In [20]:

```
##Problem 1
using JuMP, NamedArrays, Clp

A =
 [ 0 0 1 1 0 0 0 1 1 0 0 0 0
   0 1 1 0 0 0 0 0 1 1 0 0 0
   0 0 0 1 1 0 1 1 0 1 1 1 1
   0 0 0 1 1 1 1 1 1 1 1 1 0
   0 0 0 0 0 0 1 1 1 0 0 0 0
   0 1 1 0 0 0 0 0 1 1 0 0 0
   0 0 0 1 1 1 1 0 0 0 0 0 0
   1 1 0 0 0 0 0 0 0 0 1 1 1
   1 1 1 0 0 0 0 0 0 1 1 0 0
   0 0 0 0 0 0 0 1 1 0 0 0 0
   0 0 0 0 0 0 1 1 1 0 0 0 0
   1 1 0 0 0 1 1 1 1 0 0 1 1
   1 1 1 0 1 1 0 0 0 0 0 1 1
   0 1 1 1 0 0 0 0 0 0 0 0 0
   1 1 0 0 1 1 0 0 0 0 0 0 0 ]

TIMES = ["10:00","10:20","10:40","11:00","11:20","11:40","lunch","1:00","1:20","1:40","2:00","2:20","2:40"]
NAMES =
[:Manuel,:Luca,:Jule,:Michael,:Malte,:Chris,:Spyros,:Mirjam,:Matt,:Florian,:Josep,:Joel,:Tom,:Daniel,:Anne ]
times = NamedArray( availability, (NAMES,TIMES), ("NAME","TIME"))
slots = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 ]

m = Model(solver = ClpSolver())
@variable(m, 0 <= x[1:15,1:13] <= 1)
for i = 1:15
   for j = 1:13
      if A[i,j] == 0
      @constraint(m, x[i,j] == 0)
   end
end
 @constraint(m, sum(x[i,k] for k = 1:13) == 1)
end
for b = 1:13
   if b == 7
   @constraint(m, sum(x[a,b] for a = 1:15) == 3)
   else
   @constraint(m, sum(x[a,b] for a = 1:15) <= 1)
   end
end
@objective(m, Min, sum(x))
solve(m)

schedule = NamedArray([Int(getvalue(x[i,j])) for i = 1:15, j = 1:13],
(NAMES, slots), ("Name","Slot"))
```

15×13 Named Array{Int64,2}

Name \ Slot	1	2	3	4	5	6	7	8	9	10	11	12	13
Manuel	0	0	0	0	0	0	0	1	0	0	0	0	0
Luca	0	0	0	0	0	0	0	0	0	1	0	0	0
Jule	0	0	0	0	0	0	0	0	0	0	0	0	1
Michael	0	0	0	0	0	0	0	0	0	0	0	1	0
Malte	0	0	0	0	0	0	1	0	0	0	0	0	0
Chris	0	1	0	0	0	0	0	0	0	0	0	0	0
Spyros	0	0	0	0	0	0	1	0	0	0	0	0	0
Mirjam	0	0	0	0	0	0	0	0	0	0	1	0	0
Matt	1	0	0	0	0	0	0	0	0	0	0	0	0
Florian	0	0	0	0	0	0	0	0	1	0	0	0	0
Josep	0	0	0	0	0	0	1	0	0	0	0	0	0
Joel	0	0	1	0	0	0	0	0	0	0	0	0	0
Tom	0	0	0	0	1	0	0	0	0	0	0	0	0
Daniel	0	0	0	1	0	0	0	0	0	0	0	0	0
Anne	0	0	0	0	0	1	0	0	0	0	0	0	0

