SOLUTION PROBLEM I

```
set    PROD         ordered                                    ; # Clock, Radio, Toaster

param  Cst          {i in PROD}                                ; # Cost per unit per product
param  Lab          {i in PROD}                                ; # Labor hours per unit per product
param  Dmd          {i in PROD}                                ; # Maximum Demand per product
param  Spr          {i in PROD}                                ; # Selling price per product
param  Bdg                                                     ; # Production Budget
param  Cap                                                     ; # Capacity (total available labor hours)

var    QP           {i in PROD} integer >= 0                   ; # Quantity produced per product


maximize TotPrf     : sum {i in PROD} QP[i] * ( Spr[i] - Cst[i] ) ; # Maximize Total Profit

s.t. DmdCon         {i in PROD} : QP[i] <= Dmd[i]              ; # Quantity produced <= Maximum Demand for each product
s.t. BdgCon         : sum {i in PROD} QP[i] * Cst[i] <= Bdg    ; # Total production cost <= Production Budget
s.t. CapCon         : sum {i in PROD} QP[i] * Lab[i] <= Cap    ; # Total Quantity produced <= Capacity


data                                                                    ;

set    PROD         :=          Clock       Radio       Toast          ;

param               :           Cst         Lab         Dmd         Spr    :=
                    Clock        7           2          200          15
                    Radio       10           3          300          20
                    Toast        5           2          150          12    ;

param  Bdg          :=          2000                                       ;
param  Cap          :=           660                                       ;


option solver gurobi                        ;
solve                                       ;
option display_width 200, display_1col 0    ;

display QP                                  ;
```

SOLUTION PROBLEM J

```
set    ORIG         ordered                                          ; # A, B, C
set    DEST         ordered                                          ; # 1, 2, 3

param  Cst          {i in ORIG, j in DEST}                           ; # Transportation Cost from Origin to Destination
param  Sup          {i in ORIG}                                      ; # Maximum Supply Capacity per Origin
param  Dmd          {j in DEST}                                      ; # Demand per Destination

var    QS           {i in ORIG, j in DEST} integer >= 0              ; # Quantity Shipped from Origin to Destination


minimize TotCst     : sum {i in ORIG, j in DEST} QS[i,j] * Cst[i,j]  ; # Minimize Total Cost

s.t. SupCon         {i in ORIG} : sum {j in DEST} QS[i,j] <= Sup[i]  ; # Quantity Shipped <= Supply Capacity per Origin
s.t. DmdCon         {j in DEST} : sum {i in ORIG} QS[i,j] >= Dmd[j]  ; # Quantity Shipped >= Demand per Destination


data                                                                 ;

set    ORIG         :=           A            B            C         ;
set    DEST         :=           1            2            3         ;

param  Cst          :            1            2            3         :=
                    A            6            9            100
                    B            12           3            5
                    C            4            8            11        ;

param  Sup          :=           A  130       B  70        C  100    ;

param  Dmd          :=     1  80  2 110  3  60  ;


option solver gurobi                          ;
solve                                         ;
option display_width 200, display_1col 0      ;

display QS                                    ;
```

SOLUTION PROBLEM K

```
param  I                                                                    ; # Number of Rows: 3
param  J                                                                    ; # Number of Columns: 3

param  Arr          {i in 1..I, j in 1..J}                                  ; # Array Parameter
param  Ver          {i in 1..I}                                             ; # Vertical Parameter
param  Hor          {j in 1..J}                                             ; # Horizontal Parameter

var    X            {i in 1..I, j in 1..J} >= 0                             ; # Variable


minimize Z          : sum {i in 1..I, j in 1..J} X[i,j] * Arr[i,j]          ; # Minimize Function Z

s.t. VerCon  {i in 1..I} : sum {j in 1..J} X[i,j]  = Ver[i]                 ; # Vertical Constraint
s.t. HorCon  {j in 1..J} : sum {i in 1..I} X[i,j] <= Hor[j]                 ; # Horizontal Constraint


data                                                                        ;

param  I            :=          3                                           ;
param  J            :=          3                                           ;

param  Arr          :           1           2           3           :=
                    1           3           12          8
                    2           10          5           6
                    3           6           7           10          ;

param  Ver          :=          1  90       2  30       3 100       ;
param  Hor          :=          1  70       2 110       3  80       ;


option solver gurobi                        ;
solve                                       ;
option display_width 200, display_1col 0    ;

display X                                   ;
```