```
program orb;

type
     spin_dir        = (pos,  neg);
     rotation        = (left, right);
     ring            = array [1..20] of char;
     moonlist        = array [spin_dir] of ring;
     move            = record
                         source  : integer;
                         dest    : integer;
                         dir     : rotation;
                         spin    : spin_dir
                       end;

var  moons                           : moonlist;
     planets, orbits, spins          : integer;
     interval, guarded_planet        : integer;
     random_seed, num_moves          : integer;
     player_move, crit_move          : move;
     all_done                        : boolean;
     i                               : integer;

(*==============================================================*)

function rand (lo, hi : integer) : integer;
begin
     random_seed := (random_seed * 211) mod 1021;
     rand := random_seed mod (hi - lo + 1) + lo
end;

(*==============================================================*)

procedure initialize (var r : ring);
     var i, x : integer;
begin
     for i := 1 to planets * orbits do r[i] := 'b';
     for i := 1 to (planets-1) * orbits do
     begin
          repeat
               x := rand(1, planets * orbits)
          until r[x] = 'b';
          case ((i-1) div orbits + 1) of
               1 : r[x] := 'r';
               2 : r[x] := 'y';
               3 : r[x] := 'g';
          end
     end
end; (* initialize *)

(*==============================================================*)

procedure setparms;
begin
     writeln(tty); writeln(tty);
     repeat
          writeln(tty,'How many planets (3 or 4)?');
          readln(tty,planets)
     until (planets = 3) or (planets = 4);

     repeat
          writeln(tty,'How many orbits per planet (2 - 5)?');
          readln(tty,orbits)
     until (orbits >= 2) and (orbits <= 5);
```

```
program orb;

type
     spin_dir        = (pos,  neg);
     rotation        = (left, right);
     ring            = array [1..20] of char;
     moonlist        = array [spin_dir] of ring;
     move            = record
                          source  : integer;
                          dest    : integer;
                          dir     : rotation;
                          spin    : spin_dir
                       end;

var  moons                          : moonlist;
     planets, orbits, spins         : integer;
     interval, guarded_planet       : integer;
     random_seed, num_moves         : integer;
     player_move, crit_move         : move;
     all_done                       : boolean;
     i                              : integer;

(*============================================================*)

function rand (lo, hi : integer) : integer;
begin
     random_seed := (random_seed * 211) mod 1021;
     rand := random_seed mod (hi - lo + 1) + lo
end;

(*============================================================*)

procedure initialize (var r : ring);
     var i, x : integer;
begin
     for i := 1 to planets * orbits do r[i] := 'b';
     for i := 1 to (planets-1) * orbits do
     begin
          repeat
               x := rand(1, planets * orbits)
          until r[x] = 'b';
          case ((i-1) div orbits + 1) of
               1 : r[x] := 'r';
               2 : r[x] := 'y';
               3 : r[x] := 'g';
          end
     end
end; (* initialize *)

(*============================================================*)

procedure setparms;
begin
     writeln(tty); writeln(tty);
     repeat
          writeln(tty,'How many planets (3 or 4)?');
          readln(tty,planets)
     until (planets = 3) or (planets = 4);

     repeat
          writeln(tty,'How many orbits per planet (2 - 5)?');
          readln(tty,orbits)
     until (orbits >= 2) and (orbits <= 5);
     repeat
          writeln(tty,'1 or 2 moons per orbit?');
          readln(tty,spins)
     until (spins = 1) or (spins = 2)
end; (* setparms *)

(*==============================================================*)

procedure setmove;
    var direction : char;
begin
     with player_move do
     begin
          repeat
               write(tty,'Enter Source, Dest, and Rotation ');
               writeln(tty,'- eg "3 1R" or "-2 4L"');
               read(tty,source);
               if source < 0 then
                    begin spin := neg; source := -source end
                    else  spin := pos;
               read(tty,dest);
               readln(tty,direction)
          until (source > 0) and (source <= planets)
               and (dest > 0) and (dest <= planets)
               and ((direction = 'r') or (direction = 'R')
                 or (direction = 'l') or (direction = 'L'));

          if spins = 1 then spin := pos;
          if (direction = 'l') or (direction = 'L')
               then dir := left
               else dir := right;
          source := source * orbits;
          dest   := dest   * orbits
     end (* with *)
end; (* setmove *)

(*==============================================================*)

function  solved (moons : ring) : boolean;
