tick4star submission from James Wood

Name	James Wood (jdw74)
College	ROBIN
Submission contents	uk/ac/cam/jdw74/tick4star/ExceptionTest.java uk/ac/cam/jdw74/tick4star/Pattern.java uk/ac/cam/jdw74/tick4star/PatternFormatException.java uk/ac/cam/jdw74/tick4star/PatternLife.java uk/ac/cam/jdw74/tick4star/PatternLoader.java uk/ac/cam/jdw74/tick4star/Repeat.java uk/ac/cam/jdw74/tick4star/Statistics.java uk/ac/cam/jdw74/tick4star/StatisticsLife.java uk/ac/cam/jdw74/tick4star/output.txt
Ticker	UNKNOWN
Ticker signature	

ExceptionTest.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  class ExceptionTest {
 3      public static void main(String[] args) {
 4          System.out.print("C");
 5          try {
 6              a();
 7          } catch (Exception e) {
 8              System.out.print(e.getMessage());
 9          }
10          System.out.println("A");
11      }
12
13      public static void a() throws Exception {
14          System.out.print("S");
15          b();
16          System.out.print("J");
17      }
18
19      public static void b() throws Exception {
20          System.out.print("T");
21          if (1+2+3==6)
22              throw new Exception("1");
23          System.out.print("V");
24      }
25  }
```

Pattern.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  import java.text.ParseException;
 3
 4  public class Pattern {
 5
 6      private String name;
 7      private String author;
 8      private int width;
 9      private int height;
10      private int startCol;
11      private int startRow;
12      private String cells;
13
14      public String getName() { return name; }
15      public void setName(String x) { name = x; }
16
17      public String getAuthor() { return author; }
18      public void setAuthor(String x) { author = x; }
19
20      public int getWidth() { return width; }
21      public void setWidth(int x) { width = x; }
22
23      public int getHeight() { return height; }
24      public void setHeight(int x) { height = x; }
25
26      public int getStartCol() { return startCol; }
27      public void setStartCol(int x) { startCol = x; }
28
29      public int getStartRow() { return startRow; }
30      public void setStartRow(int x) { startRow = x; }
31
32      public String getCells() { return cells; }
33      public void setCells(String x) { cells = x; }
34
35      public Pattern(Pattern p) {
36          name = p.name;
37          author = p.author;
38          width = p.width;
39          height = p.height;
40          startCol = p.startCol;
41          startRow = p.startRow;
42          cells = p.cells;
43      }
44
45      public Pattern(String format) throws PatternFormatException {
46          String[] parts = format.split(":");
47          if (parts.length < 7)
48              throw new PatternFormatException("Too few arguments");
49          if (parts.length > 7)
50              throw new PatternFormatException("Too many arguments");
51
52          name = parts[0];
53          author = parts[1];
54          try {
55              width = Integer.parseInt(parts[2]);
56              if (width <= 0) throw new NumberFormatException();
57          }
58          catch (NumberFormatException e) {
59              throw new PatternFormatException(
60                  "Width argument not a positive integer");
61          }
62          try {
63              height = Integer.parseInt(parts[3]);
64              if (height <= 0) throw new NumberFormatException();
65          }
66          catch (NumberFormatException e) {
67              throw new PatternFormatException(
68                  "Height argument not a positive integer");
69          }
70          try {
71              startCol = Integer.parseInt(parts[4]);
```

```
72          }
73          catch (NumberFormatException e) {
74              throw new PatternFormatException(
75                  "x coördinate not an integer");
76          }
77          try {
78              startRow = Integer.parseInt(parts[5]);
79          }
80          catch (NumberFormatException e) {
81              throw new PatternFormatException(
82                  "y coördinate not an integer");
