Exercise 1

Code in OPL the decentralization problem stated as exercise 1 in the DM.P.4 list. This model should be named as decentralization. The required input data is provided in the enclosed decentralization.dat file.

Decentralization Model:

```
{string} Cities = ...;
{string} Depts = ...;

float Benefit[Depts][Cities]=...; //Benefits to be derived from each
relocation
float CostComm[Cities][Cities]=...; //matrix representing the unit
communication cost between cities
float Comm[Depts][Depts]=...; //matrix representing the communication
quantity between departments


int LimDeptsLoc = ...;

// Decision Variables
//Binary Variables
dvar int IsIn[Depts][Cities]in 0..1;  //1 if department i in{A, B, C, D, E}
is located
                                      //in city j in {London, Bristol,
Brighton}
                                      // 0 otherwise

dvar int Link[Depts][Cities][Depts][Cities] in 0..1;

// Generic formulation of the objective function terms
dexpr float Benef = sum(j in Cities)sum(i in Depts)
Benefit[i][j]*IsIn[i][j] ; // formulate here Benefit expression
dexpr float Cost = sum(ordered i, j in Depts)
                   sum( k, l in Cities) Comm[i][j] * CostComm[k][l] *
Link[i][k][j][l]; // formulate here Cost expression

maximize  Benef- Cost;

subject to{
  //every department must be located at one single city
  forall(i in Depts)
    sum(j in Cities) IsIn[i][j]==1;

    //None of these cities can be the location for
    //more than three of the departments
  forall(i in Cities)
     sum(j in Depts) IsIn[j][i]<=LimDeptsLoc;

        forall(ordered i, j in Depts)
      forall(k, l in Cities)
        {
//if department i is located in city k and department j is located in city
l then variable Link == 1
             (IsIn[i][k]==1) &&  (IsIn[j][l]==1) => Link[i][k][j][l]==1;
// if variable Link == 1 then department i is located in city k and
department j is located in city l
             Link[i][k][j][l]==1 =>  (IsIn[i][k]==1 ) &&  (IsIn[j][l]==1) ;
```

```
        }

}
```

```
execute DISPLAY { writeln("Maximum profit = " ,
 cplex.getObjValue(), " Cost = " , Cost, " Benefit = ", Benef );
for(var d in Depts)
  for(var c in Cities)
    if(IsIn[d][c] == 1) writeln("Department ", d, " is located in ", c);
for(var d1 in Depts)
  for(var d2 in Depts)
    for(var c1 in Cities)
       for(var c2 in Cities)
          if(Link[d1][c1][d2][c2] == 1)
             writeln("Dep. ", d1, " located in ", c1,
             " links with Dep. ", d2, " located in ", c2,
             " at communication cost ", CostComm[c1][c2] );
}
```

Decentralization Solution:(Scripting log)

```
// solution (optimal) with objective 14.9
Maximum profit = 14.9 Cost = 65.1 Benefit = 80
Department A is located in Bristol
Department B is located in Brighton
Department C is located in Brighton
Department D is located in Bristol
Department E is located in Brighton
Dep. A located in Bristol links with Dep. B located in Brighton at
communication cost 14
Dep. A located in Bristol links with Dep. C located in Brighton at
communication cost 14
Dep. A located in Bristol links with Dep. D located in Bristol at
communication cost 5
Dep. A located in Bristol links with Dep. E located in Brighton at
communication cost 14
Dep. B located in Brighton links with Dep. C located in Brighton at
communication cost 5
Dep. B located in Brighton links with Dep. D located in Bristol at
communication cost 14
Dep. B located in Brighton links with Dep. E located in Brighton at
communication cost 5
Dep. C located in Brighton links with Dep. D located in Bristol at
communication cost 14
Dep. C located in Brighton links with Dep. E located in Brighton at
communication cost 5
Dep. D located in Bristol links with Dep. E located in Brighton at
communication cost 14
```

Exercise 2:

Code in OPL the steel planning problem stated as exercise 2 in the DM.P.4 list. This model should be named as steelprod. The required input data is provided in the enclosed stellprod.dat file.

