ISEN 622 Project

Team Members: **Himanshu Gupta**

Gourav Ghoshal

Sukhada Saoji

Problem A:

Lot-Sizing with Constant Capacity (LSC) problem

Indices:

Time period: $t = 1,..,n$

Parameters (given data):

p_t: unit production cost in period t
h_t: unit storage cost in period t
d_t: demand in period t
C_t: maximum production capacity in period t

Variables:
x_t = number of units of product produced in period t
s_t = number of product units carried in inventory from period t to t+1

Formulations:

$$\min \quad \sum_{t=1}^{n} p_t x_t + h_t s_t$$

s.t.

$s_t = s_{t-1} + x_t - d_t$	$t = 1,..,n$
$s_0 = 0$	
$0 \leq x_t \leq C_t$	$t = 1,..,n$
$s_t \geq 0$	$t = 1,..,n$

Problem B:

lsc.mod

```
### Model file for Lot-Sizing with Constant Capacity (LSC) problem

# Defining the production period parameter
param n;                 # n is the production period

## Setting the time indices
set time:= {1..n};             # set of indices

## Define all other parameters of cost, demand and capacity

param p {time} >= 0;    # p is per unit production cost which should be >=0 for all periods
param h {time} >= 0;    # h is per unit holding cost which should be >=0 for all periods
param d {time} >= 0;    # d is demand in any period which should be >=0 for all periods
param C {time} >= 0;    # production capacity in any period which should be >=0 for all periods

## Declaring the variables
var x {t in time} >= 0;  # x units are produced at time t which should be >= 0 and less then capacity at time t (constant in this problem)
var s {t in 0..n} >= 0; # Remaining Inventory at time period t after satisfying the demand. It should always be positive so that backorders are not carried in next period.

## Defining objective function. This is sum of production cost and inventory storing cost for all period.
minimize cost : sum {t in time} (p[t] * x[t] + h[t] * s[t]);

# Defining constraint 1, production at time t is <= capacity in that period
s.t. prod {t in time} : x[t] <= C [t];
# Defining constraint 2, at t=0, inventory = 0
s.t. initial_inventory : s [0] = 0;

# Defining constraint 3, inventory at time t = carry over inventory from previous period + production in current period - demand of current period
s.t. flow_constraint {t in time} : s[t] = s[t-1] + x[t] - d[t];
```

lsc.dat

```
param n := 90;

param p := 92 92 104 81 108 98 99 88 113 83
93 88 101 118 85 88 89 109 113 104
90 94 99 102 112 91 93 102 113 103
89 95 110 105 110 87 113 101 118 118
90 113 82 104 91 89 108 84 88 107
84 99 114 112 104 91 83 107 85 116
101 106 104 91 101 98 86 112 102 115
83 99 118 83 108 93 91 113 84 96
87 96 111 86 85 104 103 94 96 99;
```

```
param h :=
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10;


param d :=
134 118 33 185 147 144 123 181 47 74
16 115 109 107 109 126 175 144 122 70
135 176 174 31 115 170 108 153 17 149
148 47 73 149 161 46 179 17 40 121
40 155 116 89 46 37 80 38 86 94
114 94 81 76 165 100 149 72 61 102
26 100 24 131 99 91 20 76 104 112
160 37 105 63 159 39 144 62 171 56
68 46 104 40 15 61 96 185 133 22;

param C :=
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180;
```

lsc.run

```
# reset the ampl environment
reset;

# load the model
model lsc.mod;

# load the data
data lsc.dat;

# choose CPLEX as solver
option solver cplex;

# solving step - this would calculate the minimum cost for the problem
solve;

# display and save results in the output file

printf('The optimum total cost for lsc problem is ') >lsc.out;
display cost > lsc.out;
```

```
printf('The production units in each period are ') >lsc.out;
display x > lsc.out;

printf('The storage inventory in each period are ') >lsc.out;
display s > lsc.out;
```

lsc.out

```
 The optimum total cost for lsc problem is cost = 853945

 The production units in each period are x [*] :=
 1 134    11  16    21 135    31 148    41 180    51 180    61  26    71 180    81  68
 2 156    12 180    22 176    32 120    42  15    52 109    62 100    72 122    82 150
 3   0    13 151    23 174    33   0    43 180    53   0    63  24    73   0    83   0
 4 180    14   0    24  31    34 149    44  25    54  76    64 180    74 180    84  40
 5 147    15 109    25 115    35 161    45  46    55 165    65  50    75  42    85  76
 6 144    16 180    26 170    36 180    46 117    56 100    66  91    76  39    86   0
 7 124    17 180    27 108    37  45    47   0    57 180    67  96    77 180    87 101
 8 180    18  85    28 170    38  57    48  38    58  41    68   0    78  26    88 180
 9  47    19 122    29   0    39   0    49 180    59 163    69 180    79 180    89 133
10  74    20  70    30 149    40 121    50   0    60   0    70  36    80  47    90  22
;

 The storage inventory in each period are s [*] :=
 0    0    11    0    22    0    33    0    44    0    55    0    66    0    77   36    88    0
 1    0    12   65    23    0    34    0    45    0    56    0    67   76    78    0    89    0
 2   38    13  107    24    0    35    0    46   80    57   31    68    0    79    9    90    0
 3    5    14    0    25    0    36  134    47    0    58    0    69   76    80    0
 4    0    15    0    26    0    37    0    48    0    59  102    70    0    81    0
 5    0    16   54    27    0    38   40    49   94    60    0    71   20    82  104
 6    0    17   59    28   17    39    0    50    0    61    0    72  105    83    0
 7    1    18    0    29    0    40    0    51   66    62    0    73    0    84    0
 8    0    19    0    30    0    41  140    52   81    63    0    74  117    85   61
 9    0    20    0    31    0    42    0    53    0    64   49    75    0    86    0
10    0    21    0    32   73    43   64    54    0    65    0    76    0    87    5
;
```

Problem C:

1. The following command gives the version of CPLEX software being used.
   option cplex_options 'version';
   We have used CPLEX 12.7.1.0.

2. Computer Specifications:
   Processor: Intel(R) Core™ i7-6700HQ CPU @ 2.60 GHz
   RAM: 16 GB RAM
   Operating System: 64-bit, x-64 based processor

3. The following command was used to get the input time, solve time, and output time to solve LSC.
   option cplex_options 'timing 4';
   Output :-
   Times (ticks):
   Input = 0.00516224
   Solve = 0.468305
   Output = 0.0011158

   Total time is 0.47458304 ticks.

   When the command - option cplex_options 'timing 1'; is used, time output is obtained in seconds. But time taken for processing is very small to be detected in seconds unit. Hence, we calculated time in terms of ticks.