83          }
84          cells = parts[6];
85      }
86
87      public void initialise(boolean[][] world) throws PatternFormatException {
88          String[] rows = cells.split(" ");
89          char[][] values = new char[rows.length][];
90          for (int i = 0; i < rows.length; i++)
91              values[i] = rows[i].toCharArray();
92
93          for (int i = 0; i < values.length; i++)
94              for (int j = 0; j < values[i].length; j++)
95                  if (" 01".indexOf(values[i][j]) == -1)
96                      throw new PatternFormatException(
97                          "Pattern contains characters other than ‘0’, ‘1’ and ‘ ’");
98                  else
99                      world[startRow + i][startCol + j] = values[i][j] == '1';
100     }
101
102     @Override
103     public String toString() {
104         return name + ":" + author + ":" + width + ":" + height + ":" +
105             startCol + ":" + startRow + ":" + cells;
106     }
107 }
```

PatternFormatException.java

```
0   package uk.ac.cam.jdw74.tick4star;
1
2   public class PatternFormatException extends Exception {
3       public PatternFormatException(String message) {
4           super(message);
5       }
6   }
```

PatternLife.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  class PatternLife {
 3      public static void print(boolean[][] world) {
 4          System.out.println("-");
 5          for (int row = 0; row < world.length; row++) {
 6              for (int col = 0; col < world[row].length; col++) {
 7                  System.out.print(getCell(world, col, row) ? "#" : "_");
 8              }
 9              System.out.println();
10          }
11      }
12
13      public static boolean getCell(boolean[][] world, int col, int row) {
14          return 0 <= row && row < world.length &&
15                 0 <= col && col < world[row].length ?
16              world[row][col] : false;
17      }
18
19      public static void setCell(boolean[][] world, int col, int row, boolean value) {
20          if (0 <= row && row < world.length &&
21              0 <= col && col < world[row].length)
22              world[row][col] = value;
23      }
24
25      public static int countNeighbours(boolean[][] world, int col, int row) {
26          return
27              (getCell(world, col - 1, row - 1) ? 1 : 0)
28            + (getCell(world, col    , row - 1) ? 1 : 0)
29            + (getCell(world, col + 1, row - 1) ? 1 : 0)
30            + (getCell(world, col - 1, row    ) ? 1 : 0)
31            + (getCell(world, col + 1, row    ) ? 1 : 0)
32            + (getCell(world, col - 1, row + 1) ? 1 : 0)
33            + (getCell(world, col    , row + 1) ? 1 : 0)
34            + (getCell(world, col + 1, row + 1) ? 1 : 0);
35      }
36
37      public static boolean computeCell(boolean[][] world, int col, int row) {
38          int count = countNeighbours(world, col, row);
39          return count == 3 || (getCell(world, col, row) && count == 2);
40      }
41
42      public static boolean[][] nextGeneration(boolean[][] world) {
43          boolean[][] nextWorld = new boolean[world.length][world[0].length];
44          for (int row = 0; row < world.length; row++)
45              for (int col = 0; col < world[row].length; col++)
46                  setCell(nextWorld, col, row,
47                          computeCell(world, col, row));
48          return nextWorld;
49      }
50
51      public static void play(boolean[][] world) throws Exception {
52          int userResponse = 0;
53          while (userResponse != 'q') {
54              print(world);
55              userResponse = System.in.read();
56              world = nextGeneration(world);
57          }
58      }
59
60      public static void main(String[] args) throws Exception {
61          try {
62              Pattern p = new Pattern(args[0]);
63              boolean[][] world = new boolean[p.getHeight()][p.getWidth()];
64              p.initialise(world);
65              play(world);
66          }
67          catch (PatternFormatException e) {
68              System.out.println(e.getMessage());
