Question 1

Paul Hollywood bakes two types of cakes: cheesecakes and black forest cakes. During any month, he can bake at most 65 cakes in total. The costs per cake and the demands for cakes, which must be met in time, are given in the following table.

	Month 1	**Month 1**	**Month 2**	**Month 2**	**Month 3**	**Month 3**
Item	Demand	Cost/cake($)	Demand	Cost/cake($)	Demand	Cost/cake($)
Cheesecake	40	3.00	30	3.40	20	3.80
Black Forest	20	2.50	30	2.80	10	3.40

We assume that cakes baked during a month can be used to meet demand for this month. At the end of each month (after all cakes have been baked and the current month's demand has been satisfied), a holding cost of 50 cents per cheesecake and 40 cents per black forest cake is incurred for cakes left in inventory. Those cakes can be used to satisfy future demand.

In [ ]:
```
using JuMP, HiGHS

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

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

In [ ]:
```
# defining vectors
cakes = [:cheesecake, :blackforest]
months = 1:3
```

Out[ ]:
```
1:3
```

In [ ]:
```
# dictionary to store the holding costs
holding = [0.50, 0.40]
holding_costs = Dict(zip(cakes, holding))

using NamedArrays

# NamedArray for demands and production costs
demand_mat = [ 40 30 20; 20 30 10 ]
demands = NamedArray( demand_mat, (cakes, months), ("cake", "month") )

prod_mat = [3.00 3.40 3.80; 2.50 2.80 3.40]
production_costs = NamedArray(prod_mat, (cakes, months), ("cake", "month"))

# variables for number of cakes to make each month
@variable(cake_model, x[cakes, months] >= 0)

# variable to represent the numver of cakes left in inventory at the end of e
@variable(cake_model, h[cakes, months] >= 0)
```

Out[ ]:
```
2-dimensional DenseAxisArray{VariableRef,2,...} with index sets:
    Dimension 1, [:cheesecake, :blackforest]
    Dimension 2, 1:3
And data, a 2×3 Matrix{VariableRef}:
 h[cheesecake,1]   h[cheesecake,2]   h[cheesecake,3]
 h[blackforest,1]  h[blackforest,2]  h[blackforest,3]
```

In [ ]:
```
# defining constraints

# number of cakes produced in any month must be less than 65
@constraint(cake_model, total_cakes[i in months], sum(x[j, i] for j in cakes)

# originally no cakes in inventory, cakes made should equal inventory plus am
@constraint(cake_model, monthly_demand1, x[:cheesecake, 1] == demands[1, 1] +
@constraint(cake_model, monthly_demand2, x[:blackforest, 1] == demands[2, 1]

# conservation that the cakes produced plus the previous cakes should equal t
@constraint(cake_model, monthly_demand3, h[:cheesecake, 1] + x[:cheesecake, 2
@constraint(cake_model, monthly_demand4, h[:blackforest, 1] + x[:blackforest,
@constraint(cake_model, monthly_demand5, h[:cheesecake, 2] + x[:cheesecake, 3
@constraint(cake_model, monthly_demand6, h[:blackforest, 2] + x[:blackforest,
```

Out[ ]:
monthly_demand6 : $x_{blackforest,3} + h_{blackforest,2} - h_{blackforest,3} = 10.0$

In [ ]:
```
# objective function
month1 = (x[:cheesecake, 1] * 3) + (x[:blackforest, 1] * 2.50) + (h[:cheeseca
month2 = (x[:cheesecake, 2] * 3.4) + (x[:blackforest, 2] * 2.80) + (h[:cheese
month3 = (x[:cheesecake, 3] * 3.8) + (x[:blackforest, 3] * 3.4) + (h[:cheesec

@objective(cake_model, Min, month1 + month2 + month3)

print(cake_model)
```

$$\begin{aligned}
\min \quad & 3x_{cheesecake,1} + 2.5x_{blackforest,1} + 0.4h_{cheesecake,1} + 0.5h_{blackforest,1} + 3.4x_{ch} \\
& + 0.4h \\
\text{Subject to} \quad & x_{cheesecake,1} - h_{cheesecake,1} = 40.0 \\
& x_{blackforest,1} - h_{blackforest,1} = 20.0 \\
& x_{cheesecake,2} + h_{cheesecake,1} - h_{cheesecake,2} = 30.0 \\
& x_{blackforest,2} + h_{blackforest,1} - h_{blackforest,2} = 30.0 \\
& x_{cheesecake,3} + h_{cheesecake,2} - h_{cheesecake,3} = 20.0 \\
& x_{blackforest,3} + h_{blackforest,2} - h_{blackforest,3} = 10.0 \\
& x_{cheesecake,1} + x_{blackforest,1} \leq 65.0 \\
& x_{cheesecake,2} + x_{blackforest,2} \leq 65.0 \\
& x_{cheesecake,3} + x_{blackforest,3} \leq 65.0 \\
& x_{cheesecake,1} \geq 0.0 \\
& x_{blackforest,1} \geq 0.0 \\
& x_{cheesecake,2} \geq 0.0 \\
& x_{blackforest,2} \geq 0.0 \\
& x_{cheesecake,3} \geq 0.0 \\
& x_{blackforest,3} \geq 0.0 \\
& h_{cheesecake,1} \geq 0.0 \\
& h_{blackforest,1} \geq 0.0 \\
& h_{cheesecake,2} \geq 0.0 \\
& h_{blackforest,2} \geq 0.0 \\
& h_{cheesecake,3} \geq 0.0 \\
& h_{blackforest,3} \geq 0.0
\end{aligned}$$

In [ ]:
```
optimize!(cake_model)
```

```
Solving LP without presolve or with basis
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0    -1.0000036897e-01 Ph1: 5(7); Du: 2(0.1) 0s
          4     4.6500000000e+02 Pr: 0(0) 0s
Model   status      : Optimal
Simplex   iterations: 4
Objective value     :  4.6500000000e+02
HiGHS run time      :          0.01
```