   Output in Seconds:
   Times (seconds):
   Input = 0
   Solve = 0
   Output = 0

4. Optimal objective function value for LSC is 853945.

5. C is defined as a parameter with its value equal to 180. Production is defined as a constraint which is less than or equal to C. Thus, by carrying out sensitivity analysis on production in time t; the range of parameter C, for which optimal solution remains optimal can be found out.
   The following commands were used:

```
reset;
model lsc.mod;
data lsc.dat;
option solver cplex;
option presolve 0;
option cplex_options 'sensitivity';
solve;


ampl: display prod.up;
prod.up [*] :=
 1 1e+20   14 1e+20   27 1e+20   40 1e+20   53 1e+20   66 1e+20   79   227
 2 1e+20   15 1e+20   28 1e+20   41   195   54 1e+20   67 1e+20   80 1e+20
 3 1e+20   16   265   29 1e+20   42 1e+20   55 1e+20   68 1e+20   81 1e+20
 4   185   17   265   30 1e+20   43   205   56 1e+20   69   216   82 1e+20
 5 1e+20   18 1e+20   31 1e+20   44 1e+20   57   221   70 1e+20   83 1e+20
```

```
  6 1e+20   19 1e+20   32 1e+20   45 1e+20   58 1e+20   71   302   84 1e+20
  7 1e+20   20 1e+20   33 1e+20   46 1e+20   59 1e+20   72 1e+20   85 1e+20
  8   181   21 1e+20   34 1e+20   47 1e+20   60 1e+20   73 1e+20   86 1e+20
  9 1e+20   22 1e+20   35 1e+20   48 1e+20   61 1e+20   74   222   87 1e+20
 10 1e+20   23 1e+20   36   225   49   180   62 1e+20   75 1e+20   88   185
 11 1e+20   24 1e+20   37 1e+20   50 1e+20   63 1e+20   76 1e+20   89 1e+20
 12   331   25 1e+20   38 1e+20   51   289   64   230   77   206   90 1e+20
 13 1e+20   26 1e+20   39 1e+20   52 1e+20   65 1e+20   78 1e+20
 ;

 ampl: display prod.down;
 prod.down [*] :=
  1 134   11  16   21 135   31 148   41  40   51 114   61  26   71 160   81  68
  2 156   12 151   22 176   32 120   42  15   52 109   62 100   72 122   82 150
  3   0   13 151   23 174   33   0   43 116   53   0   63  24   73   0   83   0
  4 156   14   0   24  31   34 149   44  25   54  76   64 131   74  63   84  40
  5 147   15 109   25 115   35 161   45  46   55 165   65  50   75  42   85  76
  6 144   16 126   26 170   36  46   46 117   56 100   66  91   76  39   86   0
  7 124   17 121   27 108   37  45   47   0   57 149   67  96   77 144   87 101
  8 124   18  85   28 170   38  57   48  38   58  41   68   0   78  26   88 101
  9  47   19 122   29   0   39   0   49  38   59 163   69 104   79 171   89 133
 10  74   20  70   30 149   40 121   50   0   60   0   70  36   80  47   90  22
 ;

 ampl: display prod.current;
 prod.current [*] :=
 1 180   11 180   21 180   31 180   41 180   51 180   61 180   71 180   81 180
 2 180   12 180   22 180   32 180   42 180   52 180   62 180   72 180   82 180
 3 180   13 180   23 180   33 180   43 180   53 180   63 180   73 180   83 180
 4 180   14 180   24 180   34 180   44 180   54 180   64 180   74 180   84 180
 5 180   15 180   25 180   35 180   45 180   55 180   65 180   75 180   85 180
 6 180   16 180   26 180   36 180   46 180   56 180   66 180   76 180   86 180
 7 180   17 180   27 180   37 180   47 180   57 180   67 180   77 180   87 180
 8 180   18 180   28 180   38 180   48 180   58 180   68 180   78 180   88 180
 9 180   19 180   29 180   39 180   49 180   59 180   69 180   79 180   89 180
10 180   20 180   30 180   40 180   50 180   60 180   70 180   80 180   90 180
     ;
```

6. Following command is used to set time equal to half the total run time reported in part 3.

```
reset;
option solver cplex;
option cplex_options 'dettimelim= 0.23728752';
model lsc.mod;
data lsc.dat;
solve;

display cost;
```

```
CPLEX 12.7.1.0: unrecoverable failure: CPLEX error # 25.
62 dual simplex iterations (0 in phase I)
cost = 511203

optimal objective function cost = 511203
```

Problem D:

Lot-Sizing with Capacity Module (LSCM) problem

Indices:

Time period: $t = 1,..,n$

Parameters (given data):

p_t: unit production cost in period t
h_t: unit storage cost in period t
d_t: demand in period t
C_t: maximum production capacity in period t
f_t: cost of installing unit production module in period t

Variables:
x_t: number of units of product produced in period t
s_t: number of product units carried in inventory from period t to t+1
y_t: number of modules installed in period t

Formulations:

min $\quad \sum_{t=1}^{n} p_t x_t + h_t s_t + f_t y_t$

s.t.

$s_t = s_{t-1} + x_t - d_t$	$t = 1,..,n$
$s_0 = 0$	
$y_t \geq 0$,	y_t = an integer (0, 1, 2, 3..)
$0 \leq x_t \leq y_t C_t$	$t = 1,..,n$
$s_t \geq 0$	$t = 1,..,n$

Problem E:

lscm.mod

```
### Model file for Lot-Sizing with Capacity modules (LSCM) problem

# Defining the production period parameter
param n;                # n is the production period

## Setting the time indices
set time:= {1..n};            # set of indices

## Define all other parameters of cost, demand, capacity and module installation
cost
param p {time} >= 0;    # p is per unit production cost (>=0 for any periods)
param h {time} >= 0;    # h is per unit holding cost (>=0 for any periods)
param d {time} >= 0;    # d is demand in any period (>=0 for any periods)
param C {time} >= 0;    #  production  capacity  in  any  period  (>=0  for  any
periods)
param f {time} >= 0;    # module installation cost (given as 5000)

## Declaring the variables
var y {t in 1..n} integer>= 0; # no. of modules installed at each time period
(this should be an integer value)
var x {t in time} >= 0; # x units are produced at time t which should be >= 0
and less then capacity at time t
var s {t in 0..n} >= 0; # Remaining Inventory at time period t after satisfying
the demand.

##  Defining  objective  function.  This  is  sum  of  production  cost,  inventory
storing cost and module installation cost for all period.
minimize cost : sum {t in time} (f[t] * y[t] + p[t] * x[t] + h[t] * s[t]);

# Defining constraint 1 on production units. this should be lesser than or equal
to number of modules*capacity of each module
s.t. production_constraint {t in time} : x[t] <= y[t]*C[t];

# Defining constraint 2, at t=0, inventory = 0
s.t. initial_inventory : s [0] = 0;

# Defining constraint 3, inventory at time t = carry over inventory from previous
period + production in current period - demand of current period
s.t. flow_constraint {t in time} : s[t] = s [t-1] + x[t] - d[t];
```

