In [1]:

```
using JuMP, Gurobi
```

Question 1: Thrift store

How should you make change for 99 cents if the goal is to minimize the total weight of the coins used? The following table shows the weight of each type of coin. You may use any number of each type of coin.

Type of Coin	Penny	Nickel	Dime	Quarter
Weights	2.500	5.00	2.268	5.670

In [2]:

```
m1 = Model(solver=GurobiSolver(OutputFlag=0))

# Defining 4 variables (which are integer)
@variable(m1, z[1:4]>=0, Int)

# Total should be 99 cents so the sum of all type of coins should be 99 cents
@constraint(m1, z[1] + 5z[2] + 10z[3] + 25z[4] == 99)

# Objective function is to minimize the total weight of the coins used
@objective(m1, Min, 2.5z[1] + 5z[2] + 2.268z[3] + 5.670z[4])

solve(m1)
```

Out[2]:

```
:Optimal
```

In [3]:

```
zopt = getvalue(z)
println(zopt[1], " Pennies")
println(zopt[2], " Nickel")
println(zopt[3], " Dimes" )
println(zopt[4], " Quarters" )

println("Objective Value (Min weight): ", getobjectivevalue(m1))
```

```
4.0 Pennies
0.0 Nickel
7.0 Dimes
1.0 Quarters
Objective Value (Min weight): 31.546
```

Question 2: Comquat Computers.

Comquat owns four production plants at which personal computers are produced. Comquat can sell up to 20,000 computers per year at a price of $3,500 per computer. For each plant the production capacity, cost per computer, and fixed cost of operating the plant for a year are given below. Determine how Comquat can maximize its yearly profit from computer production.

Plant	Production capacity	Plant fixed cost (Million dollars)	Cost per computer (in dollars)
1	10,000	9	1,000
2	8,000	5	1,700
3	9,000	3	2,300
4	6,000	1	2,900

In [4]:

```
m2 = Model(solver=GurobiSolver(OutputFlag=0))

# Four variables to get each plants production
@variable(m2, x[1:4] >= 0, Int)
# Binary variables to know if zi plant did any production or not
@variable(m2, z[1:4], Bin)

# Constraints as per given input (each plant cannot produce more than the produc
tion capacity given)
@constraint(m2, x[1] <= 10000)
@constraint(m2, x[2] <= 8000)
@constraint(m2, x[3] <= 9000)
@constraint(m2, x[4] <= 6000)

# Total production from each plant is upto 20,000
@constraint(m2, x[1] + x[2] + x[3] + x[4] <= 20000)  # upto 20,000 computers per
year produced in total by 4 plants

# Setting M to 30,000 (in class i remember discussing that M should be small)
@constraint(m2, x .<= 30000*z)                    # if x>0 then z=1

# Objective function is to Maximize yearly profit. Given that each computer cost
s $3500 then the profit for each plant
# will be ($3500 - Cost per computers for that plant). Hence ($3.5 - $1.0 (in th
ousand $) for Plant 1 and so on...)
# Subtracting the fixed cost of plant is necessary to calculate max. profit as i
f Plant x has done some production then
# it needs to pay the plant fixed cost.

@objective(m2, Max, 2.5x[1] + 1.8x[2] + 1.2x[3] + 0.6x[4] - 9000z[1] - 5000z[2]
- 3000z[3] - 1000z[4])

solve(m2)

xopt = getvalue(x)
println(xopt[1], " Plant1")
println(xopt[2], " Plant2")
println(Int64(xopt[3]), " Plant3" )
println(xopt[4], " Plant4" )
println()
println("\$", getobjectivevalue(m2), " of net profit (in thousand dollars)")
```

```
10000.0 Plant1
8000.0 Plant2
0 Plant3
2000.0 Plant4

$25600.0 of net profit (in thousand dollars)
```

Question 3: ABC Investments.

ABC Inc. is considering several investment options. Each option has a minimum investment required as well as a maximum investment allowed. These restrictions, along with the expected return are summarized in the following table (figures are in millions of dollars):

