```
algol,n<
Program Pentomino
begin
   comment

   Time: 280782s = 3d 5h 59m 42s

   No buffer:

   Time classic:        428386
   Time turbo:          408163 4.7pct

   Buffer:

   Time classic:        280782
   Time turbo:          251104 10.6pct


   11 solutions
   ;
   integer BOARDX,BOARDY,BOARDX1,BOARDY1,nsolutions;
   Boolean array transformed pieces[1:13,1:8];
   integer array transformedx[1:12,1:8];
   integer array ntransformed[1:12];
   Boolean array used piece[1:12];
   integer ix,iy;
   real procedure clock count;
   code clock count;
   1, 37;
     zl        , grf p-1   ; RF:=clock count; stack[p-1]:=RF;
   e;

   BOARDX := 8;
   BOARDY := 9;

   BOARDX1 := BOARDX-1;
   BOARDY1 := BOARDY-1;

   begin
      Boolean array board[0:BOARDY+4];
      Boolean array mask[0:BOARDY1];
      integer array solution board[0:BOARDY1,0:BOARDX1];

      procedure move up left(itransform);
      value itransform;
      integer itransform;
      begin
         integer i;
         for i:=i while (integer (transformed pieces[13,itransform]∧
            35 0 5 m))=0 do
         transformed pieces[13,itransform] := transformed pieces[13,itransform]
            shift -5;
         for i:=i while (integer (transformed pieces[13,itransform]∧
            15 0 5 1 5 1 5 1 5 1 5 1))=0 do
         transformed pieces[13,itransform] := transformed pieces[13,itransform]
            shift -1;
      end move up left;
      procedure rotate cw(dst, src);
      value dst, src;
      integer dst, src;
      begin
         integer i,j;
         Boolean s;
         s := 40 0;
   for i:=0 step 1 until 4 do
   begin
      for j:=0 step 1 until 4 do
      s := s  ∨ (((transformed pieces[13,src] shift -j×5) ∧
         (40 1 shift i)) shift (4-j-i+i×5))
   end;
   transformed pieces[13,dst] := s;
   move up left(dst)
end rotate cw;
procedure mirror(dst, src);
value dst, src;
integer dst, src;
begin
   integer i;
   transformed pieces[13,dst] := 40 0;
   for i:=0 step 1 until 4 do
   transformed pieces[13,dst] := (transformed pieces[13,dst] shift 5)  ∨
      ((transformed pieces[13,src] shift -i×5) ∧ 35 0 5 m);
   move up left(dst)
end mirror;
Boolean procedure compare pieces(ipiece1, itransform1, ipiece2, itransform2);
value ipiece1, itransform1, ipiece2, itransform2;
integer ipiece1, itransform1, ipiece2, itransform2;
begin
   integer i;
   compare pieces := (integer transformed pieces[ipiece1,itransform1]) =
      (integer transformed pieces[ipiece2,itransform2]);
end compare pieces;
procedure copy piece(dstpiece, dsttransform, srcpiece, srctransform);
value dstpiece, dsttransform, srcpiece, srctransform;
integer dstpiece, dsttransform, srcpiece, srctransform;
begin
   transformed pieces[dstpiece,dsttransform] :=
      transformed pieces[srcpiece,srctransform]
end copy piece;

procedure transform pieces;
begin
   integer i,ipiece,irotate,imirror,itransformed;
   Boolean piece;

   for ipiece:=1 step 1 until 12 do
   begin
      piece := 40 0;
      for i:=0 step 1 until 4 do
      piece := piece  ∨ ((Boolean read integer) shift 5×i);
      transformed pieces[13,1] := piece;
      ntransformed[ipiece] := 0;
      for irotate:=0 step 1 until 3 do
      begin
         copy piece(13,2,13,1);
         for imirror:=0 step 1 until 1 do
         begin
            if imirror=0 then
            copy piece(13,3,13,2)
            else
            mirror(3,2);
            for itransformed:=1 step 1 until ntransformed[ipiece] do
            begin
               if compare pieces(ipiece,itransformed,13,3) then
                  go to duplicate
            end check for duplicate;
            ntransformed[ipiece] := ntransformed[ipiece]+1;
            for i:=0 step 1 until 4 do
```

```
                  begin
                     if transformed pieces[13,3] shift (-i-1) then
                     begin
                        transformedx[ipiece,ntransformed[ipiece]] := i;
                        go to found first bit
                     end
                  end look for first bit in first row;
found first bit:
                  copy piece(ipiece,ntransformed[ipiece],13,3);
duplicate:
               end imirror;
               rotate cw(2,1);
               copy piece(13,1,13,2)
            end irotate
         end ipiece
      end transform pieces;
      procedure create board;
      begin
         integer i,j;
         board[0] := 24 0 4 m 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 m;
         board[1] := 24 0 4 m 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 m;
