Main Program

1. Draw colour-map
2. Circumscribe clusters and eyes
3. Identify and mark obviously dead stones
4. Redraw colour-map
5. Compute and circumscribe cluster shadows
6. Perform local life-and-death dynamic analysis
7. Redraw colour-map
8. Reompute cluster shadows
9. Compose colour-map and shadows board graphics

Draw colour-map

```
Until no new coloured points or links are discovered, Repeat:
    1. a newly-coloured point colours its links;
    2. an uncoloured empty point [edge point],
        at least 3 [2] of whose links are same-coloured
            and none opposite-coloured,
       is coloured;
    3. an uncoloured link connecting 2 uncoloured points,
        each of which has at least one coloured link
            and no opposite-colored links,
       is coloured;
if a link becomes coloured by both colours,
    its colour is neutralised.
```

Circumscribe clusters and eyes

```
clusters.numberof := 0;
for point in b do
    if all-links(point) are same-colour or neutral then
    for each coloured-link(point) do
        if member(otherpoint(link, point), cluster)
        # ie the point at the other end of the link
        then add(point, cluster)
        else makenewcluster(point)
    makenewcluster(point) =
    clusters.numberof +:= 1;
    let newcluster = [{point}, clusters.numberof]
    paint(board.point, point.newcluster.number, point.colour(point)
```

Identify and mark obviously dead stones

```
foreach cluster in clusters do
        identify(cluster.eyes);
        if number(cluster.eyes) > 2
                or size(cluster.eyes) > 3
                and shape(cluster.eye) not(in{dead-shapes})
        then cluster.lad := alive
        elsif surrounded(cluster, enemies)
                and foreach enemy in enemies (enemy.lad = alive)
        then cluster.lad := dead

identify(cluster.eyes) =
        foreach point in cluster do
                if colour-controlled(point) and
                        not border(point) or stone(point)
                then append(point, cluster.eyes)

surrounded (cluster, enemies) =
        not(forany point in border(cluster)
                path(friend(point)
                or path(openspace, point))
```

Compute and circumscribe cluster shadows

```
iboard := board;
foreach cluster in board do
        foreach point in cluster do
        if point.coloured then iboard.point := pretendstone (colour);
isboard:= Influencie (iboard);
foreach point in board do point.shadow:= isboard.point.shadow;

foreach cluster in board do
        circumscribe cluster.shadow
```

Perform dynamic local life-and-death analysis

```
foreach cluster in board do
        foreach point in cluster or cluster.shadow do
                zboard.point:= board.point;
        fillup rest of zboard with black stones;
        poke 2 eyes in rest of zboard;
        Laizy(zboard);
        foreach point in cluster do
                if board.point.occupant = enemystone and not(zboard.point.occupant = enemystone)
                then board.point.occupant:= deadenemystone
```