    var result : boolean;
        i, j   : integer;
begin
    (* check that all groups of moons have the same color. *)
    (* if the first three groups are ok, the fourth is too *)
    result := true;
    for i := 0 to planets-2 do
        for j := 1 to orbits-1 do
            result := result and
                   (moons[orbits*i+j] = moons[orbits*i+j+1]);
    solved := result
end; (* solved *)

(*==============================================================*)

procedure makemove (selection : move; var moons : ring);
    var temp : ring;
        i, j : integer;
begin
with selection do
begin
    (*----------------------------------------------------------------*)
    (* If Source < Dest and rotating clockwise, Dest is given by: *)
    (*                                                              *)
    (*     M[12] & M[1..S-1] & M[S+1..D-1] & M[S] & M[D..11]        *)
    (*----------------------------------------------------------------*)
    if (dir = right) and (source < dest) then
    begin
        j := 1;
        temp[j] := moons[orbits*planets];
        j := j + 1;
    begin
        j := 1;
        temp[j] := moons[orbits*planets];
        j := j + 1;

        for i := 1 to source-1 do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        for i := source+1 to dest-1 do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        temp[j] := moons[source]; j := j + 1;

        for i := dest to (orbits*planets-1) do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        moons := temp        (* make final assignment of new state *)
    end (* of clockwise move for SOURCE < DEST case *)

    (*-----------------------------------------------------------*)
    (* If Source > Dest and rotating clockwise, Dest is given by: *)
    (*                                                           *)
    (*         M[1..D-1] & M[S] & M[D..S-1] & M[S+1..12]         *)
    (*-----------------------------------------------------------*)
    else if (dir = right) and (source > dest) then
    begin
        j := 1;

        for i := 1 to dest-1 do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        temp[j] := moons[source]; j := j + 1;

        for i := dest to source-1 do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        for i := source+1 to (orbits*planets) do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        moons := temp          (* make final assignment of new state *)
    end (* of clockwise move for SOURCE > DEST case *)

    (*-----------------------------------------------------------*)
    (* If Source < Dest and rotating counterclock, Dest is :     *)
    (*                                                           *)
    (*         M[1..S-1] & M[S+1..D] & M[S] & M[D+1..orbits*4]   *)
    (*-----------------------------------------------------------*)
    else if (dir = left) and (source < dest) then
    begin
        j := 1;

        for i := 1 to source-1 do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        for i := source+1 to dest do
        begin
        end;

        for i := source+1 to dest do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        temp[j] := moons[source]; j := j + 1;

        for i := dest+1 to (orbits*planets) do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        moons := temp          (* make final assignment of new state *)
    end (* of counterclockwise move for SOURCE < DEST case *)

    (*-------------------------------------------------------------*)
    (* If Source > Dest and rotating counterclock, Dest is :       *)
    (*                                                             *)
    (*   M[2..D] & M[S] & M[D+1..S-1] & M[S+1..orbits*4]           *)
    (*-------------------------------------------------------------*)
    else if (dir = left) and (source > dest) then
    begin
        j := 1;

        for i := 2 to dest do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        temp[j] := moons[source]; j := j + 1;

        for i := dest+1 to source-1 do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        for i := source+1 to (orbits*planets) do
        begin
            temp[j] := moons[i];  j := j + 1
        end;

        temp[j] := moons[1];

        moons := temp          (* make final assignment of new state *)
    end (* of counterclockwise move for SOURCE > DEST case *)

end  (* with *)
end; (* procedure makemove *)

(*==================================================================*)

procedure genmove ;
    var direction : integer;
begin
with crit_move do
begin
    interval := rand(orbits,orbits*2);
    repeat
        source := rand(1, planets*orbits);
        dest   := rand(1, planets*orbits)
    until (source-1) div orbits <> (dest-1) div orbits;

    direction := rand(0,1);
    if direction = 0
        then dir := right
        else dir := left;
    if (spins = 1) or (num_moves = 0)
        then spin := pos
        then dir := right
        else dir := left;
    if (spins = 1) or (num_moves = 0)
        then spin := pos
        else spin := player_move.spin
end (* with *)
end; (* genmove *)