         board[2] := 24 0 4 m 1 1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 4 m;
         board[3] := 24 0 4 m 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 4 m;
         board[4] := 24 0 4 m 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 4 m;
         board[5] := 24 0 4 m 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 m;
         board[6] := 24 0 4 m 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 4 m;
         board[7] := 24 0 4 m 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 m;
         board[8] := 24 0 4 m 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 m;
         board[9] := 40 m;
         board[10] := 40 m;
         board[11] := 40 m;
         board[12] := 40 m;
         for i := 0 step 1 until BOARDY1 do
         mask[i] := (board[i] shift -4) ∧ 32 0 8 m;
         for i:=0 step 1 until BOARDY1 do
         for j:=0 step 1 until BOARDX1 do
         solution board[i,j] := -1
      end create board;
      procedure find first free;
      begin
next:
         if board[iy] shift -(ix+5) then
         begin
            ix:=ix+1;
            if ix≥BOARDX then
            begin
               ix := 0;
               iy := iy+1
            end next row;
            go to next
         end bit is one
      end find first free;
      Boolean procedure piece fit(ix,iy,ipiece,itransform);
      value ix,iy,ipiece,itransform;
      integer ix,iy,ipiece,itransform;
      begin
         integer i;
         piece fit := true;
         for i:=0 step 1 until 4 do
         begin
            if (integer(board[iy+i]∧
               (((transformed pieces[ipiece,itransform] shift -5×i) ∧
               35 0 5 m)shift (ix+4))))≠0 then
            begin
```

```
               piece fit := false;
               go to not fit
            end
         end for;
not fit:
      end piece fit;
      procedure print piece(ipiece,itransform);
      value ipiece,itransform;
      integer ipiece,itransform;
      begin
         Boolean s;
         integer i,j;
         s:=transformed pieces[ipiece,itransform];
         writecr;
         for i:=0 step 1 until 4 do
         begin
            for j:=0 step 1 until 4 do
            begin
               s:=s shift -1;
               write(≮d≯, if s then 1 else 0)
            end;
            writecr
         end;
         i:=select(17);
         lyn;
         select(i)
      end print piece;
      procedure print board;
      begin
         integer i,j;
         writecr;
         for i:=0 step 1 until BOARDY1 do
         begin
            for j:=0 step 1 until BOARDX1 do
            write(≮d≯, if board[i] shift -(j+5) then 1 else 0);
            writecr
         end row;
         lyn
      end print board;
      procedure set piece(ix,iy,ipiece,itransform);
      value ix,iy,ipiece,itransform;
      integer ix,iy,ipiece,itransform;
      begin
         integer i;
         for i:=0 step 1 until 4 do
         board[iy+i] := board[iy+i]  ∨
            (((transformed pieces[ipiece,itransform] shift -5×i)
            ∧ 35 0 5 m) shift (ix+4))
      end set piece;
      procedure remove piece(ix,iy,ipiece,itransform);
      value ix,iy,ipiece,itransform;
      integer ix,iy,ipiece,itransform;
      begin
         integer i;
         for i:=0 step 1 until 4 do
         board[iy+i] := board[iy+i] ∧
            -,(((transformed pieces[ipiece,itransform] shift -5×i)
            ∧ 35 0 5 m) shift (ix+4))
      end remove piece;
      procedure set solution(ix,iy,ipiece,itransform);
      value ix,iy,ipiece,itransform;
      integer ix,iy,ipiece,itransform;
      begin
         integer i,j;
```

```
   for i:=0 step 1 until 4 do
   for j:=0 step 1 until 4 do
   begin
      if transformed pieces[ipiece,itransform] shift -(j+1+5×i) then
      solution board[iy+i,ix+j] := ipiece
   end
end set solution;
procedure print solution;
begin
   integer i,j,k;
   writecr;
   write text(≮<Solution: ≯);
   write(≮dddd≯, nsolutions);
   writecr;
   writetext(≮<+---≯);
   for j:=1 step 1 until BOARDX1 do
   begin
      if mask[0] shift -j-1 then
      writetext(≮<XXXX≯)
      else if solution board[0,j-1]=
         solution board[0,j] then
      writetext(≮<----≯)
      else
      writetext(≮<+---≯)
   end first row;
   if mask[0] shift -BOARDX1-1 then
   writetext(≮<X≯)
   else
   writetext(≮<+≯);
   writecr;
   for i:=0 step 1 until BOARDY1 do
   begin
      for k:=1 step 1 until 2 do
      begin
         writetext(≮<I   ≯);
         for j:=1 step 1 until BOARDX1 do
         begin
            if mask[i] shift -j-1 then