69          }
70          catch (ArrayIndexOutOfBoundsException e) {
71              System.out.println("No argument given");
```

```
72          }
73      }
74  }
```

PatternLoader.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  import java.io.Reader;
 3  import java.io.BufferedReader;
 4  import java.io.IOException;
 5  import java.io.InputStreamReader;
 6  import java.io.FileReader;
 7  import java.util.List;
 8  import java.util.LinkedList;
 9  import java.net.URL;
10  import java.net.URLConnection;
11
12  public class PatternLoader {
13
14      public static List<Pattern> load(Reader r) throws IOException {
15          BufferedReader buff = new BufferedReader(r);
16          List<Pattern> results = new LinkedList<>();
17
18          String line;
19          while ((line = buff.readLine()) != null)
20              try {
21                  results.add(new Pattern(line));
22              }
23              catch (PatternFormatException e) {}
24
25          return results;
26      }
27
28      public static List<Pattern> loadFromURL(String url) throws IOException {
29          URL destination = new URL(url);
30          URLConnection conn = destination.openConnection();
31          return load(new InputStreamReader(conn.getInputStream()));
32      }
33
34      public static List<Pattern> loadFromDisk(String filename)
35          throws IOException {
36          return load(new FileReader(filename));
37      }
38  }
```

Repeat.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  public class Repeat {
 3      public static void main(String[] args) {
 4          System.out.println(parseAndRep(args));
 5      }
 6
 7      public static String parseAndRep(String[] args) {
 8          if (args.length < 2)
 9              return "Error: insufficient arguments";
10          int n;
11          try {
12              n = Integer.parseInt(args[1]);
13              if (n < 1) throw new NumberFormatException();
14          } catch (NumberFormatException ex) {
15              return "Error: second argument is not a positive integer";
16          }
17
18          int i = n;
19          String r = "";
20          while (i > 1) {
21              r += args[0] + " ";
22              i--;
23          }
24          return r + args[0];
25      }
26  }
```

Statistics.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  import java.util.List;
 3  import java.util.ArrayList;
 4
 5  public class Statistics {
 6      private String mName;
 7      private boolean[][] mWorld;
 8      private double mMaximumGrowthRate;
 9      private double mMaximumDeathRate;
10      private int mLoopStart;
11      private int mLoopEnd;
12      private int mMaximumPopulation;
13
14      public String getName() {
15          return mName;
16      }
17
18      private static int getPopulation(boolean[][] w) {
19          int r = 0;
20          for (boolean[] a : w)
21              for (boolean c : a)
22                  if (c) r++;
23          return r;
24      }
25
26      private static boolean[][] copyWorld(boolean[][] w) {
27          boolean[][] copy = new boolean[w.length][w[0].length];
28          for (int i = 0; i < w.length; i++)
29              for (int j = 0; j < w[i].length; j++)
30                  copy[i][j] = w[i][j];
31          return copy;
32      }
33
34      public double getMaximumGrowthRate() {
35          return mMaximumGrowthRate;
36      }
37
38      public double getMaximumDeathRate() {
39          return mMaximumDeathRate;
40      }
41
42      public int getLoopStart() {
43          return mLoopStart;
44      }
45
46      public int getLoopEnd() {
47          return mLoopEnd;
48      }
49
50      public int getMaximumPopulation() {
51          return mMaximumPopulation;
52      }
53
54      private static boolean worldsEqual(boolean[][] u, boolean[][] v) {
55          for (int i = 0; i < u.length; i++)
56              for (int j = 0; j < u[i].length; j++)
57                  if (u[i][j] != v[i][j])
58                      return false;
59          return true;
60      }
61
62      public Statistics(boolean[][] w, String name) {
63          mWorld = w;