Steel Planning Model:

```
 {string} Products =...; //products to be manufactured
 int T=...;
 {string}Resources=...;
 range TimePeriod= 1..T;//a vector

 float Avail[Resources][TimePeriod]=...; //availability of resources over
the periods
 float ResourceReq[Products][Resources]=...; //resource requirements are
per production unit
 float Demand[Products][TimePeriod]=...; //matrix to represent demands over
the periods

 float Inv0[Products]=...; //initial inventory
 float Backorder0[Products]=...; //initial backorder
 float EndInv[Products]=...; //Ending inventory
 float EndBlg[Products]=...; //Ending backorder

float Prodcost[Products][TimePeriod]=...;//Production costs (€/ton)
float Backlogcost[Products][TimePeriod]=...;//Backlog costs (€/ton)
float Invcost[Products][TimePeriod]=...;//Inventory costs (€/ton)

// Decision Variables
dvar float+ Make[Products][TimePeriod]; //what to make of product p during
the period T
dvar float+ Inv[Products][0..T];  //what to store of product p during the
period T
dvar float+ Backorder[Products][0..T];//what to accept as backorder of
product P during the period T

// objective function
dexpr float Cost = sum(i in Products)sum(j in TimePeriod)Prodcost[i][j]*
Make[i][j]+
                   sum(i in Products)sum(j in
TimePeriod)Backlogcost[i][j]*Backorder[i][j]+
                   sum(i in Products)sum(j in
TimePeriod)Invcost[i][j]*Inv[i][j];

minimize Cost;

subject to{

  //initial and final inventory
  forall(i in Products){
    Inv[i][0]==Inv0[i];
    Inv[i][T]==EndInv[i];
  }

  //initial and final Backorder
   forall(i in Products){
    Backorder[i][0]==Backorder0[i];
    Backorder[i][T]<=EndBlg[i];
  }

  //the balance equation between two consecutive periods
   forall(i in Products,j in TimePeriod)
     Inv[i][j-1]-Backorder[i][j-1]+Make[i][j]==Inv[i][j]-
Backorder[i][j]+Demand[i][j];
```

```
  //balance equation between required and available resources
  forall(i in Resources, j in TimePeriod)
      sum(p in Products)(Make[p][j]*ResourceReq[p][i]) <= Avail[i][j];

   }


execute DISPLAY { writeln("Minimum cost = " , cplex.getObjValue());
writeln("Initial inventories and backorders:");
for(var p in Products)
 { writeln(" Product ",p,": <Inventory: ",Inv[p][0],", Backorder: ",
Backorder[p][0],">");}

writeln();
writeln("Production Scheduling:");
 for(var t in TimePeriod) {
  writeln(" Period ", t);
  writeln(" -------- ");
    for(p in Products) {
       writeln(" Product ",p,": <Inventory: ",Inv[p][t],
        ", Backorder: ", Backorder[p][t], ", Make: ", Make[p][t],">");
                        }
                                }
writeln();
writeln("Final inventories and backorders:");
for(p in Products) {
   writeln(" Product ",p,": <Inventory: ",Inv[p][T], ", Backorder: ",
Backorder[p][T],">");
}
  }
```

Steel Planning Solution:

```
// solution (optimal) with objective 264200
Minimum cost = 264200
Initial inventories and backorders:
 Product bands: <Inventory: 0, Backorder: 200>
 Product coils: <Inventory: 500, Backorder: 0>

Production Scheduling:
 Period 1
 --------
 Product bands: <Inventory: 0, Backorder: 0, Make: 3200>
 Product coils: <Inventory: 1900, Backorder: 0, Make: 2400>
 Period 2
 --------
 Product bands: <Inventory: 100, Backorder: 0, Make: 6100>
 Product coils: <Inventory: 0, Backorder: 0, Make: 600>
 Period 3
 --------
 Product bands: <Inventory: 0, Backorder: 0, Make: 3900>
 Product coils: <Inventory: 0, Backorder: 1100, Make: 1400>
 Period 4
 --------
 Product bands: <Inventory: 200, Backorder: 0, Make: 2200>
 Product coils: <Inventory: 200, Backorder: 0, Make: 4300>

Final inventories and backorders:
```

```
Product bands: <Inventory: 200, Backorder: 0>
Product coils: <Inventory: 200, Backorder: 0>
```