            writetext(≮<XXXX≯)
            else if solution board[i,j-1]=
               solution board[i,j] then
            writetext(≮<    ≯)
            else
            if mask[i] shift -j then
            writetext(≮<X   ≯)
            else
            writetext(≮<I   ≯)
         end;
         if mask[i] shift -BOARDX1-1 then
         writetext(≮<X≯)
         else
         writetext(≮<I≯);
         writecr
      end;
      if i<BOARDY1 then
      begin
         if solution board[i,0]=
            solution board[i+1,0] then
         writetext(≮<I   ≯)
         else
         writetext(≮<+---≯);
         for j:=1 step 1 until BOARDX1 do
         begin
            if (mask[i] shift -j-1)  ∨
```

```
              (mask[i+1] shift -j-1) then
         writetext(≮<XXXX≯)
         else if solution board[i,j]=
            solution board[i+1,j] then
         begin
            if solution board[i,j-1]=
               solution board[i+1,j-1] then
            begin
               if solution board[i,j] ≠
                  solution board[i,j-1]  ∨
                  solution board[i+1,j] ≠
                  solution board[i+1,j-1] then
               begin
                  if mask[i] shift -j then
                  writetext(≮<X   ≯)
                  else
                  writetext(≮<I   ≯)
               end
               else
               writetext(≮<    ≯)
            end
            else
            if (mask[i] shift -j)  ∨
                 (mask[i+1] shift -j) then
            writetext(≮<X   ≯)
            else
            writetext(≮<+   ≯)
         end
         else
         begin
            if solution board[i,j] =
               solution board[i,j-1] ∧
               solution board[i+1,j] =
               solution board[i+1,j-1] then
            writetext(≮<----≯)
            else
            if (mask[i] shift -j)  ∨
                 (mask[i+1] shift -j) then
            writetext(≮<X---≯)
            else
            writetext(≮<+---≯)
         end
      end first row;
      if (mask[i] shift -BOARDX1-1)  ∨
           (mask[i+1] shift -BOARDX1-1) then
      writetext(≮<X≯)
      else if solution board[i,BOARDX1]=
         solution board[i+1,BOARDX1] then
      writetext(≮<I≯)
      else
      writetext(≮<+≯);
      writecr
   end not last row
end each row;
writetext(≮<+---≯);
for j:=1 step 1 until BOARDX1 do
begin
   if mask[BOARDY1] shift -j-1 then
   writetext(≮<XXXX≯)
   else if solution board[BOARDY1,j-1]=
      solution board[BOARDY1,j] then
   writetext(≮<----≯)
   else
   writetext(≮<+---≯)
   end first row;
   if mask[BOARDY1] shift -BOARDX1-1 then
   writetext(⋖<X⋗)
   else
   writetext(⋖<+⋗);
   writecr;
end print solution;
procedure test piece(piece count);
value piece count;
integer piece count;
begin
   integer ipiece,itransform,saveix,saveiy;
   for ipiece:=1 step 1 until 12 do
   begin
      if -, used piece[ipiece] then
      begin
         used piece[ipiece] := true;
         for itransform:=1 step 1 until ntransformed[ipiece] do
         begin
            if piece fit(ix-transformedx[ipiece,itransform],
               iy,ipiece,itransform) then
            begin
               set piece(ix-transformedx[ipiece,itransform],iy,
                  ipiece,itransform);
               set solution(ix-transformedx[ipiece,itransform],iy,
                  ipiece,itransform);
               if piece count=11 then
               begin
                  nsolutions:=nsolutions+1;
                  print solution
               end solution found
               else
               begin
                  saveix := ix;
                  saveiy := iy;
                  find first free;
                  test piece(piece count+1);
                  ix := saveix;
                  iy := saveiy
               end next piece;
               remove piece(ix-transformedx[ipiece,itransform],iy,
                  ipiece,itransform)
            end piece fit
         end itransform;
         used piece[ipiece] := false
      end unused piece
   end ipiece
end test piece;
procedure solve;
begin
   integer ipiece;
   for ipiece:=1 step 1 until 12 do
   used piece[ipiece]:=false;
   ix:=0;
   iy:=0;
   test piece(0)
end solve;

select(16);
nsolutions:=0;
transform pieces;
select(17);
create board;
clock count;
```

```
      solve;
      writecr;
      write text(≮<Solutions: ≯);
      write(≮dddd≯, nsolutions);
      writecr;
      write text(≮<Time: ≯);
      write(≮dddddd≯, clock count);
      write text(≮< sec.≯);
      writecr
   end
end;
t<
```