Micro Brewery 2

In this assignment the Micro Brewery assignment is expanded by making several different beers. They can however not be produced at the same time. Hence, for each month, you need to decide **which** beer to produce. This is a discrete decision, i.e. binary variables has to be used.

Problem

- Minimize the storage cost while satisfying the demand and planning the production of several different beer types

Sets

- Months: $m \in Months = \{jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec\}$.
- Beers: $Beers = \{"TSP - Stout", "Knapsack - Dark", "SetPartioning - Light"\}$

Parameters

- $Demand_{b,m}$: The number of liters of beer of type b you have to deliver to the cafe's in month m
- $BrewCap$: No. of liters of beer you at most can brew pr. month
- $StoreCap$: No. of liters of beer you at most can store in the basement
- $Cost$: Cost pr. liter for storing beers in the basement (to your mum)
- $InitialStorage_b$: How much is on store of beer b initially

Decision variables

- Amount of beer b produced each month m: $0 \leq x_{b,m} \leq BrewCap$.
- Amount of beer b stored each month m: $0 \leq y_{b,m} \leq StoreCap$.
- Which beer b should be produced in month m: $q_{b,m} \in \{0, 1\}$

Model

Objective:

- Total storage costs to be minimized:

$$\sum_{b,m} Cost \cdot y_{b,m}$$

Constraints:

- Storage (balance) constraint, what goes in, must come out:

  $y_m = x_{b,m} - Demand_{b,m} + y_{b,m-1} | (m > 1) + InitialStorage_b | (m = 1) \; \forall \, b, m$

- Brew capacity limitation:

$$x_{b,m} \leq BrewCap \cdot q_{b,m} \; \forall \; b, m$$

- Storage capacity limitation:

$$\sum_{b} y_{b,m} \leq StoreCap \; \forall \; m$$

- Limit production to at most one type each month:

$$\sum_{b} q_{b,m} \leq \; \forall \; m$$

The full model in Julia/JuMP, available with the name

`MicroBrewery2.jl`

from the book web-site, is given below:

```
#*************************************************************************
# Microbreweri2 Assignment, "Mathematical Programming Modelling" (42112)
# Intro definitions
using JuMP
using HiGHS
#*************************************************************************

#*************************************************************************
# PARAMETERS
Months = ["Jan","Feb","Mar","Apr","May",
```

```
          "Jun","Jul","Aug","Sep","Oct","Nov","Dec"]
Beers = ["TSP-Stout","Knapsack-Dark","SetPartioning-Light"]

# Prod and store data
BrewCap = 120

StoreCap = 300
Cost = 1

# demand from cafes of beers of different types Demand[b,m]
Demand =
    [35 20 15 45 25  65  40  50  35 85 50 55;
     15 10 20 15 15  55  90  80  25 45  5 30;
     5 20 20 35 35  80   60  30  35 20 20 40]
InitialStorage=[25 65 75]

# size of sets
M = length(Months)
B = length(Beers)
#*****************************************************************************

#*****************************************************************************
# Model
microbreweri2 = Model(HiGHS.Optimizer)

# production
@variable(microbreweri2, 0 <= x[1:B,1:M])

# storage in the end of the month
@variable(microbreweri2, 0 <= y[1:B,1:M])

# production allowed
@variable(microbreweri2, q[1:B,1:M], Bin)


# Minimize storage cost
@objective(microbreweri2, Min,
           sum( Cost*y[b,m] for m=1:M,b=1:B))

# storage balance constraint
@constraint(microbreweri2, [b=1:B,m=1:M],
            y[b,m] == (m>1 ? y[b,m-1] : InitialStorage[b]) +
            x[b,m] - Demand[b,m])

# limit production and set binary variable
@constraint(microbreweri2, [b=1:B,m=1:M], x[b,m] <= BrewCap*q[b,m])
```

```
# limit stored quanties
@constraint(microbreweri2, [m=1:M], sum(y[b,m] for b=1:B) <= StoreCap)

# only produce one type of beer each month
@constraint(microbreweri2, [m=1:M], sum(q[b,m] for b=1:B) <= 1)
#*****************************************************************************


#*****************************************************************************
# solve
optimize!(microbreweri2)
println("Termination status: $(termination_status(microbreweri2))")
#*****************************************************************************

#*****************************************************************************
# Report results
println("---------------------------------------");
if termination_status(microbreweri2) == MOI.OPTIMAL
    println("RESULTS:")
    println("Objective: $(objective_value(microbreweri2))")
else
  println("  No solution")
end
println("---------------------------------------");
#*****************************************************************************


#*****************************************************************************
println("Successfull end of $(PROGRAM_FILE)")
#*****************************************************************************
```