```
MODEL:
!  Cutting stock.  How to cut long raw material lengths
   into needed shorter Finished Good lengths;
! Uses LINGO's programming capability to do    (LoopCut*.lng)
  on-the-fly column generation for a
  cutting-stock problem.
  Cutting stock problems arise in the steel and paper industries;
! Keywords: Column Generation, Cutting Stock, Knapsack Model, LINGO, Multi-Model,
      Paper, Steel, Submodel;
SETS:
! Each Raw Material (RM) has a size, quantity available, and cost/pattern;
  RM: LENRM, QRM, CSTRM;
! We will generate a number of patterns;
  PATTERN: COST, X, YU;
! there is a set of Finished Goods (FG);
  FG: WIDTH, DEM, PRICE, Y, YIELD, MXFG, SLK;
  FXP( FG, PATTERN): NBR;
ENDSETS

DATA:
  RMWIDTH = 45;     ! Raw material width;
  FG =  F34  F24  F15  F18  F10; !Finished goods...;
  WIDTH= 34   24   15   18   10; !their widths...;
  DEM = 350  100  800  377 1001; !and demands;
  PATTERN = 1..30; ! Max patterns allowed;
  BIGM = 999;
  COSTPAT = 0; ! Cost of a pattern, regardless of copies run;
  COSTOVR = 0; ! Cost/unit over of any demand;

 ! In the above we read the inputs in-place.  There are 5 ways of exchanging
 data with LINGO:
   1) In-place, as we do above,
   2) With Excel, using statements like X = @OLE() and @OLE() = X.
       The LINGO model can also be stored in an Excel spreadsheet.
   3) With text files, using statements like X= @FILE() and @WRITE( X).
   4) With an ODBC compatible database, using statements like X=@ODBC() and @ODBC() = X.
       Some compatible databases include Access, Oracle and SQL.
   5) Via a calling application's memory locations, using statements like
      X = @POINTER(1), and @POINTER( 2) = Y.
;
ENDDATA

SUBMODEL MASTER_PROB:
! Master problem: Decide which patterns to use
    of the many patterns generated by the PATTERN_GEN submodel;
  [MSTROBJ] MIN= @SUM( PATTERN( J)| J #LE# NPATS:
         COST( J)*X( J)
        + COSTPAT*YU(j))
        + COSTOVR*@SUM( FG(i): SLK(i));
  @FOR( FG( I):
   [R_DEM]
    @SUM( PATTERN( J)| J #LE# NPATS:
     NBR( I, J) * X( J)) - SLK(i)= DEM( I);
  );
ENDSUBMODEL

SUBMODEL INTEGER_REQ:
! Model fragment to make variables integer;
  @FOR( PATTERN(j):
       @GIN( X(j));  ! Restrict to general integer values;
       @BIN( YU(j)); ! Restrict to binary (0/1) integer values;
       X(j) <= BIGM* YU(j); ! If X > 0, then YU = 1;
       );
ENDSUBMODEL

SUBMODEL PATTERN_GEN:
! Sub problem to generate another candidate column/pattern
  to add to the candidate set.
```

```
  This is a knapsack problem;
  [SUBOBJ] MAX = @SUM( FG( i): PRICE( i)* Y( i));
  @SUM( FG( i): WIDTH( i)* Y( i)) <= RMWIDTH;
  @FOR( FG( i):
      @GIN(Y( i));
      @BND(0, Y( i), MXFG(i)); ! Lower and upper bounds on Y;
       );
ENDSUBMODEL

CALC:
  ! Set parameters;
  @SET( 'DEFAULT');
  @SET( 'TERSEO',2); !Output level (0:verbose, 1:terse, 2:only errors, 3:none);

  ! Maximum number of patterns we'll allow;
  MXPATS = @SIZE( PATTERN);

  ! Generate some initial patterns, one for each FG;
  NPATS = 0;
  @FOR( FG(i):
    NPATS = NPATS + 1;
    COST( NPATS) = 1;
! first make it an empty pattern;
    @FOR( FG( i1):
       NBR( i1 , NPATS) = 0;
        );
! Then put in as many of width i as possible;
     NBR( i, NPATS) = @FLOOR( RMWIDTH/ WIDTH(i));
      );

  ! Loop as long as the reduced cost is
