Problem 1:

```
#---------------------------Sets-----------------------------------
set GENERATOR;              #Generators in question
set PERIOD;                 #Period of production

#-----------------------Parameters---------------------------------

param S {GENERATOR} >= 0;       #Start-up cost
param F {GENERATOR} >= 0;       #Fixed cost of running per period
param C {GENERATOR} >= 0;       #Cost per megawatt
param U {GENERATOR} >= 0;       #Production limit per period
param min_prod {PERIOD} >=0;    #Minimum production per period

#-----------------------Decision Variables-------------------------

var V {GENERATOR, PERIOD} >= 0;     #Megawatts generated
var G {GENERATOR, PERIOD} binary;   #Startup bools

#-----------------------Objective Function-------------------------

minimize Cost: sum {g in GENERATOR} (G[g,'P1']*(S[g]+F[g]+C[g]*V[g,'P1'])+G[g,'P2']*(S[g]-S[g]*G[g,'P1']+F[g]+C[g]*V[g,'P2']));

#-----------------------Constraints--------------------------------

subject to demand {p in PERIOD}: sum {g in GENERATOR} V[g,p] >= min_prod[p];    #Have to meet production requirements
subject to supply {g in GENERATOR, p in PERIOD}: V[g,p] <= U[g];                #Each generator has production limits
subject to bools {p in PERIOD, g in GENERATOR}: V[g,p] <= G[g,p]*5000;          #Generator bool must be on to produce
#-----------------------------Data---------------------------------------

data "C:\Users\Dylan\Documents\OU\Classes\DSA5113\HW3\group12_HW3_p1.dat";

solve;

display V;
display G;
display Cost;
```

Results:

```
V :=
G1 P1   1100
G1 P2   2100
G2 P1   1800
G2 P2   1800
G3 P1      0
G3 P2     -9.09495e-13
;

G :=
G1 P1   1
G1 P2   1
G2 P1   1
G2 P2   1
G3 P1   0
G3 P2   0
;

Cost = 38400
```

Explanation:

The V table represents the Megawatts produced by each generator in each period. The G table represents a Boolean describing whether a generator should be run in each period, using the “Big M” method to ensure the code has reason to change these variables.

We can see that it is optimal to run both generators 1 and 2 in each period, which makes sense because this helps avoid paying start up costs in period 2. Generator 1 produces 1100 and 2100 megawatts in periods 1 and 2 respectively. Generator 2 produces 1800 megawatts in both periods 1 and 2. Generator 3 is not run. This totals a cost of 38,400.

Problem 2:

a)

```
#---------------------------Sets-----------------------------------
set PRODUCT;                #Products in question
set SILO;                   #Available Silos

#------------------------Parameters---------------------------------

param cost {PRODUCT, SILO} >= 0;    #Silo-Loading cost per ton
param supply {PRODUCT} >= 0;        #Tons of each product to store
param cap {SILO} >= 0;              #Max tons per Silo

#-------------------------Decision Variables-------------------------

var STORE {PRODUCT, SILO} >= 0;     #Tons of product in silo
var BOOLS {PRODUCT, SILO} binary;   #If product is in silo

#-------------------------Objective Function-------------------------

minimize Cost: sum{p in PRODUCT, s in SILO}cost[p,s]*STORE[p,s];

#--------------------------Constraints-------------------------------

subject to no_mixing {s in SILO}: sum {p in PRODUCT} BOOLS[p,s] <= 1;          #Only 1 product per Silo
subject to capacity {s in SILO}: sum{p in PRODUCT} STORE[p,s] <= cap[s];       #Limited to silo capacity
subject to store_all {p in PRODUCT}: sum {s in SILO} STORE[p,s] = supply[p];   #Everything must be stored
subject to bools1 {p in PRODUCT, s in SILO}: STORE[p,s] >= BOOLS[p,s]*.001;    #If bool is on, something must be stored
subject to bools0 {p in PRODUCT, s in SILO}: STORE[p,s] <= BOOLS[p,s]*500;     #If bool is off, nothing stored

#---------------------------Data-------------------------------------

data "C:\Users\Dylan\Documents\OU\DSA5113\HW3\group12_HW3_p2.dat";

solve;

display Cost;
display STORE;
display BOOLS;
```

Results:

```
STORE [*,*] (tr)
:    A    B    C    D    E     :=
S1   25    0    0    0    0
S2    0    0    0   25    0
S3    0    0   25    0    0
S4   50    0    0    0    0
S5    0   20    0    0    0
S6    0    0    0    0   20
S7    0    0    0    5    0
S8    0    0    0   50    0
;

BOOLS [*,*] (tr)
:    A   B   C   D   E    :=
S1   1   0   0   0   0
S2   0   0   0   1   0
S3   0   0   1   0   0
S4   1   0   0   0   0
S5   0   1   0   0   0
S6   0   0   0   0   1
S7   0   0   0   1   0
S8   0   0   0   1   0
;
```

Explanation:

The STORE table represents how many tons of each product are in each silo. For example, row S1 column A has a value of 25 – so there are 25 tons of product A in silo 1. Each row only has 1 non-zero value because we were constrained to a single product per silo. The BOOL table is just a table showing whether or not a product is in a given silo, 1's representing it is, 0's represnting it's not. Resulting in a minimum cost of 290.