lscm.dat

```
param n := 90;

param p := 92 92 104 81 108 98 99 88 113 83
93 88 101 118 85 88 89 109 113 104
90 94 99 102 112 91 93 102 113 103
89 95 110 105 110 87 113 101 118 118
90 113 82 104 91 89 108 84 88 107
84 99 114 112 104 91 83 107 85 116
101 106 104 91 101 98 86 112 102 115
83 99 118 83 108 93 91 113 84 96
```

```
87 96 111 86 85 104 103 94 96 99;

param h :=
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10;


param d :=
134 118 33 185 147 144 123 181 47 74
16 115 109 107 109 126 175 144 122 70
135 176 174 31 115 170 108 153 17 149
148 47 73 149 161 46 179 17 40 121
40 155 116 89 46 37 80 38 86 94
114 94 81 76 165 100 149 72 61 102
26 100 24 131 99 91 20 76 104 112
160 37 105 63 159 39 144 62 171 56
68 46 104 40 15 61 96 185 133 22;

param C :=
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180;

param f :=

1 5000   13 5000   25 5000   37 5000   49 5000   61 5000   73 5000   85 5000
 2 5000   14 5000   26 5000   38 5000   50 5000   62 5000   74 5000   86 5000
 3 5000   15 5000   27 5000   39 5000   51 5000   63 5000   75 5000   87 5000
 4 5000   16 5000   28 5000   40 5000   52 5000   64 5000   76 5000   88 5000
 5 5000   17 5000   29 5000   41 5000   53 5000   65 5000   77 5000   89 5000
 6 5000   18 5000   30 5000   42 5000   54 5000   66 5000   78 5000   90 5000
 7 5000   19 5000   31 5000   43 5000   55 5000   67 5000   79 5000
 8 5000   20 5000   32 5000   44 5000   56 5000   68 5000   80 5000
 9 5000   21 5000   33 5000   45 5000   57 5000   69 5000   81 5000
10 5000   22 5000   34 5000   46 5000   58 5000   70 5000   82 5000
11 5000   23 5000   35 5000   47 5000   59 5000   71 5000   83 5000
12 5000   24 5000   36 5000   48 5000   60 5000   72 5000   84 5000
```

lscm.run

```
# reset the ampl environment
reset;

# load the model
model lscm.mod;

# load the data
data lscm.dat;

# choose CPLEX as solver
option solver cplex;

#option cplex_options "timing 4";
# solving step
solve;

# display and save results in the output file
printf('The optimum total cost for lscm problem is ') >lscm.out;
display cost > lscm.out;

printf('The number of modules installed in each period are ') >lscm.out;
display y > lscm.out;

printf('The production units in each period are ') >lscm.out;
display x > lscm.out;

printf('The storage inventory in each period are ') >lscm.out;
display s > lscm.out;
```