In [8]:

```
m3 = Model(solver=GurobiSolver(OutputFlag=0))

# Variables - how much to invest in each option 1 - 6.
@variable(m3, x[1:6] >= 0)
# 6 Binary variables to know if any amount was invested in option i where i (1-6
)
@variable(m3, z[1:6], Bin)

# Adding min and max constraints for all the options
@constraint(m3, x[1] >= 3z[1])
@constraint(m3, x[1] <= 27z[1])

@constraint(m3, x[2] >= 2z[2])
@constraint(m3, x[2] <= 12z[2])

@constraint(m3, x[3] >= 9z[3])
@constraint(m3, x[3] <= 35z[3])

@constraint(m3, x[4] >= 5z[4])
@constraint(m3, x[4] <= 15z[4])

@constraint(m3, x[5] >= 12z[5])
@constraint(m3, x[5] <= 46z[5])

@constraint(m3, x[6] >= 4z[6])
@constraint(m3, x[6] <= 18z[6])

# Total investment should not be more than 80
@constraint(m3, x[1] + x[2] + x[3] + x[4] + x[5] + x[6] <= 80)  # 80 million $ t
o invest

# Because of the high-risk nature of Option 5, company policy requires that the
total amount invested
# in Option 5 be no more that the combined amount invested in Options 2, 4 and 6
@constraint(m3, x[5] <= x[2] + x[4] + x[6])

# In addition, if an investment is made in Option 3, it is required that at leas
t a minimum investment be made in
# Option 6.
@constraint(m3, z[3] <= 4z[6])

@constraint(m3, x .<= 100*z)
```

```
# wants to maximize its total expected return on investment.

@objective(m3, Max, 0.13x[1] + 0.09x[2] + 0.17x[3] + 0.10x[4] + 0.22x[5] + 0.12x
[6])

solve(m3)

xopt = getvalue(x)
println(xopt[1], " Investment in Option 1 (in Million dollars)")
println(xopt[2], " Investment in Option 2 (in Million dollars)")
println(xopt[3], " Investment in Option 3 (in Million dollars)" )
println(xopt[4], " Investment in Option 4 (in Million dollars)" )
println(xopt[5], " Investment in Option 5 (in Million dollars)" )
println(xopt[6], " Investment in Option 6 (in Million dollars)" )
println()
println("\$", getobjectivevalue(m3), " of net profit (in Million dollars)")
```

```
0.0 Investment in Option 1 (in Million dollars)
0.0 Investment in Option 2 (in Million dollars)
35.0 Investment in Option 3 (in Million dollars)
5.0 Investment in Option 4 (in Million dollars)
22.5 Investment in Option 5 (in Million dollars)
17.5 Investment in Option 6 (in Million dollars)

$13.5 of net profit (in Million dollars)
```

Question 4: Lights Out.

In Tiger Electronic's handheld solitaire game Lights Out, the player strives to turn out all 25 lights that make up a 5 × 5 grid of cells. On each turn, the player is allowed to click on any one cell. Clicking on a cell activates a switch that causes the states of the cell and its (edge) neighbors to change from on to off, or from off to on. Corner cells are considered to have 2 neighbors, edge cells to have three, and interior cells to have four. Find a way to turn out all the lights in as few turns as possible (starting from the state where all lights are on).

In [6]:

```
m4 = Model(solver=GurobiSolver(OutputFlag=0))

@variable(m4, z[1:5,1:5], Bin)  # Binary variable for each cell
@variable(m4, t[1:5,1:5], Int)  # auxiliary variable

# Adding constraints by going through each cell - The number of 1's by summing u
p should be an odd number and so
# the sum should be equal to 2*any integer + 1. We know that 2*any number + 1 wi
ll always be odd

# Corner cells are considered to have 2 neighbors - (1, 1) (5, 1) (1, 5) (5, 5)
are the four corners
# edge cells to have three (when i == 1 & 5 or j == 1 and j == 5)
# interior cells to have four
```

```
for i in 1:5
    for j in 1:5
        if i == 1
            if j == 1
                @constraint(m4, z[i,j] + z[i+1,j] + z[i,j] + z[i,j+1] - z[i,j] =
= (2t[i,j] + 1))
            elseif j == 5
                @constraint(m4, z[i,j] + z[i+1,j] + z[i,j] + z[i,j-1] - z[i,j] =
= (2t[i,j] + 1))
            else
                @constraint(m4, z[i,j] + z[i+1,j] + z[i,j] + z[i,j+1] + z[i,j-1]
- z[i,j] == (2t[i,j] + 1))
            end
        elseif i == 5
            if j == 1
                @constraint(m4, z[i,j] + z[i-1,j] + z[i,j] + z[i,j+1] - z[i,j] =
= (2t[i,j] + 1))
            elseif j == 5
                @constraint(m4, z[i,j] + z[i-1,j] + z[i,j] + z[i,j-1] - z[i,j] =
= (2t[i,j] + 1))
            else
                @constraint(m4, z[i,j] + z[i-1,j] + z[i,j] + z[i,j+1] + z[i,j-1]
- z[i,j] == (2t[i,j] + 1))
            end
        elseif j == 1
            @constraint(m4, z[i,j] + z[i-1,j] + z[i+1,j] + z[i,j] + z[i,j+1] - z
[i,j] == (2t[i,j] + 1))
        elseif j == 5
            @constraint(m4, z[i,j] + z[i-1,j] + z[i+1,j] + z[i,j] + z[i,j-1] - z
[i,j] == (2t[i,j] + 1))
        else
            @constraint(m4, z[i,j] + z[i-1,j] + z[i+1,j] + z[i,j] + z[i,j-1] + z
[i,j+1] - z[i,j] == (2t[i,j] + 1))
        end
    end
end

# Objective function is to minimize the sum
@objective(m4, Min, sum(z[i,j] for i in 1:5, j in 1:5))

solve(m4)
```

Out[6]:

:Optimal

In [7]:

```
for i in 1:5
    for j in 1:5
        print(Int64(getvalue(z[i,j])), " ")
    end
    print("\n")
end
```

```
1  1  0  0  0
1  1  0  1  1
0  0  1  1  1
0  1  1  1  0
0  1  1  0  1
```

In [9]:

```
println("Objective Value: ", getobjectivevalue(m4))
```

```
Objective Value: 15.0
```

In [ ]: