```
# Importing libraries
using JuMP
using HiGHS

# Create a model with HiGHS as the solver
model = Model(HiGHS.Optimizer)

# Decision variables representing salaries
@variable(model, Tom >= 35000)  # Tom salary
@variable(model, Peter >= 0)  # Peter salary
@variable(model, Nina >= 0)  # Nina salary
@variable(model, Samir >= 0)  # Samir salary
@variable(model, Gary >= 0)  # Gary salary
@variable(model, Linda >= 0)  # Linda salary
@variable(model, Bob >= 0)  # Bob salary

# Variable to represent the maximum salary
@variable(model, MaxSalary)

# Objective is minimizeing the highest salary
@objective(model, Min, MaxSalary)

# Adding the constraint that MaxSalary should be greater than or equal
to each employee's salary
@constraint(model, Tom <= MaxSalary)
@constraint(model, Peter <= MaxSalary)
@constraint(model, Nina <= MaxSalary)
@constraint(model, Samir <= MaxSalary)
@constraint(model, Gary <= MaxSalary)
@constraint(model, Linda <= MaxSalary)
@constraint(model, Bob <= MaxSalary)

# Adding the basic constraints

@constraint(model, Peter >= Tom + 8500)  # Peters salary should be at
least 8500 more than Toms
@constraint(model, Nina >= Tom + 8500)  # Ninas salary should be at
least 8500 more than Toms
@constraint(model, Samir >= Tom + 8500)  # Samirs salary should be at
least 8500 more than Toms
@constraint(model, Gary >= Tom + Peter)  # Garys salary should be at
least Toms + Peters combined
@constraint(model, Linda == Gary + 1000)  # Lindas salary should be
$1000 more than Garys
@constraint(model, Nina + Samir >= 2 * (Tom + Peter))  # Nina and
Samir combined should earn at least twice Tom and Peterss combined
salary
@constraint(model, Bob >= Peter)  # Bobs salary should be at least as
much as Peters
@constraint(model, Bob >= Samir)  # Bobs salary should be at least as
```

```
much as Samirs
@constraint(model, Bob + Peter >= 75000)  # Bob and Peter combined
should earn at least $75000
@constraint(model, Linda <= Bob + Tom)  # Linda shouldnt earn more
than the combined salaries of Bob and Tom

# Solve
optimize!(model)

# Results
println("Optimal Salaries:")
println("Tom's salary: ", value(Tom))
println("Peter's salary: ", value(Peter))
println("Nina's salary: ", value(Nina))
println("Samir's salary: ", value(Samir))
println("Gary's salary: ", value(Gary))
println("Linda's salary: ", value(Linda))
println("Bob's salary: ", value(Bob))
println("Maximum salary: ", value(MaxSalary))
```

```
Running HiGHS 1.7.2 (git hash: 5ce7a2753): Copyright (c) 2024 HiGHS
under MIT licence terms
Coefficient ranges:
  Matrix [1e+00, 2e+00]
  Cost   [1e+00, 1e+00]
  Bound  [4e+04, 4e+04]
  RHS    [1e+03, 8e+04]
Presolving model
16 rows, 7 cols, 36 nonzeros  0s
13 rows, 4 cols, 30 nonzeros  0s
2 rows, 2 cols, 4 nonzeros  0s
2 rows, 2 cols, 4 nonzeros  0s
Presolve : Reductions: rows 2(-15); columns 2(-6); elements 4(-34)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     7.9500147039e+04 Pr: 1(34000) 0s
          1     7.9500000000e+04 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 1
Objective value     :  7.9500000000e+04
HiGHS run time      :          0.00
Optimal Salaries:
Tom's salary: 35000.0
Peter's salary: 43500.0
Nina's salary: 79500.0
Samir's salary: 77500.0
Gary's salary: 78500.0
Linda's salary: 79500.0
```

```
Bob's salary: 79500.0
Maximum salary: 79500.0
```