#include <iostream>

#include <fstream>

#include <vector>

#include <time.h>

#include "ilcplex/ilocplex.h"

using namespace std;

typedef IloArray<IloNumVarArray> NumVar2D;

typedef IloArray<NumVar2D> NumVar3D;

typedef IloArray<NumVar3D> NumVar4D;

int main()

{

clock\_t start = 0, end = 0;

start = clock();

#pragma region get data

int C = 4; // Classes

int T = 4; // Teachers

int R = 4; // rooms

int D = 5; // days per week

int P = 6; // periods per day

ifstream get\_dim;

get\_dim.open("dim.txt");

if (get\_dim.is\_open()) {

get\_dim >> C >> T >> R >> D >> P;

get\_dim.close();

}

else {

cout << "couldn't find dim.txt file\n";

return 0;

}

int slots = P \* D;

vector<vector<int>>M(C \* R, vector<int>(T, 0));

ifstream read;

read.open("matrix.txt");

if (read.is\_open())

{

cout << "matrix.txt file found!\n";

for (int i = 0; i < C \* R; i++)

for (int j = 0; j < T; j++)

read >> M[i][j];

read.close();

}

else

{

cout << "couldn't find matrix.txt file\n";

return 0;

}

/\*for (int i = 0; i < C \* R; i++){

for (int j = 0; j < T; j++)

cout << M[i][j] << " ";

cout << endl;

}\*/

#pragma endregion

IloEnv env;

IloModel model(env);

NumVar4D X(env, C);

for (int i = 0; i < C; i++) {

X[i] = NumVar3D(env, T);

for (int j = 0; j < T; j++) {

X[i][j] = NumVar2D(env, slots);

for (int k = 0; k < slots; k++) {

X[i][j][k] = IloNumVarArray(env, R, 0, 1, ILOINT); // 0-1 Integer Decision Variable

}

}

}

IloRangeArray cons1(env); // Matching the required defined in matrix.txt

for (int i = 0; i < C; i++) {

for (int j = 0; j < T; j++) {

for (int l = 0; l < R; l++) {

IloExpr exp(env);

for (int k = 0; k < slots; k++) {

exp += X[i][j][k][l];

}

int tmp = C \* l + i;

cons1.add(exp == M[tmp][j]);

exp.end();

}

}

}

model.add(cons1);

IloRangeArray cons2(env); // only one class in kth slot and lth room

for (int k = 0; k < slots; k++) {

for (int l = 0; l < R; l++) {

IloExpr exp(env);

for (int i = 0; i < C; i++) {

for (int j = 0; j < T; j++) {

exp += X[i][j][k][l];

}

}

cons2.add(exp <= 1);

exp.end();

}

}

model.add(cons2);

IloRangeArray cons3(env); // one teacher in a fixed slot can take only one class

for (int j = 0; j < T; j++) {

for (int k = 0; k < slots; k++) {

IloExpr exp(env);

for (int i = 0; i < C; i++) {

for (int l = 0; l < R; l++) {

exp += X[i][j][k][l];

}

}

cons3.add(exp <= 1);

exp.end();

}

}

model.add(cons3);

IloRangeArray cons4(env); // for one slot only one class can happen

for (int i = 0; i < C; i++) {

for (int k = 0; k < slots; k++) {

IloExpr exp(env);

for (int j = 0; j < T; j++) {

for (int l = 0; l < R; l++) {

exp += X[i][j][k][l];

}

}

cons4.add(exp <= 1);

exp.end();

}

}

model.add(cons4);

IloExpr objfn(env);

objfn += 1;

//Feasibility problem so objective function doesn't really matter.

/\*for (int i = 0; i < C; i++) {

for (int j = 0; j < T; j++) {

for (int k = 0; k < slots; k++) {

for (int l = 0; l < R; l++) {

objfn += X[i][j][k][l]; This was our old objective function.

}

}

}

}\*/

model.add(IloMinimize(env, objfn)); //objective function added to model

objfn.end();

IloCplex cplex(model);

cplex.setOut(env.getNullStream()); //calculation steps not shown on command prompt

cplex.solve();

vector<vector<pair<int,int>>>ans(R, vector<pair<int,int>>(slots));

//ans[l][k] stores two integers {i, j}, which means during kth slot in the lth room jth teacher teaches ith class

for (int i = 0; i < C; i++) {

for (int j = 0; j < T; j++) {

for (int k = 0; k < slots; k++) {

for (int l = 0; l < R; l++) {

int val = cplex.getValue(X[i][j][k][l]);

if (val == 1) {

ans[l][k] = { i, j };

}

}

}

}

}

end = clock();

double time\_taken = double(end - start) / double(CLOCKS\_PER\_SEC);

cout << "Time taken by the program is : " << fixed << time\_taken << setprecision(5) << " seconds" << endl;

ofstream out("output.txt");

for (int l = 0; l < R; l++) {

for (int k = 0; k < slots; k++) {

cout << ans[l][k].first << " " << ans[l][k].second << " |";

out << ans[l][k].first << " " << ans[l][k].second << " |";

}

cout << endl;

out << endl;

}

out.close();

env.end();

}