lscm.out

```
The optimum total cost for lscm problem is cost = 1134640

The number of modules installed in each period are y [*] :=
 1 1    10 0    19 0    28 1    37 0    46 1    55 1    64 2    73 0    82 1
 2 1    11 0    20 0    29 0    38 1    47 0    56 1    65 0    74 2    83 0
 3 0    12 2    21 1    30 1    39 0    48 0    57 1    66 0    75 0    84 0
 4 2    13 0    22 1    31 1    40 0    49 1    58 0    67 1    76 0    85 1
 5 0    14 0    23 1    32 0    41 1    50 0    59 1    68 0    77 1    86 0
 6 1    15 1    24 1    33 0    42 0    51 2    60 0    69 1    78 0    87 0
 7 0    16 1    25 0    34 1    43 2    52 0    61 0    70 0    79 2    88 1
 8 2    17 2    26 1    35 1    44 0    53 0    62 1    71 2    80 0    89 1
 9 0    18 0    27 1    36 1    45 0    54 0    63 0    72 0    81 0    90 0
;

The production units in each period are x [*] :=
1 164    11   0    21 135    31 180    41 180    51 360    61   0    71 302    81   0
2 180    12 357    22 176    32   0    42   0    52   0    62 124    72   0    82 180
3   0    13   0    23 174    33   0    43 251    53   0    63   0    73   0    83   0
4 360    14   0    24 146    34 180    44   0    54   0    64 321    74 287    84   0
5   0    15 180    25   0    35 180    45   0    55 165    65   0    75   0    85 177
6 180    16 180    26 170    36 180    46 160    56 150    66   0    76   0    86   0
7   0    17 360    27 163    37   0    47   0    57 180    67 132    77 180    87   0
8 318    18   0    28 180    38 180    48   0    58   0    68   0    78   0    88 180
```

```
  9    0    19    0    29    0    39    0    49 180    59 180    69 180    79 305    89 155
 10    0    20    0    30 180    40    0    50    0    60    0    70    0    80    0    90    0
;
```

```
   The storage inventory in each period are s [*] :=
  0    0    11    0    22    0    33    8    44   46    55    0    66    0    77   62    88    0
  1   30    12  242    23    0    34   39    45    0    56   50    67  112    78    0    89   22
  2   92    13  133    24  115    35   58    46  123    57   81    68   36    79  134    90    0
  3   59    14   26    25    0    36  192    47   43    58    9    69  112    80   78
  4  234    15   97    26    0    37   13    48    5    59  128    70    0    81   10
  5   87    16  151    27   55    38  176    49   99    60   26    71  142    82  144
  6  123    17  336    28   82    39  136    50    5    61    0    72  105    83   40
  7    0    18  192    29   65    40   15    51  251    62   24    73    0    84    0
  8  137    19   70    30   96    41  155    52  157    63    0    74  224    85  162
  9   90    20    0    31  128    42    0    53   76    64  190    75   65    86  101
 10   16    21    0    32   81    43  135    54    0    65   91    76   26    87    5
 ;
```

Problem F:

Part -1

LSCM run time –
```
Times (seconds):
Input =  0
Solve =  4.65625
Output = 0
```
Total time = `4.65625`

LSC run time
Total time = 0.47458304 ticks which is equal to 0.000000004745 seconds

LSCM problem takes much larger time to solve compared to LSC problem.

Part -2

Lsc problem for part F has additional term of 5000 in the objective function from previous problem.

- Optimal objective function value for LSCM = 1134640 units
- Optimal objective function value for LSC plus 5000n = 1303945 units
- Theoretically the value of LSCM is smaller than LSC with installation cost model.

This is because in LSCM, only 53 modules of production units are installed to meet the required demand in the stipulated time period. Whereas, in the LSC model, there are 90 modules installed in time n = 90 unit, since, for each cycle, it was required to install one production module.

By allowing to have various number of modules installed in the system, the LSCM became more flexible, and the system produced better optimal solution. It is possible to have no production module for a time period and cater demand from the inventory stored. Although this increase the inventory carrying cost from 19,090 units in LSC model to 66,480 units in LSCM, the module installation cost reduced from 450,000 units to 265,000 units in LSCM, resulting in a lower optimal value. Also, higher inventory carrying opportunity allowed the LSCM to produce product during the period when the unit production costs are lower, reducing the production cost from 834,855 units in LSC model to 803,160 units in LSCM, which also helps in generating better optimal value.

Scripts for this problem -

`lsc_F.mod`

```
### Model file for Lot-Sizing with Constant Capacity (LSC) problem with
installation cost = 5000

# Defining the production period parameter
param n;                # n is the production period

## Setting the time indices
set time:= {1..n};          # set of indices
```

```
## Define all other parameters of cost, demand and capacity

param p {time} >= 0;  # p is per unit production cost which should be >=0 for all periods
param h {time} >= 0;  # h is per unit holding cost which should be >=0 for all periods
param d {time} >= 0;  # d is demand in any period which should be >=0 for all periods
param C {time} >= 0;  # production capacity in any period which should be >=0 for all periods

## Declaring the variables
var x {t in time} >= 0, <= C [t];   # x units are produced at time t which should be >= 0 and less then capacity at time t (constant in this problem)
var s {t in 0..n} >= 0; # Remaining Inventory at time period t after satisfying the demand. It should always be positive so that backorders are not carried in next period.

## Defining objective function. This is sum of production cost and inventory storing cost for all period.
minimize cost : sum {t in time} (p[t] * x[t] + h[t] * s[t] + 5000);

# Defining constraint 1, at t=0, inventory = 0
s.t. initial_inventory : s [0] = 0;

# Defining constraint 2, inventory at time t = carry over inventory from previous period + production in current period - demand of current period
s.t. flow_constraint {t in time} : s[t] = s [t-1] + x[t] - d[t];
```