64          mName = name;
65
66          mMaximumGrowthRate = 0.0;
67          mMaximumDeathRate = 0.0;
68          mMaximumPopulation = getPopulation(w);
69
70          List<boolean[][]> history = new ArrayList<>();
71          history.add(w);
```

```
72          int population = mMaximumPopulation;
73          int generation = 0;
74          boolean notFoundLoop = true;
75          while (notFoundLoop) {
76              double dLastPopulation = (double)population;
77              history.add(StatisticsLife.nextGeneration(history.get(generation)));
78              generation++;
79
80              population = getPopulation(history.get(generation));
81              double growthRate = ((double)population - dLastPopulation) /
82                  dLastPopulation;
83              double deathRate = -growthRate;
84
85              if (population > mMaximumPopulation)
86                  mMaximumPopulation = population;
87              if (growthRate > mMaximumGrowthRate)
88                  mMaximumGrowthRate = growthRate;
89              if (deathRate > mMaximumDeathRate)
90                  mMaximumDeathRate = deathRate;
91
92              for (int i = 0; i < generation; i++) {
93                  if (worldsEqual(history.get(i), history.get(generation))) {
94                      mLoopStart = i;
95                      mLoopEnd = generation - 1;
96                      notFoundLoop = false;
97                  }
98              }
99          }
100     }
101 }
```

StatisticsLife.java

```
 0  package uk.ac.cam.jdw74.tick4star;
 1
 2  import java.util.List;
 3  import java.util.ArrayList;
 4
 5  class StatisticsLife {
 6      public static void print(boolean[][] world) {
 7          System.out.println("-");
 8          for (int row = 0; row < world.length; row++) {
 9              for (int col = 0; col < world[row].length; col++) {
10                  System.out.print(getCell(world, col, row) ? "#" : "_");
11              }
12              System.out.println();
13          }
14      }
15
16      public static boolean getCell(boolean[][] world, int col, int row) {
17          return 0 <= row && row < world.length &&
18                 0 <= col && col < world[row].length ?
19              world[row][col] : false;
20      }
21
22      public static void setCell(boolean[][] world, int col, int row, boolean value) {
23          if (0 <= row && row < world.length &&
24              0 <= col && col < world[row].length)
25              world[row][col] = value;
26      }
27
28      public static int countNeighbours(boolean[][] world, int col, int row) {
29          return
30              (getCell(world, col - 1, row - 1) ? 1 : 0)
31            + (getCell(world, col    , row - 1) ? 1 : 0)
32            + (getCell(world, col + 1, row - 1) ? 1 : 0)
33            + (getCell(world, col - 1, row    ) ? 1 : 0)
34            + (getCell(world, col + 1, row    ) ? 1 : 0)
35            + (getCell(world, col - 1, row + 1) ? 1 : 0)
36            + (getCell(world, col    , row + 1) ? 1 : 0)
37            + (getCell(world, col + 1, row + 1) ? 1 : 0);
38      }
39
40      public static boolean computeCell(boolean[][] world, int col, int row) {
41          int count = countNeighbours(world, col, row);
42          return count == 3 || (getCell(world, col, row) && count == 2);
43      }
44
45      public static boolean[][] nextGeneration(boolean[][] world) {
46          boolean[][] nextWorld = new boolean[world.length][world[0].length];
47          for (int row = 0; row < world.length; row++)
48              for (int col = 0; col < world[row].length; col++)
49                  setCell(nextWorld, col, row,
50                          computeCell(world, col, row));
51          return nextWorld;
52      }
53
54      public static void play(boolean[][] world) throws Exception {
55          int userResponse = 0;
56          while (userResponse != 'q') {
57              print(world);
58              userResponse = System.in.read();
59              world = nextGeneration(world);
60          }
61      }
62
63      public static Statistics analyse(Pattern p) throws PatternFormatException {
