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

 \* OPL 12.8.0.0 Model

 \* Author: rodrigo rosa

 \* Creation Date: 10/05/2019 at 13:35:34

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

//numero de clientes

int n = ...;

//numero de veiculos

int k = ...;

//capacidade do veiculo

int q = ...;

//conjunto de clientes

range C = 1 .. n;

//conjunto de clientes mais depositos

range N = 0 .. n+1;

//conjunto de veiculos

range K = 1 .. k;

//matriz de distancia entre nos

float c[i in N][j in N] = ...;

//vetor de demanda

float d[i in C] = ...;

//variavel de decisao-se = 0 entao nao e pecorrido o arco i-j pelo veiculo k,

//se = 1 entao o veiculo k percorre o arco i-j

dvar int x[i in N][j in N][k in K];

//funcao objetivo

minimize sum(k in K, i in N, j in N) (c[i][j] \* x[i][j][k]);

subject to {

forall (k in K) {

 const01: sum(j in N) x[0][j][k] == 1;}

forall (i in C) {

 const02: sum(k in K, j in N) x[i][j][k] == 1;}

forall (k in K) {

 const03: sum(i in C, j in N) (d[i] \* x[i][j][k]) <= q; }

forall (k in K, h in C) {

 const04: sum(i in N) x[i][h][k] - sum(j in N) x[h][j][k] == 0;}

forall (k in K) {

 const05: sum(i in N) x[i][n+1][k] == 1;}

forall (i in N, j in N, k in K) {

 const06: 0 <= x[i][j][k] <= 1;}

}

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Aqui comeca o dat

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

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

 \* OPL 12.8.0.0 Data

 \* Author: rodrigo rosa

 \* Creation Date: 10/05/2019 at 13:35:34

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

//numero de clientes (int)

n = 9;

//numero de veiculos da frota(int)

k = 2;

// capacidade do veiculo (int)

q = 60;

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

//DEFINIÇÃO DOS PARÂMETROS

// distância (ou custo) do arco ij (int)

 c = [

 [9999.99 38.47 55.95 18.03 49.09 51.48 38.21 38.42 52.39 14.87 0.00]

 [9999.99 9999.99 31.78 40.80 45.54 89.90 62.51 7.21 85.91 53.23 38.47]

 [9999.99 9999.99 9999.99 66.22 77.28 101.86 59.62 25.18 108.23 67.80 55.95]

 [9999.99 9999.99 9999.99 9999.99 33.24 57.31 55.90 43.83 45.28 26.00 18.03]

 [9999.99 9999.99 9999.99 9999.99 9999.99 88.20 87.01 52.17 65.46 59.24 49.09]

 [9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 50.70 89.19 37.48 36.67 51.48]

 [9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 58.03 74.33 34.01 38.21]

 [9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 88.23 52.70 38.42]

 [9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 43.57 52.39]

 [9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 14.87]

 [9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99 9999.99]

 ];

// demanda de entrega

d = [11 35 2 9 3 18 8 10 11];