**Problem – B (Assembly)**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OPL 12.6.0.0 Model

\* Author: Hakeem-ur-Rehman

\* Creation Date: Apr 3, 2017 at 11:53:53 AM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//main{

// //thisOplModel.convertAllIntVars(); // Converting IP/MIP to LP

// thisOplModel.generate();

// cplex.solve();

// writeln("The best Objective Value (Total Cost): ",cplex.getBestObjValue());

// // Computation time with respect to CPU Elapsed time "In setting(.ops); computation repoting time = CPU time"

// writeln("Solving Elapsed Time: CPU (Ending time stamp - Starting time stamp) in (Seconds): ", cplex.getCplexTime());

// }

// Define and Initialize INDICES & PARAMETERS DATA

int FP=...; range Final\_Products=1..FP;

int RP=...; range Remaining\_Products=2..RP;

int J=...; range products=1..J; // Represents All the products

int L=...; range productionstages=1..L;

int S=...; range microperiods=1..S;

int T=...; range macroperiods=1..T;

// Define & Initialize Sets

{int} allproductsonstage1 = ...; // All products at Stage One

{int} allproductsonstage2 = ...;

{int} allproductsonstage3 = ...;

{int} family1stage1 = ...; // Family-1 who has same successor in the following stage

{int} family2stage1 = ...;

{int} family3stage1 = ...;

{int} family1stage2 = ...;

{int} family2stage2 = ...;

{int} family3stage2 = ...;

{int} family1stage3 = ...;

{int} microperiods1tomacroperiod = ...;

{int} microperiods2tomacroperiod = ...;

{int} microperiods3tomacroperiod = ...;

{int} microperiods4tomacroperiod = ...;

// Declare & Initialize CONSTANT DATA

// Minimum Lotsize of the jth product

int min\_lotsize=...;

// Production cost per unit

int production\_cost=...;

// production time per unit

int production\_time=...;

// Idle time (i.e. Stand by) cost

int standby\_cost=...;

/\* Pijl --> Number of units of product 'i' required to produce

one unit of product 'j' on production stage 'l'\*/

int BOM = ...;

int BigM = ...;

// Arrays Delcarations through indicies & tuple sets

// Capacity of the Production Stages

float productstagecapacity[productionstages]=...;

// Product Holding Cost

int holdingcost[products]=...;

// Products Changeover Cost

int setupcost[products]=...;

// Products Changeover Time

int setuptime[products]=...;

// Products Demand

float primary\_demand[Final\_Products][macroperiods]=...;

float secondary\_demand[Remaining\_Products][macroperiods]=...;

// Defining Decision Variables

// Inventory Level of jth Product on Lth production stage in Tth macroperiod

dvar float+ inventory[products][0..T];

// Total Production Quality of the products on machines'm' in 't'

dvar float+ productionquantity[products][productionstages][microperiods];

// Product Changeover in Microperiod 's'

dvar float+ Pchangeover[products][products][productionstages][microperiods];

// Fractional setup time for changeover at the begining of microperiod 's'

dvar float+ B\_setuptime[productionstages][microperiods];

// Fractional setup time for changeover at the end of microperiod 's'

dvar float+ E\_setuptime[productionstages][microperiods];

// Standby (idle) time on machine 'l' in microperiod 's'

dvar float+ sb[productionstages][microperiods];

//dvar float+ sb[allproductstage][micro\_macroperiods];

// Lth Machine setup for jth Product in sth Microperiod

dvar boolean stagesetup[products][productionstages][microperiods];

// Computing the objective function value

dexpr float TotalProductionCost = sum(j in products, l in productionstages, s in microperiods)

production\_cost\*productionquantity[j][l][s];

dexpr float TotalHoldingCost = sum(j in products, t in macroperiods)

holdingcost[j]\*inventory[j][t];

dexpr float TotalSetupCost = sum(i,j in products, l in productionstages, s in microperiods)

setupcost[j]\*Pchangeover[i][j][l][s];

dexpr float TotalStandbyCost = sum(l in productionstages, s in microperiods)

standby\_cost\*sb[l][s];