64          boolean[][] world = new boolean[p.getHeight()][p.getWidth()];
65          p.initialise(world);
66          return new Statistics(world, p.getName());
67      }
68
69      public static void main(String[] args) throws Exception {
70          if (args.length == 0) {
71              System.out.println("No argument given");
```

```
72              return;
73          }
74
75          List<Pattern> ps;
76          String path = args[0];
77          if (path.contains("://"))
78              ps = PatternLoader.loadFromURL(path);
79          else
80              ps = PatternLoader.loadFromDisk(path);
81
82          if (args.length == 1) {
83              List<Statistics> stats = new ArrayList<>();
84              for (Pattern p : ps) {
85                  System.out.println("Analysing " + p.getName());
86                  stats.add(analyse(p));
87              }
88
89              System.out.println();
90
91              int longestStart = 0;
92              String nameWithLongestStart = "";
93              int longestCycle = 0;
94              String nameWithLongestCycle = "";
95              double biggestGrowthRate = 0.0;
96              String nameWithBiggestGrowthRate = "";
97              double biggestDeathRate = 0.0;
98              String nameWithBiggestDeathRate = "";
99              int largestPopulation = 0;
100             String nameWithLargestPopulation = "";
101             for (Statistics s : stats) {
102                 String name = s.getName();
103                 int start = s.getLoopStart();
104                 int cycle = s.getLoopEnd() - start;
105                 double growthRate = s.getMaximumGrowthRate();
106                 double deathRate = s.getMaximumDeathRate();
107                 int population = s.getMaximumPopulation();
108                 if (start > longestStart) {
109                     longestStart = start;
110                     nameWithLongestStart = name;
111                 }
112                 if (cycle > longestCycle) {
113                     longestCycle = cycle;
114                     nameWithLongestCycle = name;
115                 }
116                 if (growthRate > biggestGrowthRate) {
117                     biggestGrowthRate = growthRate;
118                     nameWithBiggestGrowthRate = name;
119                 }
120                 if (deathRate > biggestDeathRate) {
121                     biggestDeathRate = deathRate;
122                     nameWithBiggestDeathRate = name;
123                 }
124                 if (population > largestPopulation) {
125                     largestPopulation = population;
126                     nameWithLargestPopulation = name;
127                 }
128             }
129
130             System.out.println("Longest start: " + nameWithLongestStart + " (" +
131                                Integer.toString(longestStart) + ")");
132             System.out.println("Longest cycle: " + nameWithLongestCycle + " (" +
133                                Integer.toString(longestCycle) + ")");
134             System.out.println("Biggest growth rate: " +
135                                nameWithBiggestGrowthRate + " (" +
136                                Double.toString(biggestGrowthRate) + ")");
137             System.out.println("Biggest death rate: " +
138                                nameWithBiggestDeathRate + " (" +
139                                Double.toString(biggestDeathRate) + ")");
140             System.out.println("Largest population: " +
141                                nameWithLargestPopulation + " (" +
142                                Integer.toString(largestPopulation) + ")");
143         }
```

```
144           else if (args.length == 2)
145               try {
146                   Pattern p = ps.get(Integer.parseInt(args[1]));
147                   boolean[][] world = new boolean[p.getHeight()][p.getWidth()];
148                   p.initialise(world);
149                   play(world);
150               }
151               catch (IndexOutOfBoundsException | NumberFormatException e) {