(*================================================================*)

procedure adjust (var pos       : integer);
begin
with crit_move do
begin
    if (dir = right)
        then
            if (source < dest) and ((pos < source) or (pos >= dest))
                then pos := (pos mod (orbits * planets)) + 1
            else if (source > dest) and (pos < source) and (pos >= dest)
                then pos := pos + 1;

    if (dir = left)
        then
            if (source < dest) and (pos > source) and (pos <= dest)
                then pos := pos - 1
            else if (source > dest) and ((pos > source) or (pos <= dest))
                then if pos = 1
                        then pos := orbits * planets
                        else pos := pos - 1;
end  (* with *)
end; (* adjust *)

(*================================================================*)

procedure warn ;
begin
    write(tty,'Warning: ');
    if spins = 2 then if crit_move.spin = pos
        then write(tty,'positive ')
        else write(tty,'negative ');
    write(tty,'moon from orbit ');
    write(tty,((crit_move.source-1) mod orbits + 1):1);
    write(tty,' on planet ');
    write(tty,((crit_move.source-1) div orbits + 1):1);
    write(tty,' will become critical in ');
    write(tty,interval:1);
    writeln(tty,' moves')
end; (* warn *)

(*================================================================*)

procedure inform ;
begin
with crit_move do
begin
    write(tty,'Attention: ');
    if spins = 2 then if crit_move.spin = pos
        then write(tty,'positive ')
        else write(tty,'negative ');
    write(tty,'moon from orbit ');
    write(tty,((source-1) mod orbits + 1):1);
    write(tty,', planet ');
    write(tty,((source-1) div orbits + 1):1);
    write(tty,' has moved to orbit ');
    write(tty,((dest-1) mod orbits + 1):1);
    write(tty,', planet ');
    writeln(tty,((dest-1) div orbits + 1):1)
end  (* with *)
end; (* inform *)
```

```
        then dir := right
        else dir := left;
    if (spins = 1) or (num_moves = 0)
        then spin := pos
        else spin := player_move.spin
end (* with *)
end; (* genmove *)

(*============================================================*)

procedure adjust (var pos      : integer);
begin
with crit_move do
begin
    if (dir = right)
        then
            if (source < dest) and ((pos < source) or (pos >= dest))
                then pos := (pos mod (orbits * planets)) + 1
            else if (source > dest) and (pos < source) and (pos >= dest)
                then pos := pos + 1;

    if (dir = left)
        then
            if (source < dest) and (pos > source) and (pos <= dest)
                then pos := pos - 1
            else if (source > dest) and ((pos > source) or (pos <= dest))
                then if pos = 1
                        then pos := orbits * planets
                        else pos := pos - 1;
end  (* with *)
end; (* adjust *)

(*============================================================*)

procedure warn ;
begin
    write(tty,'Warning: ');
    if spins = 2 then if crit_move.spin = pos
        then write(tty,'positive ')
        else write(tty,'negative ');
    write(tty,'moon from orbit ');
    write(tty,((crit_move.source-1) mod orbits + 1):1);
    write(tty,' on planet ');
    write(tty,((crit_move.source-1) div orbits + 1):1);
    write(tty,' will become critical in ');
    write(tty,interval:1);
    writeln(tty,' moves')
end; (* warn *)

(*============================================================*)

procedure inform ;
begin
with crit_move do
begin
    write(tty,'Attention: ');
    if spins = 2 then if crit_move.spin = pos
        then write(tty,'positive ')
        else write(tty,'negative ');
    write(tty,'moon from orbit ');
    write(tty,((source-1) mod orbits + 1):1);
    write(tty,', planet ');
    write(tty,((source-1) div orbits + 1):1);
    write(tty,' has moved to orbit ');
    write(tty,((dest-1) mod orbits + 1):1);
    write(tty,', planet ');
    writeln(tty,((dest-1) div orbits + 1):1)
end  (* with *)
end; (* inform *)
```

```
    writeln(tty,((dest-1) div orbits + 1):1)
end  (* with *)
end; (* inform *)

(*==============================================================*)

procedure tab (n : integer);
    var j : integer;
begin
    for j := 1 to n do write(tty,' ')
end; (* tab *)

(*==============================================================*)

procedure disp_4_1;
    var i : integer;
begin (* display *)

    tab(4 + 3*(orbits-1)); writeln(tty,'1'); tab(2);

    for i := 1 to orbits do
    begin
        tab(2); write(tty,moons[pos][i]:1)
    end;

    writeln(tty); write(tty,' 4');
    write(tty,moons[pos][4*orbits]:1);
    tab(3*orbits); write(tty,moons[pos][orbits+1]:1);

    for i := 1 to orbits-1 do
    begin
        writeln(tty); writeln(tty); tab(2);
        write(tty,moons[pos][4*orbits-i]:1);
        tab(3*orbits); write(tty,moons[pos][orbits+i+1]:1)
    end;

    writeln(tty,'2'); tab(2);

    for i := 1 to orbits do
    begin
        tab(2); write(tty,moons[pos][3*orbits-i+1]:1)
    end;

    writeln(tty); writeln(tty,'    3')

end;  (* disp_4_1 *)