// Total Value of the Objective Function

dexpr float TOTAL\_COST = TotalProductionCost + TotalHoldingCost + TotalSetupCost + TotalStandbyCost;

// The Model

minimize TOTAL\_COST;

subject to

{

// Inventory Balancing constraints for final\_products on final\_stage

forall (j in Final\_Products, l in productionstages:l==L, t in macroperiods)

Inventory\_Balancing: {

if(t==1)

inventory[j][t-1] + sum(s in microperiods1tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + primary\_demand[j][t];

if(t==2)

inventory[j][t-1] + sum(s in microperiods2tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + primary\_demand[j][t];

if(t==3)

inventory[j][t-1] + sum(s in microperiods3tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + primary\_demand[j][t];

if(t==4)

inventory[j][t-1] + sum(s in microperiods4tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + primary\_demand[j][t];

}

// WIP Balancing constraints

forall (j in Remaining\_Products, l in productionstages:l<=L-1, t in macroperiods)

WIPInventory\_Balancing: {

if(t==1 && j in allproductsonstage1 && l==1)

inventory[j][t-1] + sum(s in microperiods1tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==2 && j in allproductsonstage1 && l==1)

inventory[j][t-1] + sum(s in microperiods2tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==3 && j in allproductsonstage1 && l==1)

inventory[j][t-1] + sum(s in microperiods3tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==4 && j in allproductsonstage1 && l==1)

inventory[j][t-1] + sum(s in microperiods4tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==1 && j in allproductsonstage2 && l==2)

inventory[j][t-1] + sum(s in microperiods1tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==2 && j in allproductsonstage2 && l==2)

inventory[j][t-1] + sum(s in microperiods2tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==3 && j in allproductsonstage2 && l==2)

inventory[j][t-1] + sum(s in microperiods3tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

if(t==4 && j in allproductsonstage2 && l==2)

inventory[j][t-1] + sum(s in microperiods4tomacroperiod) productionquantity[j][l][s]

== inventory[j][t] + BOM \* secondary\_demand[j][t];

}

//Capacity Constraints

forall (l in productionstages, t in macroperiods)

Capacity\_Stage: {

if(l==1 && t==1)

sum(j in allproductsonstage1, s in microperiods1tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage1, s in microperiods1tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods1tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==1 && t==2)

sum(j in allproductsonstage1, s in microperiods2tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage1, s in microperiods2tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods2tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==1 && t==3)

sum(j in allproductsonstage1, s in microperiods3tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage1, s in microperiods3tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods3tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==1 && t==4)

sum(j in allproductsonstage1, s in microperiods4tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage1, s in microperiods4tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods4tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==2 && t==1)

sum(j in allproductsonstage2, s in microperiods1tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage2, s in microperiods1tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods1tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==2 && t==2)

sum(j in allproductsonstage2, s in microperiods2tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage2, s in microperiods2tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods2tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==2 && t==3)

sum(j in allproductsonstage2, s in microperiods3tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage2, s in microperiods3tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods3tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==2 && t==4)

sum(j in allproductsonstage2, s in microperiods4tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage2, s in microperiods4tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods4tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==3 && t==1)

sum(j in allproductsonstage3, s in microperiods1tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage3, s in microperiods1tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods1tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==3 && t==2)

sum(j in allproductsonstage3, s in microperiods2tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage3, s in microperiods2tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods2tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==3 && t==3)

sum(j in allproductsonstage3, s in microperiods3tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage3, s in microperiods3tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods3tomacroperiod) sb[l][s] <= productstagecapacity[l];

if(l==3 && t==4)

sum(j in allproductsonstage3, s in microperiods4tomacroperiod) production\_time\*

productionquantity[j][l][s] + sum(i,j in allproductsonstage3, s in microperiods4tomacroperiod)

setuptime[j]\*Pchangeover[i][j][l][s] + sum(s in microperiods4tomacroperiod) sb[l][s] <= productstagecapacity[l];

}

//Production Flow between Stages (Sequence & Position) Constraints

forall (j in allproductsonstage1, p in allproductsonstage2, l in productionstages:l<=L-1, s in microperiods)