In [2]:

```
##Problem 2
using JuMP, NamedArrays, Clp

cords = [0 20 18 30 35 33 5 5 11 2;
        0 20 10 12 0 25 27 10 0 15]

#find distance between agencies
cost = Matrix(10,10)
for i in 1:10
   for j in 1:10
      cost[i,j] = 0.5*1.3*sqrt((cords[1,i]-cords[1,j])^2
      + (cords[2,i]-cords[2,j])^2)
   end
end

demand = [ 10 6 8 11 9 7 15 7 9 12 ]
supply = [ 8 13 4 8 12 2 14 11 15 7 ]

m = Model(solver = ClpSolver())
@variable(m, x[1:10,1:10] >= 0)

for i in 1:10
   if demand[i] < supply[i]
      @constraint(m, sum(x[i,j] for j in 1:10) == supply[i])
   else
      @constraint(m, sum(x[j,i] for j in 1:10) == demand[i])
```

```
    end
    @constraint(m, supply[i] + sum(x[j,i] for j in 1:10) - sum(x[i,j]
          for j in 1:10) == demand[i])
end

@objective(m, Min, sum(x[i,j]*cost[i,j] for i in 1:10, j in 1:10))
solve(m)

agency = [ 1 2 3 4 5 6 7 8 9 10]
trans = NamedArray(Int[getvalue(x[i,j]) for i in 1:10, j in 1:10],
                (agency,agency), ("agency","agency"))
println(getobjectivevalue(m))
println(trans)
```

```
152.63901632295628
10×10 Named Array{Int64,2}
agency ╲ agency │  1   2   3   4   5   6   7   8   9  10
────────────────┼──────────────────────────────────────────
1               │  8   0   0   0   0   0   0   0   0   0
2               │  0   6   1   0   0   5   1   0   0   0
3               │  0   0   4   0   0   0   0   0   0   0
4               │  0   0   0   8   0   0   0   0   0   0
5               │  0   0   0   3   9   0   0   0   0   0
6               │  0   0   0   0   0   2   0   0   0   0
7               │  0   0   0   0   0   0  14   0   0   0
8               │  0   0   0   0   0   0   0   6   0   5
9               │  2   0   3   0   0   0   0   1   9   0
10              │  0   0   0   0   0   0   0   0   0   7
```

In [43]:

```
##Problem 3a

using JuMP, Clp

tasks = 1:18
durations = [2 16 9 8 10 6 2 2 9 5 3 2 1 7 4 3 9 1]
duration = Dict(zip(tasks,durations))
predecessors = ( [], [1], [2], [2], [3], [4,5], [4], [6], [4,6], [4], [6], [9], [7], [2], [4,14], [8,11,14], [12], [17] )
pred_dict = Dict(zip(tasks,predecessors));   # dictionary mapping tasks --> predecessors.
pred = Dict(zip(tasks,predecessors))

m = Model(solver = ClpSolver())
@variable(m, tstart[tasks] >= 0 )

@constraint(m, link[i in tasks, j in pred[i]],
tstart[i] >= tstart[j] + duration[j])

for i in 1:18
```

```
    for j in pred_dict[i]
    @constraint(m, tstart[i] >= tstart[j] + duration[j])
    end
end
@objective(m, Min, tstart[18] + duration[18])
solve(m)
println(getvalue(tstart))
println("Earliest week of completion: ",getvalue(tstart[18])+ duration[18])


##Problem 3b

max_reduction =  [0,  3,  1,  2,  2,  1, 1, 0,  2,  1,  1, 0, 0,  2,  2, 1,  3, 0]  # max reduction (in weeks)
cost_reduction = [0, 30, 26, 12, 17, 15, 8, 0, 42, 21, 18, 0, 0, 22, 12, 6, 16, 0]  # cost of reduction
($1,000/week)
bonus_amount = 30     # bonus for expediting the project ($1,000/week )
cost_reduction = cost_reduction*1000
base = getvalue(tstart[18]) + duration[18]

m = Model(solver = ClpSolver())
# x = weeks to reduce
@variable(m, x[1:18] >= 0)
@constraint(m, a[i in 1:18], x[i] <= max_reduction[i])
@objective(m, Max, sum(x[i]*30000 for i in 1:18)
- sum(x[i]*cost_reduction[i] for i in 1:18))
solve(m)

println()
println("Optial Finish Week: ", base - getvalue(sum(x)))
println("Max Profit: \$", getobjectivevalue(m))

tstart: 1 dimensions:
[ 1] = 0.0
[ 2] = 2.0
[ 3] = 18.0
[ 4] = 18.0
[ 5] = 27.0
[ 6] = 37.0
[ 7] = 26.0
[ 8] = 43.0
[ 9] = 43.0
[10] = 26.0
[11] = 43.0
[12] = 52.0
[13] = 28.0
[14] = 18.0
[15] = 26.0
[16] = 46.0
[17] = 54.0
[18] = 63.0
Earliest week of completion: 64.0

Optial Finish Week: 47.0
Max Profit: $242000.0
```