2b)

```
#---------------------------Sets-----------------------------------
set PRODUCT;                #Products in question
set SILO;                   #Available Silos

#-------------------------Parameters---------------------------------

param cost {PRODUCT, SILO} >= 0;    #Silo-Loading cost per ton
param supply {PRODUCT} >= 0;        #Tons of each product to store
param cap {SILO} >= 0;              #Max tons per Silo
param weight_cost = .99             #Weight of how important cost is
param weight_waste = .01;           #Weight of how important empty space is

#-------------------------Decision Variables--------------------------

var STORE {PRODUCT, SILO} >= 0;     #Tons of product in silo
var BOOLS {PRODUCT, SILO} binary;   #If product is in silo
var WASTE {SILO} >= 0;      #Empty space in used silos

#-------------------------Objective Function--------------------------

minimize Cost: sum{p in PRODUCT, s in SILO}(cost[p,s]*STORE[p,s]*weight_cost+WASTE[s]*weight_waste*BOOLS[p,s]);

#-------------------------Constraints---------------------------------

subject to no_mixing {s in SILO}: sum {p in PRODUCT} BOOLS[p,s] <= 1;           #Only 1 product per silo
subject to capacity {s in SILO}: sum{p in PRODUCT} STORE[p,s] <= cap[s];        #Limited to silo capacity
subject to store_all {p in PRODUCT}: sum {s in SILO} STORE[p,s] = supply[p];    #Everything must be stored
subject to bools1 {p in PRODUCT, s in SILO}: STORE[p,s] >= BOOLS[p,s]*.001;     #If bool is on, something must be stored
subject to bools0 {p in PRODUCT, s in SILO}: STORE[p,s] <= BOOLS[p,s]*500;      #If bool is off, nothing stored
subject to emptiness {s in SILO}: WASTE[s] = cap[s] - sum{p in PRODUCT}STORE[p,s];

#---------------------------Data-----------------------------------

data "C:\Users\Dylan\Documents\OU\DSA5113\HW3\group12_HW3_p2.dat";

solve;

display STORE;
display BOOLS;
```

In our ".mod" file have now added the amount of wasted space (cap – amount stored) to our objective function along with two coefficients representing the "weight" of each objective (total cost and empty space). We can then use the weighted sum approach to solve for the optimal solution based on the given weights. Note: With the small weights used fort the waste we still get the same optimal solution.

Problem 3:

```
#---------------------------Sets-----------------------------------
set SUPPLIERS;              #Possible suppliers

#-------------------------Parameters-------------------------------

param cap {SUPPLIERS} >= 0;             #Max widgets per supplier
param demand_amt = 55000;


#-------------------------Decision Variables---------------------------

var WIDGETS {SUPPLIERS} >= 0;   #Widget total from each supplier
var BOOLS {SUPPLIERS} binary;   #If widget ordered from supplier
var d1 >= 0 <= 3000;            #WOW widgets up to 3k
var d2 >= 0 <= 6000;            #WOW widgets up to 6k
var d3 >= 0;                    #WOW widgets over 6k
var y1 binary;                  #is d1 = 3k
var y2 binary;                  #is d2 = 3k

#-------------------------Objective Function---------------------------

minimize Cost:  WIDGETS['WII']*4.95 +
                BOOLS['WRS']*20000 + WIDGETS['WRS']*2.3 +
                BOOLS['WRS']*WIDGETS['WE']*3.95 - (BOOLS['WRS'] - 1)*WIDGETS['WE']*4.1 +
                WIDGETS['WU']*4.25 +
                9.5*d1 + 4.9*d2 + 2.75*d3;

#---------------------------Constraints--------------------------------

subject to demand: sum{s in SUPPLIERS} WIDGETS[s] >= demand_amt;                #need enough widgets
subject to supply {s in SUPPLIERS}: WIDGETS[s] <= cap[s];                       #Supplier caps
subject to bools1 {s in SUPPLIERS}: WIDGETS[s] >= BOOLS[s];                     #if widgets ordered, bool on
subject to bools0 {s in SUPPLIERS}: WIDGETS[s] <= BOOLS[s]*demand_amt;          #if bool off, no widgets ordered
subject to WE_restraints: BOOLS['WE'] + BOOLS['WII'] <= BOOLS['WRS'] + 1;       #Can't buy from WE and WII unless ordering from WRS
subject to WU_minimum: WIDGETS['WU'] >= 15000*BOOLS['WU'];                      #WU has 15k order minimum
subject to WOW_linear_cost: WIDGETS['WOW'] = d1 + d2 + d3;                      #WOW is sum of d1,d2,d3
subject to y1_bool: d1 >= y1*3000;                                              #If y1 on, d1 = 3k
subject to y2_bool: d2 >= y2*6000;                                              #If y2 on, d2 = 6k
subject to logic: 6000*y2 <= d2;                                                #if d2 < 3000 y2 off
subject to logic1: d2 <= 6000*y1;                                               #If d2 > 0, y1 on
subject to logic2: d3 <= demand_amt*y2;                                         #If d3 > 0, y2 on

#---------------------------Data---------------------------------------

data "C:\Users\Dylan\Documents\OU\DSA5113\HW3\group12_HW3_p3.dat";


solve;

display Cost;
display WIDGETS;
display BOOLS;
```

Result:

| Demand | WII | WRS | WE | WU | WOW | Cost |
|---|---|---|---|---|---|---|
| 5000 | 0 | 5000 | 0 | 0 | 0 | $20500 |
| 10000 | 0 | 10000 | 0 | 0 | 0 | $43000 |
| 25000 | 4000 | 14000 | 7000 | 0 | 0 | $99650 |
| 35000 | 0 | 14000 | 6000 | 15000 | 0 | $139650 |
| 45000 | 0 | 14000 | 6000 | 0 | 25000 | $177800 |
| 50000 | 4000 | 14000 | 7000 | 0 | 25000 | $201550 |
| 55000 | 0 | 14000 | 1000 | 15000 | 25000 | $221800 |

Explanation: Above shows the optimal order amounts for each corresponding Widget demand along with the resulting costs. Again using a Boolean table and a few extra variables to make sure all given constraints were met (see .mod comments).