Position\_Sequence1:{

if (j in family1stage1 && p in family1stage2 && l==1)

BigM \* (stagesetup[j][l][s]-1) + sb[l][s] + E\_setuptime[l][s]<= BigM\*(1-stagesetup[p][l+1][s])+

sb[l+1][s] + E\_setuptime[l+1][s];

if (j in family2stage1 && p in family2stage2 && l==1)

BigM \* (stagesetup[j][l][s]-1) + sb[l][s] + E\_setuptime[l][s]<= BigM\*(1-stagesetup[p][l+1][s])+

sb[l+1][s] + E\_setuptime[l+1][s];

if (j in family3stage1 && p in family3stage2 && l==1)

BigM \* (stagesetup[j][l][s]-1) + sb[l][s] + E\_setuptime[l][s]<= BigM\*(1-stagesetup[p][l+1][s])+

sb[l+1][s] + E\_setuptime[l+1][s];

if (j in family1stage2 && p in family1stage3 && l==2)

BigM \* (stagesetup[j][l][s]-1) + sb[l][s] + E\_setuptime[l][s]<= BigM\*(1-stagesetup[p][l+1][s])+

sb[l+1][s] + E\_setuptime[l+1][s];

if (j in family2stage2 && p in family1stage3 && l==2)

BigM \* (stagesetup[j][l][s]-1) + sb[l][s] + E\_setuptime[l][s]<= BigM\*(1-stagesetup[p][l+1][s])+

sb[l+1][s] + E\_setuptime[l+1][s];

if (j in family3stage2 && p in family1stage3 && l==2)

BigM \* (stagesetup[j][l][s]-1) + sb[l][s] + E\_setuptime[l][s]<= BigM\*(1-stagesetup[p][l+1][s])+

sb[l+1][s] + E\_setuptime[l+1][s];

}

forall (j in allproductsonstage1, p in allproductsonstage2, l in productionstages:l<=L-1, s in microperiods)

Position\_Sequence2:{

if (j in family1stage1 && p in family1stage2 && l==1)

BigM \* (stagesetup[j][l][s]-1) + B\_setuptime[l][s] + production\_time\*productionquantity[j][l][s]

<= BigM\*(1-stagesetup[p][l+1][s])+ B\_setuptime[l+1][s] + production\_time\*productionquantity[p][l+1][s];

if (j in family2stage1 && p in family2stage2 && l==1)

BigM \* (stagesetup[j][l][s]-1) + B\_setuptime[l][s] + production\_time\*productionquantity[j][l][s]

<= BigM\*(1-stagesetup[p][l+1][s])+ B\_setuptime[l+1][s] + production\_time\*productionquantity[p][l+1][s];

if (j in family3stage1 && p in family3stage2 && l==1)

BigM \* (stagesetup[j][l][s]-1) + B\_setuptime[l][s] + production\_time\*productionquantity[j][l][s]

<= BigM\*(1-stagesetup[p][l+1][s])+ B\_setuptime[l+1][s] + production\_time\*productionquantity[p][l+1][s];

if (j in family1stage2 && p in family1stage3 && l==2)

BigM \* (stagesetup[j][l][s]-1) + B\_setuptime[l][s] + production\_time\*productionquantity[j][l][s]

<= BigM\*(1-stagesetup[p][l+1][s])+ B\_setuptime[l+1][s] + production\_time\*productionquantity[p][l+1][s];

if (j in family2stage2 && p in family1stage3 && l==2)

BigM \* (stagesetup[j][l][s]-1) + B\_setuptime[l][s] + production\_time\*productionquantity[j][l][s]

<= BigM\*(1-stagesetup[p][l+1][s])+ B\_setuptime[l+1][s] + production\_time\*productionquantity[p][l+1][s];

if (j in family3stage2 && p in family1stage3 && l==2)

BigM \* (stagesetup[j][l][s]-1) + B\_setuptime[l][s] + production\_time\*productionquantity[j][l][s]

<= BigM\*(1-stagesetup[p][l+1][s])+ B\_setuptime[l+1][s] + production\_time\*productionquantity[p][l+1][s];