lsc_F.run

```
# reset the ampl environment
reset;

# load the model
model lsc_F.mod;

# load the data
data lsc.dat;

# choose CPLEX as solver
option solver cplex;

# solving step - this would calculate the minimum cost for the problem
solve;

# display and save results in the output file
printf('The optimum total cost for lsc problem with installation cost is ') >lsc_F.out;
display cost > lsc_F.out;
```

```
printf('The production units in each period are ') >lsc_F.out;
display x > lsc_F.out;

printf('The storage inventory in each period are ') >lsc_F.out;
display s > lsc_F.out;
```

lsc.dat

```
param n := 90;

param p := 92 92 104 81 108 98 99 88 113 83
93 88 101 118 85 88 89 109 113 104
90 94 99 102 112 91 93 102 113 103
89 95 110 105 110 87 113 101 118 118
90 113 82 104 91 89 108 84 88 107
84 99 114 112 104 91 83 107 85 116
101 106 104 91 101 98 86 112 102 115
83 99 118 83 108 93 91 113 84 96
87 96 111 86 85 104 103 94 96 99;

param h :=
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10
10    10    10    10    10    10    10    10    10    10;


param d :=
134 118 33 185 147 144 123 181 47 74
16 115 109 107 109 126 175 144 122 70
135 176 174 31 115 170 108 153 17 149
148 47 73 149 161 46 179 17 40 121
40 155 116 89 46 37 80 38 86 94
114 94 81 76 165 100 149 72 61 102
26 100 24 131 99 91 20 76 104 112
160 37 105 63 159 39 144 62 171 56
68 46 104 40 15 61 96 185 133 22;

param C :=
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180
180   180   180   180   180   180   180   180   180   180;
```

lsc_F.out

```
  The optimum total cost for lsc problem with installation cost is cost = 1303940

  The production units in each period are x [*] :=
 1 134    11  16    21 135    31 148    41 180    51 180    61  26    71 180    81  68
 2 156    12 180    22 176    32 120    42  15    52 109    62 100    72 122    82 150
 3   0    13 151    23 174    33   0    43 180    53   0    63  24    73   0    83   0
 4 180    14   0    24  31    34 149    44  25    54  76    64 180    74 180    84  40
 5 147    15 109    25 115    35 161    45  46    55 165    65  50    75  42    85  76
 6 144    16 180    26 170    36 180    46 117    56 100    66  91    76  39    86   0
 7 124    17 180    27 108    37  45    47   0    57 180    67  96    77 180    87 101
 8 180    18  85    28 170    38  57    48  38    58  41    68   0    78  26    88 180
 9  47    19 122    29   0    39   0    49 180    59 163    69 180    79 180    89 133
10  74    20  70    30 149    40 121    50   0    60   0    70  36    80  47    90  22
;

  The storage inventory in each period are s [*] :=
 0   0    11   0    22   0    33   0    44   0    55   0    66   0    77  36    88   0
 1   0    12  65    23   0    34   0    45   0    56   0    67  76    78   0    89   0
 2  38    13 107    24   0    35   0    46  80    57  31    68   0    79   9    90   0
 3   5    14   0    25   0    36 134    47   0    58   0    69  76    80   0
 4   0    15   0    26   0    37   0    48   0    59 102    70   0    81   0
 5   0    16  54    27   0    38  40    49  94    60   0    71  20    82 104
 6   0    17  59    28  17    39   0    50   0    61   0    72 105    83   0
 7   1    18   0    29   0    40   0    51  66    62   0    73   0    84   0
 8   0    19   0    30   0    41 140    52  81    63   0    74 117    85  61
 9   0    20   0    31   0    42   0    53   0    64  49    75   0    86   0
10   0    21   0    32  73    43  64    54   0    65   0    76   0    87   5
;
```