152                   System.out.println("Second argument is not a valid index");
153               }
154               catch (PatternFormatException e) {
155                   System.out.println("Malformed pattern");
156               }
157           else {
158               System.out.println("Too many arguments");
159               return;
160           }
161       }
162   }
```

output.txt

```
  0   Analysing Glider
  1   Analysing Blinkers
  2   Analysing Octogon
  3   Analysing Block+Boat+Beehive
  4   Analysing The Phoenix
  5   Analysing Glider Gun
  6   Analysing Pi-heptomino
  7   Analysing 101
  8   Analysing 1-2-3
  9   Analysing 1-2-3-4
 10   Analysing 4-8-12 diamond
 11   Analysing 4 boats
 12   Analysing Achim's p144
 13   Analysing Achim's p16
 14   Analysing Achim's p4
 15   Analysing Achim's p8
 16   Analysing acorn
 17   Analysing A for all
 18   Analysing aircraft carrier
 19   Analysing airforce
 20   Analysing AK47 reaction
 21   Analysing almosymmetric
 22   Analysing ants
 23   Analysing anvil
 24   Analysing ark
 25   Analysing aVerage
 26   Analysing B29
 27   Analysing B-52 bomber
 28   Analysing babbling brook
 29   Analysing backrake
 30   Analysing baker
 31   Analysing baker's dozen
 32   Analysing bakery
 33   Analysing barge
 34   Analysing beacon
 35   Analysing beacon maker
 36   Analysing beehive
 37   Analysing beehive and dock
 38   Analysing beehive with tail
 39   Analysing bent keys
 40   Analysing B-heptomino
 41   Analysing bi-block
 42   Analysing biclock
 43   Analysing big glider
 44   Analysing big S
 45   Analysing bi-loaf
 46   Analysing bipole
 47   Analysing bi-pond
 48   Analysing biting off more than they can chew
 49   Analysing blinker
 50   Analysing blinker puffer
 51   Analysing blinkers bit pole
 52   Analysing blinker ship
 53   Analysing block
 54   Analysing blockade
 55   Analysing block and dock
 56   Analysing block and glider
 57   Analysing blocker
 58   Analysing block on table
 59   Analysing block pusher
 60   Analysing blom
 61   Analysing blonker
 62   Analysing boat
 63   Analysing boat-bit
 64   Analysing boat maker
 65   Analysing boat-tie
 66   Analysing boojum reflector
 67   Analysing bookend
 68   Analysing bookends
 69   Analysing boss
 70   Analysing bottle
 71   Analysing brain
```

```
 72   Analysing buckaroo
 73   Analysing bullet heptomino
 74   Analysing bun
 75   Analysing bunnies
 76   Analysing burloaferimeter
 77   Analysing butterfly
 78   Analysing by flops
 79   Analysing Canada goose
 80   Analysing candelabra
 81   Analysing candlefrobra
 82   Analysing canoe
 83   Analysing cap
 84   Analysing carnival shuttle
 85   Analysing caterer
 86   Analysing Caterpillar
 87   Analysing cauldron
 88   Analysing centinal
 89   Analysing century
 90   Analysing chemist
 91   Analysing C-heptomino
 92   Analysing Cheshire cat
 93   Analysing chicken wire
 94   Analysing cis-beacon on anvil
 95   Analysing cis-beacon on table
 96   Analysing cis-boat with tail
 97   Analysing cis fuse with two tails
 98   Analysing cis-mirrored R-bee
 99   Analysing clock
100   Analysing clock II
101   Analysing Coe ship
102   Analysing Coe's p8
103   Analysing colour of a glider
104   Analysing conduit
105   Analysing confused eaters
106   Analysing converter
107   Analysing Cordership
108   Analysing cousins
109   Analysing cover
110   Analysing cow
111   Analysing crane
112   Analysing cross
113   Analysing crowd
114   Analysing crown
115   Analysing crystal
116   Analysing cuphook
117   Analysing dart
118   Analysing dead spark coil
119   Analysing diamond ring
120   Analysing diehard
121   Analysing dinner table
122   Analysing diuresis
123   Analysing dock
124   Analysing do-see-do
125   Analysing double block reaction
126   Analysing double caterer
127   Analysing double ewe
128   Analysing dove
129   Analysing dragon
130   Analysing eater
131   Analysing eater1
132   Analysing eater2
133   Analysing eater3
134   Analysing eater4
135   Analysing eater/block frob
136   Analysing eater plug
137   Analysing ecologist
138   Analysing edge-repair spaceship
139   Analysing edge shooter
140   Analysing E-heptomino