(*==============================================================*)

procedure disp_4_2;
     var i : integer;
begin
     tab(3*orbits+1); writeln(tty,'(1)'); tab(3);
     for i := 1 to orbits do
     begin
          tab(2); write(tty,moons[pos][i]:1)
     end;
     writeln(tty); tab(3);
     for i := 1 to orbits do
     begin
          tab(2); write(tty,moons[neg][i]:1)
     end;
     writeln(tty); write(tty,'(4)');
     write(tty,moons[pos][4*orbits]:1);
     write(tty,moons[neg][4*orbits]:1);
     tab(3*(orbits-1) + 1);
     write(tty,moons[neg][orbits+1]:1);
     write(tty,moons[pos][orbits+1]:1);
     for i := 1 to orbits-1 do
     begin
     tab(3*(orbits-1) + 1);
     write(tty,moons[neg][orbits+1]:1);
     write(tty,moons[pos][orbits+1]:1);
     for i := 1 to orbits-1 do
     begin
          writeln(tty); writeln(tty); tab(3);
          write(tty,moons[pos][4*orbits-i]:1);
          write(tty,moons[neg][4*orbits-i]:1);
          tab(3*(orbits-1)+1);
          write(tty,moons[neg][orbits+i+1]:1);
          write(tty,moons[pos][orbits+i+1]:1)
     end;
     writeln(tty,'(2)'); tab(3);
     for i := 1 to orbits do
     begin
          tab(2); write(tty,moons[neg][3*orbits-i+1]:1)
     end;
     writeln(tty); tab(3);
     for i := 1 to orbits do
     begin
          tab(2); write(tty,moons[pos][3*orbits-i+1]:1)
     end;
     writeln(tty); writeln(tty,'    (3)')
end; (* disp_4_2 *)

(*===============================================================*)

procedure disp_3_1;
     var i : integer;
begin
     tab(2*orbits-3); write(tty,'(3)');
     for i := 1 to orbits do
     begin
          writeln(tty); writeln(tty);
          tab(12-(2*i)); write  (tty,moons[pos][3*orbits-i+1]:1);
          tab(4*i-1);    write  (tty,moons[pos][i]:1)
     end;
     writeln(tty,'  (1)'); writeln(tty); tab(11-2*orbits);
     for i := 2*orbits downto orbits+1 do
     begin
          tab(3); write(tty,moons[pos][i]:1)
     end;
     writeln(tty); writeln(tty);
     tab(13-2*orbits); writeln(tty,'(2)')
end; (* disp_3_1 *)

(*===============================================================*)

procedure disp_3_2;
     var i : integer;
begin
     tab(2*orbits-3); write(tty,'(3)');
     for i := 1 to orbits do
     begin
          writeln(tty);
          tab(11-(2*i)); write  (tty,moons[pos][3*orbits-i+1]:1);
          tab(4*i+1);    writeln(tty,moons[pos][i]:1);
          tab(12-(2*i)); write  (tty,moons[neg][3*orbits-i+1]:1);
          tab(4*i-1);    write  (tty,moons[neg][i]:1)
     end;
     tab(2); writeln(tty,'(1)'); tab(11-2*orbits);
     for i := 2*orbits downto orbits+1 do
     begin
          tab(3); write(tty,moons[neg][i]:1)
     end;
     writeln(tty); tab(11-2*orbits);
     for i := 2*orbits downto orbits+1 do
     begin
          tab(3); write(tty,moons[neg][i]:1)
     end;
     writeln(tty); tab(13-2*orbits); writeln(tty,'(2)')
     for i := 2*orbits downto orbits+1 do
     begin
          tab(3); write(tty,moons[neg][i]:1)
     end;
     writeln(tty); tab(13-2*orbits); writeln(tty,'(2)')
end; (* disp_3_2 *)

(*=================================================================*)

procedure display;
begin
     case spins * planets of
          3 : disp_3_1;
          4 : disp_4_1;
          6 : disp_3_2;
          8 : disp_4_2;
     end
end;

(*=================================================================*)
(* MAIN BODY OF ORB PROGRAM ++++++++++++++++++++++++++++++++++    *)
(*=================================================================*)

begin
     random_seed := time div 1000;
     num_moves := 0;

     setparms;

     initialize(moons[pos]);
     if spins = 2 then initialize(moons[neg]);

     display;
     genmove;

     repeat
         if interval = 0 then
         begin
             inform;
             makemove(crit_move,moons[crit_move.spin]);
             genmove
         end
         else
         begin
             if interval <= 2 then warn;

             setmove;
             makemove(player_move, moons[player_move.spin]);

             if (interval <= 2) and
                  (player_move.source = crit_move.source) then
                genmove
             else begin
                if player_move.spin = crit_move.spin then
                begin
                    adjust(crit_move.source);
                    adjust(crit_move.dest)
                end;
                interval := interval - 1
             end;

             num_moves := num_moves + 1
         end;
         display;

         if spins = 2
              then all_done := solved(moons[pos])
                   and solved(moons[neg])
                   and (moons[pos][1] = moons[neg][1])
              else all_done := solved(moons[pos])
```

this is happening
nothing get modif

```
              then all_done := solved(moons[pos])
                   and solved(moons[neg])
                   and (moons[pos][1] = moons[neg][1])
              else all_done := solved(moons[pos])
     until all_done;

     writeln(tty);
     write(tty,'Congratulations - you solved it in only ');
     write(tty,num_moves:2);  writeln(tty,' moves !!')
end.
@
```