```python
#############################################################################
# File Name: model27_cplex_Q1.py                                            #
# Author: Geonsik Yu, Purdue University, IE Dept                            #
# LP problem (Model 27: Hydrological Model) from:                           #
# https://sites.math.washington.edu/~burke/crs/407/models/m27.html          #
#############################################################################
import cplex

def oneHot(length, hotIdx):
    hotVec = [0.0]*length
    hotVec[hotIdx] = 1.0
    return hotVec

## STEP 1. Set up what we need. ----------------------------------------------------------
## Declare small constant epsilon for strict inequality removal:
EPSILON = 0.000000000001
## Declare variable names:
variables = ["b0", "b1", "b2", "A3", "A4", "A5", "A6", "A7", "A8", "A9", "A10", "A11", "A12"]
## Delare a list of coefficients  of each variable in the objective function (same order):
obj_coeffs = 3*[0.0] + 10*[1.0]
## Delare a list of upperbounds of each variable:
upperbounds = 13*[cplex.infinity]
## Delare a list of lowerbounds of each variable:
#lowerbounds = 3*[EPSILON] + 10*[-cplex.infinity]
lowerbounds = 3*[EPSILON] + 10*[0]

## Declare contraint names:
constraint_names = ["Period 3-(1)", "Period 3-(2)",
                                    "Period 4-(1)", "Period 4-(2)",
                                    "Period 5-(1)", "Period 5-(2)",
                                    "Period 6-(1)", "Period 6-(2)",
                                    "Period 7-(1)", "Period 7-(2)",
                                    "Period 8-(1)", "Period 8-(2)",
                                    "Period 9-(1)", "Period 9-(2)",
                                    "Period 10-(1)", "Period 10-(2)",
                                    "Period 11-(1)", "Period 11-(2)",
                                    "Period 12-(1)", "Period 12-(2)",
                                    "b2", "b1 - b2", "b0 - b1",
                                    "b0+b1+b2"]
## Declare a list of RHS constants of each constraints:
righthand = [1.0, -1.0, 2.1, -2.1, 3.7, -3.7, 4.2, -4.2, 4.3, -4.3,
                     4.4, -4.4, 4.3, -4.3, 4.2, -4.2, 3.6, -3.6, 2.7, -2.7,
                     EPSILON, EPSILON, EPSILON, 1.0]

## Declare a list of inequality directions of each constraints:
senses = 23*['G'] + ['E']

## Declare and complete a coefficient matrix for the constraints:
Mat = []
Precip = [3.8, 4.4, 5.7, 5.2, 7.7, 6.0, 5.4, 5.7, 5.5, 2.5, 0.8, 0.4]
for i in range(2, 12):
    tmp1 = [Precip[i], Precip[i-1], Precip[i-2]] + oneHot(length=10, hotIdx=(i-2))
    tmp2 = [-Precip[i], -Precip[i-1], -Precip[i-2]] + oneHot(length=10, hotIdx=(i-2))
    Mat.append(tmp1)
    Mat.append(tmp2)
Mat.append([0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
Mat.append([0.0, 1.0,-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
Mat.append([1.0,-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
Mat.append([1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])

## Set coefficients of each variables in each constraints:
lin_expr = []
for row in Mat:
    print(row)
    lin_expr.append( cplex.SparsePair(ind=variables, val=row) )

## STEP 2. Generate LP problem object -----------------------------------------------------
## Generate an LP problem
problem = cplex.Cplex()
## Set objective as minimization
problem.objective.set_sense( problem.objective.sense.minimize )
## Set variables and objective function
problem.variables.add( obj=obj_coeffs, ub=upperbounds, lb=lowerbounds, names=variables )
## Set constraints
problem.linear_constraints.add(lin_expr = lin_expr, senses = senses, rhs = righthand, names = constraint_names)
## Solve the problem
problem.solve()

## STEP 3. Print out results ------------------------------------------------------------
numrows = problem.linear_constraints.get_num()
numcols = problem.variables.get_num()

print("Solution status = "+ repr(problem.solution.get_status())+ ": " +repr(problem.solution.status[problem.solution.get_status()]))
print("Solution value  = "+ repr(problem.solution.get_objective_value()))

x = problem.solution.get_values()
shadow_price = problem.solution.get_dual_values()
for i in range(numcols):
     print("Variable " + variables[i] + ": Value = " + repr(x[i]))
for i in range(numrows):
     print("Constraint " + constraint_names[i] + ": Shadow Price = " + repr(shadow_price[i]))
```