141   Analysing electric fence
142   Analysing elevener
143   Analysing Elkies' p5
```

```
144   Analysing en retard
145   Analysing Enterprise
146   Analysing Eureka
147   Analysing extremely impressive
148   Analysing factory
149   Analysing Fast Forward Force Field
150   Analysing F-heptomino
151   Analysing figure-8
152   Analysing filter
153   Analysing fire-spitting
154   Analysing fleet
155   Analysing flotilla
156   Analysing fly
157   Analysing fore and back
158   Analysing fountain
159   Analysing fourteener
160   Analysing fox
161   Analysing French kiss
162   Analysing frog II
163   Analysing frothing puffer
164   Analysing fumarole
165   Analysing Gabriel's p138
166   Analysing Garden of Eden
167   Analysing germ
168   Analysing glasses
169   Analysing glider
170   Analysing glider-block cycle
171   Analysing glider duplicator
172   Analysing gliderless
173   Analysing glider pusher
174   Analysing gliders by the dozen
175   Analysing glider synthesis
176   Analysing glider turner
177   Analysing Gosper glider gun
178   Analysing gourmet
179   Analysing Gray counter
180   Analysing great on-off
181   Analysing grin
182   Analysing grow-by-one object
183   Analysing hammer
184   Analysing hammerhead
185   Analysing handshake
186   Analysing harbor
187   Analysing harvester
188   Analysing hat
189   Analysing hebdarole
190   Analysing hectic
191   Analysing Heisenburp device
192   Analysing helix
193   Analysing heptapole
194   Analysing Herschel
195   Analysing Herschel receiver
196   Analysing Herschel transmitter
197   Analysing Hertz oscillator
198   Analysing hexapole
199   Analysing H-heptomino
200   Analysing hivenudger
201   Analysing honeycomb
202   Analysing honey farm
203   Analysing hook with tail
204   Analysing houndstooth agar
205   Analysing house
206   Analysing hustler
207   Analysing hustler II
208   Analysing HW emulator
209   Analysing HWSS
210   Analysing HW volcano
211   Analysing I-heptomino
212   Analysing infinite growth
213   Analysing inline inverter
214   Analysing integral sign
215   Analysing interchange
```

```
216   Analysing Iwona
217   Analysing jack
218   Analysing jam
219   Analysing Jolson
220   Analysing Justyna
221   Analysing kickback reaction
222   Analysing killer toads
223   Analysing Kok's galaxy
224   Analysing lake
225   Analysing Laputa
226   Analysing Lidka
227   Analysing light bulb
228   Analysing lightspeed wire
229   Analysing loading dock
230   Analysing loaf
231   Analysing loaflipflop
232   Analysing loaf siamese barge
233   Analysing long barge
234   Analysing long boat
235   Analysing long bookend
236   Analysing long canoe
237   Analysing long integral
238   Analysing long long barge
239   Analysing long long boat
240   Analysing long long canoe
241   Analysing long long ship
242   Analysing long long snake
243   Analysing long shillelagh
244   Analysing long ship
245   Analysing long snake
246   Analysing loop
247   Analysing LW emulator
248   Analysing LWSS
249   Analysing mango
250   Analysing mathematician
251   Analysing mazing
252   Analysing metamorphosis
253   Analysing metamorphosis II
254   Analysing Mickey Mouse
255   Analysing mini pressure cooker
256   Analysing mold
257   Analysing monogram
258   Analysing moose antlers
259   Analysing multum in parvo
260   Analysing muttering moat
261   Analysing MW emulator
262   Analysing MWSS
263   Analysing MWSS out of the blue
264   Analysing MW volcano
265   Analysing negentropy
266   Analysing new five
267   Analysing new gun
268   Analysing Noah's ark
269   Analysing non-monotonic
270   Analysing octagon II
271   Analysing octagon IV
272   Analysing octomino
273   Analysing odd keys
274   Analysing onion rings
275   Analysing O-pentomino
276   Analysing Orion
277   Analysing p54 shuttle
278   Analysing p6 shuttle
279   Analysing paperclip
280   Analysing pedestle
281   Analysing penny lane
282   Analysing pentadecathlon
283   Analysing pentant
284   Analysing pentapole
285   Analysing pentoad
286   Analysing phase change
287   Analysing phi
```

```
Analysing phoenix
Analysing pi-heptomino
Analysing pinwheel
