Question 1

The director of Awesome Tech startup needs to decide what salaries to offer its employees for the coming year. In order to keep the employees satisfied, she needs to satisfy the following constraints:

- Tom wants at least $30,000 or he will quit
- Peter, Nina, and Samir each want to be paid at least $8000 more than Tom
- Gary wants his salary to be at least as high as the combined salary of Tom and Perry
- Linda wants her salary to be $500 more than Gary
- The combined salary of Nina and Samir should be at least twice the combined salary of Tom and Peter
- Bob's salary is at least as high as that of Peter and at least as high as that of Samir
- The combined salary of Bob and Peter should be at least $75,000
- Linda should not make more money than the combined salary of Bob and Tom

Suppose that we divide the employees into two groups:

- IT professionals: This group consists of Tom, Peter, Nina, and Samir
- Customer Service representatives: This is the rest of the employees

Instead of minimizing the original objective, we instead want to minimize:

- the largest IT professional salary + the largest customer service representative salary

The employees demands for salaries must still be satisfied. Note that this problem is a min-max problem and should be reformulated using the epigraph form discussed in class.

In [ ]:
```
using JuMP, HiGHS

# defining model
salaries = Model(HiGHS.Optimizer)
```

Out[ ]:
```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: HiGHS
```

In [ ]:
```
# variables for each person's salaries
@variable(salaries, Tom >= 0)
@variable(salaries, Peter >= 0)
@variable(salaries, Nina >= 0)
@variable(salaries, Samir >= 0)
@variable(salaries, Linda >= 0)
@variable(salaries, Gary >= 0)
@variable(salaries, Bob >= 0)

# variables for min-max problem
@variable(salaries, highest_IT >= 0)
@variable(salaries, highest_CustomerService >= 0)
```

Out[ ]:

$$highest\_CustomerService$$

In [ ]:
```
# constraints for the model
@constraint(salaries, TomC, Tom >= 30000)
@constraint(salaries, PeterC, Peter >= 8000 + Tom)
@constraint(salaries, NinaC, Nina >= 8000 + Tom)
@constraint(salaries, SamirC, Samir >= 8000 + Tom)
@constraint(salaries, GaryC, Gary >= Peter + Tom)
@constraint(salaries, LindaC, Linda == 500 + Gary)
@constraint(salaries, NinaSamirC, Nina + Samir >= 2(Peter + Tom))
@constraint(salaries, BobC, Bob >= Peter)
@constraint(salaries, BobC2, Bob >= Samir)
@constraint(salaries, BobPeterC, Bob + Peter >= 75000)
@constraint(salaries, LindaC2, Linda <= Bob + Tom)

@constraint(salaries, highest_ITC, highest_IT >= Tom)
@constraint(salaries, highest_IT2, highest_IT >= Peter)
@constraint(salaries, highest_ITC3, highest_IT >= Nina)
@constraint(salaries, highest_ITC4, highest_IT >= Samir)
@constraint(salaries, highest_CustomerServiceC, highest_CustomerService >= Li
@constraint(salaries, highest_CustomerServiceC2, highest_CustomerService >= B
@constraint(salaries, highest_CustomerServiceC3, highest_CustomerService >= G
```

Out[ ]: highest_CustomerServiceC3 : $-Gary + highest\_CustomerService \geq 0.0$

In [ ]:
```
# formulate the objective function
@objective(salaries, Min, highest_CustomerService + highest_IT)
```

Out[ ]:

$$highest\_CustomerService + highest\_IT$$

In [ ]:
```
# solve the optimization problem
optimize!(salaries);

# outputs detailed information about the solution process
@show solution_summary(salaries);
```

```
Presolving model
16 rows, 8 cols, 36 nonzeros
13 rows, 5 cols, 30 nonzeros
6 rows, 4 cols, 13 nonzeros
3 rows, 3 cols, 6 nonzeros
Presolve : Reductions: rows 3(-15); columns 3(-6); elements 6(-33)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     1.0650020661e+05 Pr: 1(60000) 0s
          2     1.3650000000e+05 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 2
Objective value     :  1.3650000000e+05
HiGHS run time      :          0.02
solution_summary(salaries) = * Solver : HiGHS

* Status
  Termination status : OPTIMAL
  Primal status      : FEASIBLE_POINT
  Dual status        : FEASIBLE_POINT
  Message from the solver:
  "kHighsModelStatusOptimal"

* Candidate solution
  Objective value      : 1.36500e+05
  Objective bound      : 0.00000e+00
  Relative gap         : Inf
  Dual objective value : 1.36500e+05

* Work counters
  Solve time (sec)   : 1.66457e-02
  Simplex iterations : 2
  Barrier iterations : 0
  Node count         : -1
```

In [ ]: