R R R R R R R G R R R R R R) 24
44    (R B R R R R R R R R R R R R R R R R R R R R R R R) 24
45    (R R R R R R R R R R R R R R R R R R R R R R G R R) 24
46    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
47    (R R R R R R R R B R R R R R R R R R R R R R R R R) 24
48    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
49    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
50    (R R R R R R R R R R G R R R R R R R R R R R R R R) 24
51    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
52    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
53    (R R R R R R R R R R R R R R R R R R R R R R G R R) 24
54    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
55    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
56    (R R R R R R R R R R B R R R R R R R R R R R R R R) 24
57    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
58    (R R R R R R R R R R R G R R R R R R R R R R R R R) 24
59    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
60    (R R R R R R R R R B R R R R R R R R R R R R R R R) 24
61    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
62    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
63    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
64    (R R R R R R R R R R R R G R R R R R R R R R R R R) 24
65    (R R R R R R R B R R R R R R R R R R R R R R R R R) 24
66    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
67    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
68    (R R R R R R R R G R R R R R R R R R R R R R R R R) 24
69    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
70    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
71    (R R R R R R R R R G R R R R R R R R R R R R R R R) 24
72    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
73    (R R R R G R R R R R R R R R R R R R R R R R R R R) 24
74    (R R R R R R R R R B R R R R R R R R R R R R R R R) 24
75    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
76    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
77    (R R R R R R R B R R R R R R R R R R R R R R R R R) 24
78    (B R R R R R R R R R R R R R R R R R R R R R R R R) 24
79    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
80    (R R G R R R R R R R R R R R R R R R R R R R R R R) 24
81    (R R R R R R R R R B R R R R R R R R R R R R R R R) 24
82    (R R R R R R R R R R R R R R R R R R R R R R R R R) 25
83    (R R R R G R R R R R R R R R R R R R R R R R R R R) 24
```

```
84    (R R R R R R R R R R R R R G R R R R R R R R R) 24
85    (R R R R R R R R R R R R R R R R R R R R R R R) 25
86    (R G R R R R R R R R R R R R R R R R R R R R R) 24
87    (R R R R R R R R R R R R R R R R R R R R R R R) 25
88    (R R R R R R R R R R B R R R R R R R R R R R R) 24
89    (R R R R R R R R R R R R R R R R R R R R R R R) 25
90    (R R R R R R R R R R R R R R R R R R R R R R R) 25
91    (R R R R R R R R R R R R R R R R R R R R R R R) 25
92    (R R R R R R R R R R R R R R R R R R R R R R R) 25
93    (R R R R R R R R R R R R R R R R R R R R R R R) 25
94    (R R R R R R R R R R R R R R R R R R R R R R R) 25
95    (R R R R R R R R R R R R R R R R R R R R R R R) 25
96    (R R R R R R R R R R R R R R R R R R R R R R R) 25
97    (R R R R R R R G R R R R R R R R R R R R R R R) 24
98    (R R R R R R R R R R R R R R R R R R R B R R R) 24
99    (R R R R R R R R R R R R R R R R R R R B R R R) 24
100   (R R R R B R R R R R R R R R R R R R R R R R R) 24

average fitness of population 50 = 24.52

THE WORLD IS GREEN

average fitness of population 51 = 1.13
average fitness of population 52 = 2.31
average fitness of population 53 = 3.43
average fitness of population 54 = 5.27
average fitness of population 55 = 6.6
average fitness of population 56 = 7.73
average fitness of population 57 = 9.15
average fitness of population 58 = 10.45
average fitness of population 59 = 12.13
average fitness of population 60 = 13.89
average fitness of population 61 = 15.56
average fitness of population 62 = 16.66
average fitness of population 63 = 17.58
average fitness of population 64 = 18.63
average fitness of population 65 = 19.66
average fitness of population 66 = 20.75
average fitness of population 67 = 21.64
average fitness of population 68 = 22.25
average fitness of population 69 = 23.01
average fitness of population 70 = 23.55
average fitness of population 71 = 23.99
average fitness of population 72 = 24.39
average fitness of population 73 = 24.44
average fitness of population 74 = 24.52
average fitness of population 75 = 24.54



Generation 75 population ...
1     (G G G G G G G G G G G G G G G G G G G G G G G) 25
2     (G G G G G G G G G G G G G G G G G G G G G G G) 25
3     (G B G G G G G G G G G G G G G G G G G G G G G) 24
4     (G G G G G G G G G G G G G G G G G G G G G G G) 25
5     (G R G G G G G G G G G G G G G G G G G G G G G) 24
```