Analysing pi orbital
Analysing pi portraitor
Analysing pipsquirter
Analysing pi ship
Analysing piston
Analysing pi wave
Analysing pond
Analysing pond on pond
Analysing popover
Analysing P-pentomino
Analysing pre-beehive
Analysing pre-block
Analysing pre-pulsar
Analysing pressure cooker
Analysing protein
Analysing pseudo-barberpole
Analysing puffer
Analysing puffer train
Analysing puff suppressor
Analysing pulsar
Analysing pulsar quadrant
Analysing pulse divider
Analysing pulshuttle V
Analysing pure glider generator
Analysing pushalong
Analysing pyrotechnecium
Analysing Q-pentomino
Analysing quad
Analysing quadpole
Analysing quarter
Analysing quasar
Analysing queen bee shuttle
Analysing R2D2
Analysing rabbits
Analysing $rats
Analysing reflector
Analysing relay
Analysing rephaser
Analysing reverse fuse
Analysing revolver
Analysing ring of fire
Analysing roteightor
Analysing R-pentomino
Analysing rumbling river
Analysing sailboat
Analysing Schick engine
Analysing scorpion
Analysing scrubber
Analysing seal
Analysing sesquihat
Analysing shillelagh
Analysing ship
Analysing ship in a bottle
Analysing ship-tie
Analysing ship tie boat
Analysing short keys
Analysing side
Analysing sidecar
Analysing sidewalk
Analysing siesta
Analysing Silver's p5
Analysing singular flip flop
Analysing six Ls
Analysing sixty-nine
Analysing skewed quad
Analysing skewed traffic light
Analysing slide gun
Analysing small lake
Analysing smiley
```

```
360   Analysing snacker
361   Analysing snail
362   Analysing snake
363   Analysing snake bridge snake
364   Analysing snake dance
365   Analysing snake pit
366   Analysing snake siamese snake
367   Analysing sombreros
368   Analysing spacefiller
369   Analysing space rake
370   Analysing spaceship
371   Analysing spark coil
372   Analysing sparky
373   Analysing S-pentomino
374   Analysing spider
375   Analysing spiral
376   Analysing squaredance
377   Analysing stairstep hexomino
378   Analysing star
379   Analysing stillater
380   Analysing still life
381   Analysing still life tagalong
382   Analysing superfountain
383   Analysing superstring
384   Analysing surprise
385   Analysing swan
386   Analysing switch engine
387   Analysing table
388   Analysing table on table
389   Analysing teardrop
390   Analysing technician
391   Analysing test tube baby
392   Analysing tetromino
393   Analysing thumb
394   Analysing thunderbird
395   Analysing time bomb
396   Analysing titanic toroidal traveler
397   Analysing T-nosed p4
398   Analysing T-nosed p6
399   Analysing toad
400   Analysing toad-flipper
401   Analysing toad-sucker
402   Analysing toaster
403   Analysing total aperiodic
404   Analysing T-pentomino
405   Analysing tractor beam
406   Analysing traffic circle
407   Analysing traffic jam
408   Analysing traffic light
409   Analysing trans-beacon on table
410   Analysing trans-boat with tail
411   Analysing trans-loaf with tail
412   Analysing transparent block reaction
413   Analysing trice tongs
414   Analysing triple caterer
415   Analysing triplet
416   Analysing tripole
417   Analysing tritoad
418   Analysing true
419   Analysing T-tetromino
420   Analysing tub
421   Analysing tubber
422   Analysing tubeater
423   Analysing tubstretcher
424   Analysing tub with tail
425   Analysing tumbler
426   Analysing tumbling T-tetson
427   Analysing turning toads
428   Analysing turtle
429   Analysing twin bees shuttle
430   Analysing twinhat
431   Analysing twirling T-tetsons II
```

```
Analysing two eaters
Analysing two pulsar quadrants
Analysing unix
Analysing U-pentomino
Analysing very long house
Analysing V-pentomino
Analysing washerwoman
Analysing washing machine
Analysing wasp
Analysing wavefront
Analysing weekender
Analysing why not
Analysing wickstretcher
Analysing windmill
Analysing wing
Analysing worker bee
Analysing W-pentomino
Analysing x66
Analysing X-pentomino
Analysing Y-pentomino
Analysing Z-hexomino
Analysing Z-pentomino

Longest start: gliderless (1304)
Longest cycle: factory (287)
Biggest growth rate: glider synthesis (1.2222222222222223)
Biggest death rate: brain (1.0)
Largest population: spacefiller (849)
```