}

//Upper bound on production quantities

forall (j in products, l in productionstages, s in microperiods)

UB\_ProductionQTY:{

if(j in allproductsonstage1 && l==1)

productionquantity[j][l][s] <= (productstagecapacity[l]/production\_time) \* stagesetup[j][l][s];

if(j in allproductsonstage2 && l==2)

productionquantity[j][l][s] <= (productstagecapacity[l]/production\_time) \* stagesetup[j][l][s];

if(j in allproductsonstage3 && l==3)

productionquantity[j][l][s] <= (productstagecapacity[l]/production\_time) \* stagesetup[j][l][s];

}

//Lower bound on production quantities - Minimum Lot-size needed / Triangle inequality not always true

forall (j in products, l in productionstages, s in microperiods)

min\_litsizes:{

if(s==1){

if(j in allproductsonstage1 && l==1)

productionquantity[j][l][s] >= min\_lotsize\* (stagesetup[j][l][s]);

if(j in allproductsonstage2 && l==2)

productionquantity[j][l][s] >= min\_lotsize\* (stagesetup[j][l][s]);

if(j in allproductsonstage3 && l==3)

productionquantity[j][l][s] >= min\_lotsize\* (stagesetup[j][l][s]);

}

if(s>1){

if(j in allproductsonstage1 && l==1)

productionquantity[j][l][s] >= min\_lotsize\* (stagesetup[j][l][s]-stagesetup[j][l][s-1]);

if(j in allproductsonstage2 && l==2)

productionquantity[j][l][s] >= min\_lotsize\* (stagesetup[j][l][s]-stagesetup[j][l][s-1]);

if(j in allproductsonstage3 && l==3)

productionquantity[j][l][s] >= min\_lotsize\* (stagesetup[j][l][s]-stagesetup[j][l][s-1]);

}

}

//Only one production stage setup allowed in each microperiod

forall (l in productionstages, s in microperiods)

Onlyone\_Setup1: {

if(l==1)

sum(j in allproductsonstage1)

stagesetup[j][l][s]==1;

if(l==2)

sum(j in allproductsonstage2)

stagesetup[j][l][s]==1;

if(l==3)

sum(j in allproductsonstage3)

stagesetup[j][l][s]==1;

}

//Only one product changeover allowed in each microperiod

forall (l in productionstages, s in microperiods:s>=2)

Onlyone\_Changeover: {

if (l==1)

sum (i,j in allproductsonstage1)

Pchangeover[i][j][l][s]==1;

if (l==2)

sum (i,j in allproductsonstage2)

Pchangeover[i][j][l][s]==1;

if (l==3)

sum (i,j in allproductsonstage3)

Pchangeover[i][j][l][s]==1;

}

//Setup Spliting idea constrinats

forall (l in productionstages, s in microperiods:s>=2)

Setup\_Splitting:

{

if (l==1)

E\_setuptime[l][s-1] + B\_setuptime[l][s] ==

sum (i,j in allproductsonstage1) setuptime[j]\*Pchangeover[i][j][l][s];

if (l==2)

E\_setuptime[l][s-1] + B\_setuptime[l][s] ==

sum (i,j in allproductsonstage2) setuptime[j]\*Pchangeover[i][j][l][s];

if (l==3)

E\_setuptime[l][s-1] + B\_setuptime[l][s] ==

sum (i,j in allproductsonstage3) setuptime[j]\*Pchangeover[i][j][l][s];

}

// Linking between product changeover and machine setup constrinats

forall (i,j in products, l in productionstages, s in microperiods:s>=2)

Changeover\_setup:

{

if(l==1)

Pchangeover[i][j][l][s] >= stagesetup[i][l][s-1]+ stagesetup[j][l][s]-1;

if(l==2)

Pchangeover[i][j][l][s] >= stagesetup[i][l][s-1]+ stagesetup[j][l][s]-1;

if(l==3)

Pchangeover[i][j][l][s] >= stagesetup[i][l][s-1]+ stagesetup[j][l][s]-1;

}

}