    attractive and there is space;
  ! Notice that we can sequentially/iteratively solve several different optimization
    models, using the output from one as input to another optimization model;
  ! First limit ourselves to 1-up patterns;
  @FOR( FG(i): MXFG(i) = 1;); ! Max copies of any FG allowed in a pattern;
  MORE = 1; ! Keep generating as long as MORE > 0;
  @WHILE( MORE:
    RC = -1;   ! Clearly attractive initially;
    @WHILE( RC #LT# 0 #AND# NPATS #LT# MXPATS:
    ! Solve for best patterns to run among ones
       generated so far;
      @SOLVE( MASTER_PROB);
    ! Copy dual prices to PATTERN_GEN submodel;
      @FOR( FG( I): PRICE( I) = -@DUAL( R_DEM( I)));
    ! Generate the current most attractive pattern;
      @SOLVE( PATTERN_GEN);
    ! Marginal value of current best pattern;
      RC = 1 - SUBOBJ;
    ! Add the pattern to the Master if it is attractive;
      @IFC( RC #LT# 0:
        NPATS = NPATS + 1;
        @FOR( FG( I): NBR( I, NPATS) = Y( I));
        COST( NPATS) = 1;
        );
      ); ! @WHILE ( RC #LT#  ;
      @IFC( MXFG(1) #LT# BIGM:
           @FOR( FG(i): MXFG(i) = BIGM;);
         @ELSE
           MORE = 0; ! We are done;
          );
    ); ! @WHILE MORE;

  ! Finally solve Master as an IP;
  @SOLVE( MASTER_PROB, INTEGER_REQ);

! This following calc section displays the
  solution in a tabular format;
```

```
  ! Compute yield of each FG;
  @FOR( FG( F): YIELD( F) =
   @SUM( PATTERN( J)| J #LE# NPATS:
    NBR( F, J) * X(J))
  );
  ! Compute some stats;
  TOTAL_FT_USED = @SUM( PATTERN: X) * RMWIDTH;
  TOTAL_FT_YIELD = @SUM( FG: YIELD * WIDTH);
  PERC_WASTE =  100 * ( 1 - ( TOTAL_FT_YIELD / TOTAL_FT_USED)) ;
 ! Display the table of patterns and their usage;
  FW = 6;
  @WRITE( @NEWLINE( 1));
  @WRITE( ' Total raws used:    ', @SUM(PATTERN: X) , @NEWLINE( 2),
          ' Total feet yield:   ', TOTAL_FT_YIELD , @NEWLINE( 1),
          ' Total feet used:    ', TOTAL_FT_USED , @NEWLINE( 2),
          ' Percent waste:    ', @FORMAT( PERC_WASTE, '#5.2G'), '%', @NEWLINE( 1));
  @WRITE( @NEWLINE( 1), 24*' ', 'Pattern:', @NEWLINE( 1));
  @WRITE( '   FG  Demand Yield');
  Inum = 0;
  @FOR( PATTERN( I) | I #LE# NPATS #AND# X( I) #GT# 0.5:
      Inum = Inum + 1;
       @WRITE( @FORMAT( Inum, '6.6G'))
       );
  @WRITE( @NEWLINE( 1));

  @WRITE( '====================');
  @FOR( PATTERN( i )| I #LE# NPATS #AND# X( I) #GT# 0.5:
     @WRITE( '======');
      );
  @WRITE( @NEWLINE( 1));

  @FOR( FG( F):
    @WRITE((FW - @STRLEN( FG( F)))*' ', FG( F), ' ',
      @FORMAT( DEM( F), '6.6G'), @FORMAT( YIELD( F), '6.6G'));
    @FOR( FXP( F, P) | P #LE# NPATS #AND# X( P) #GT# 0.5:
      @WRITE( @IF( NBR( F, P) #GT# 0,
      @FORMAT( NBR( F, P), "6.6G"), '     .')));
    @WRITE( @NEWLINE( 1))
       );
  @WRITE( '====================');
  @FOR( PATTERN( i )| I #LE# NPATS #AND# X( I) #GT# 0.5:
     @WRITE( '======');
      );
  @WRITE( @NEWLINE( 1));

  @WRITE( 2*FW*' ', ' Usage:');
  @FOR( PATTERN( P) | P #LE# NPATS #AND# X( P) #GT# 0.5 :
    @WRITE( @FORMAT( X( P), '6.6G'));
      );
  @WRITE( @NEWLINE( 1));
  @WRITE( '            Length:');
  @FOR( PATTERN( P) | P #LE# NPATS #AND# X( P) #GT# 0.5 :
    PLen= @SUM( FG( f): WIDTH( f) * NBR( f, p));
    @WRITE( @FORMAT( PLen, '6.6G'));
      );
ENDCALC

END
```