```
6     (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
7     (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
8     (G G G G G B G G G G G G G G G G G G G G G G G G G) 24
9     (G G G G G G G G G G B G G G G G G G G G G G G G G) 24
10    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
11    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
12    (G G G G G G G G G G G G G G G G G G G G G G B G G) 24
13    (G G G G G G G G G G G G G G G G R G G G G G G G G) 24
14    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
15    (G G G G G G G G G G G R G G G G G G G G G G G G G) 24
16    (B G G G G G G G G G G G G G G G G G G G G G G G G) 24
17    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
18    (G G G G G G G G G G G G G G G B G G G G G G G G G) 24
19    (G G G G G G B G G G G G G G G G G G G G G G G G G) 24
20    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
21    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
22    (G G G G G G G G G G G G G G G G G G G R G G G G) 24
23    (G R G G G G G G G G G G G G G G G G G G G G G G G) 24
24    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
25    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
26    (G G G G G G B G G G G G G G G G G G G G G G G G G) 24
27    (G G G G G B G G G G G G G G G G G G G G G G G G G) 24
28    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
29    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
30    (G G G G G G G G G G G G G B G G G G G G G G G G G) 24
31    (G G G G G G G G G B G G G G G G G G G G G G G G G) 24
32    (G G G G G G G G G G G G G G G G G B G G G G G G G) 24
33    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
34    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
35    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
36    (G G G G G G G G G G G G G R G G G G G G G G G G G) 24
37    (R G G G G G G G G G G G G G G G G G G G G G G G G) 24
38    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
39    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
40    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
41    (G G G G G G G G G G G G G G G G G G G G G G G R G) 24
42    (R G G G G G G G G G G G G G G G G G G G G G G G G) 24
43    (G G G G G G G G R G G G G G G G G G G G G G G G G) 24
44    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
45    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
46    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
47    (G G G G G G G G R G G G G G G G G G G G G G G G G) 24
48    (G G G G G G G G G G G G G G G G G G G B G G G G G) 24
49    (G G G G G G G G G G G G G B G G G G G G G G G G G) 24
50    (G G G G G G G G G G G G G R G G G G G G G G G G G) 24
51    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
52    (G G G G G G G G G G G G G G G B G G G G G G G G G) 24
53    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
54    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
55    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
56    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
57    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
58    (G G G G G G G G G G G G G G G G G G G G G G G G G) 25
59    (G G G G G G G G G G G G G G G G G G G B G G G G) 24
60    (G G G G G G G G G G G G G R G G G G G G G G G G G) 24
61    (G G G G G G G R G G G G G G G G G G G G G G G G G) 24
```

```
62    (G G G G G G G G G G G G G G G G G B G G G G G G) 24
63    (G G G G G G G G G G G G G G G G G G G G G G B G) 24
64    (G G G G G G G G R G G G G G G G G G G G G G G G) 24
65    (G G G G G G G G G G G G G G G G G G G B G G G G) 24
66    (G G G G G G G B G G G G G G G G G G G G G G G G) 24
67    (R G G G G G G G G G G G G G G G G G G G G G G G) 24
68    (G G G G G R G G G G G G G G G G G G G G G G G G) 24
69    (G G G G G G G B R G G G G G G G G G G G G G G G) 23
70    (G G G G G G G G G G G G G G G G B G G G G G G G) 24
71    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
72    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
73    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
74    (G G G G G R G G G G G G G G G G G G G G G G G G) 24
75    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
76    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
77    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
78    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
79    (G G G G G G G G G R G G G G G G G G G G G G G G) 24
80    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
81    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
82    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
83    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
84    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
85    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
86    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
87    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
88    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
89    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
90    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
91    (G G G G G G G G G G G G G G G B G G G G G G G G) 24
92    (G G G G G G G B G G G G G G G G G G G G G G G G) 24
93    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
94    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
95    (G G G G G G G G B G G G G G G G G G G G G G G G) 24
96    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
97    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
98    (G G G G G G G G G G G G G G G G G G G G G G G G) 25
99    (G G G G G G G G B G G G G G G G G G G G G G G G) 24
100   (G G G G G G G G G G G G G G G G G G G G G G G G) 25

average fitness of population 75 = 24.54

NIL
```