Soluções Laboratório 1

Questão 1 (Formulação Matemática)

Especificação em Julia

A solução é

Julia Notebook

```
using JuMP
using GLPK

me = Model(GLPK.Optimizer);

@variable(me, x[1:3] >= 0)
@variable(me, y[1:3,1:5] >= 0);

@objective(me, Max, 0.5*x[1]+1.5*x[2]+0.7*x[3]
    -0.05*y[1,1]-0.07*y[1,2]-0.11*y[1,3]-0.15*y[1,4]-0.15*y[1,5]
    -0.08*y[2,1]-0.06*y[2,2]-0.1*y[2,3]-0.12*y[2,4]-0.15*y[2,5]
    -0.1*y[3,1]-0.09*y[3,2]-0.09*y[3,3]-0.1*y[3,4]-0.16*y[3,5])

@constraints(me, begin
    y[1,1]+y[2,1]+y[3,1]==2700
    y[1,2]+y[2,2]+y[3,2]==2700
    y[1,3]+y[2,3]+y[3,3]==9000
    y[1,4]+y[2,4]+y[3,4]==4500
    y[1,5]+y[2,5]+y[3,5]==3600
    end)

@constraints(me, begin
  y[1,1]+y[1,2]+y[1,3]+y[1,4]+y[1,5]<=4500
  y[2,1]+y[2,2]+y[2,3]+y[2,4]+y[2,5]<=9000
  y[3,1]+y[3,2]+y[3,3]+y[3,4]+y[3,5]<=11250
    end)

@constraints(me, begin
    x[1]==y[1,1]+y[1,2]+y[1,3]+y[1,4]+y[1,5]
```

```
    x[2]==y[2,1]+y[2,2]+y[2,3]+y[2,4]+y[2,5]
    x[3]==y[3,1]+y[3,2]+y[3,3]+y[3,4]+y[3,5]
    end)

c = [[0.05 0.07 0.11 0.15 0.15]; [0.08 0.06 0.10 0.12 0.15]; [0.10
↪  0.09 0.09 0.10 0.16]]
C = [4500,9000,11250]
d = [2700,2700,9000,4500,3600]
p = [2.0,1.0,1.8];

mc = Model(GLPK.Optimizer);

@variable(mc, x[1:3] >= 0)
@variable(mc, y[1:3,1:5] >= 0);

@objective(mc, Max, sum((2.5-p[i])*x[i] for i=1:3)-sum(c[i,j]*y[i,j]
↪  for i=1:3, j=1:5))

@constraints(mc, begin
    [j=1:5], sum(y[i,j] for i=1:3)==d[j]
    [i=1:3], sum(y[i,j] for j=1:5)<=C[i]
    [i=1:3], x[i] == sum(y[i,j] for j=1:5)
    end)
```

Questão 2 (Formulação Matemática)

Especificação em Julia

A solução é

Julia Notebook

```
using JuMP
using GLPK

m = Model(GLPK.Optimizer);
@variable(m, x[1:2] >= 0)

@objective(m, Min, 3*x[1]+6*x[2])
```

```
@constraint(m, 3x[1]+2x[2] <= 18)

@constraint(m, x[1]+x[2] >=5)

@constraint(m, x[2] <= 7.0/8*x[1])

@constraints(m ,begin
    x[1] <= 4
    x[2] <= 7
    end)
```

Questão 3 (Formulação Matemática)

Especificação em Julia

A solução é

Julia Notebook

```
using JuMP
using GLPK

n = 10
ac = 10*rand(n) # acidez
dc = 10*rand(n) # doçura
al = 10*rand(n) # álcool
r  = 100*rand(n) # custo
P  = rand(Bool,n);

m = Model(GLPK.Optimizer);

@variable(m, x[1:n] >= 0)
@variable(m, X >= 0);

@objective(m, Min, sum(r[i]*x[i] for i=1:n))

@constraints(m, begin
    X == sum(x[i] for i=1:n)
```

```
    X == 1
    0.2 *X <= sum(ac[i]*x[i] for i=1:n)
              sum(ac[i]*x[i] for i=1:n) <= 0.3 *X
    0.3 *X <= sum(ac[i]*x[i] for i=1:n)
              sum(ac[i]*x[i] for i=1:n) <= 0.4 *X
    0.03*X <= sum(al[i]*x[i] for i=1:n)
              sum(al[i]*x[i] for i=1:n) <= 0.04*X
    sum(x[i] for i=1:n if P[i]) >= 2*sum(x[i] for i=1:5 if !P[i])
    end
    )
```

Questão 4 (Formulação Matemática)

Especificação em Julia

A solução é [encoding=utf8,outencoding=utf8]constructionconstruction

Questão 5 (Formulação Matemática)

Especificação em Julia

A solução é

Julia Notebook

```
using JuMP
using GLPK

pe = [ i-1 for i=1:6]; pe[1]=6;
nm = [ 22, 55, 88, 110, 44, 33];

m = Model(GLPK.Optimizer);

@variable(m, x[1:6] >= 0)

@objective(m, Min, sum(x[i] for i=1:6))

@constraint(m, [i=1:6], x[i]+x[pe[i]] >= nm[i])
```