Glassco

Glassco manufactures wine glasses, beer glasses, champagne glasses and whiskey glasses. Each type of glass uses time in the molding shop, time in the packaging shop, and a certain amount of glass. The resources required to make each type of glass are given in the following table:

Data	WINE GLASS	BEER GLASS	CHMPGNE GLASS	WHISKEY GLASS
Molding time	4 minutes	9 minutes	7 minutes	10 minutes
Packaging time	1 minute	1 minute	3 minutes	39 minutes
Glass	3 oz	4 oz	2 oz	1 oz
Selling price	$6	$10	$9	$20

At present, 950 minutes of molding time, 600 minutes of packaging time and 200 oz of glass are available.

Write down the LP (in GAMSPy) that Glassco should solve, assuming the company wishes to maximize revenue.

In [31]:
```
import sys
import numpy as np

from gamspy import (
    Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
    Domain,Number,Sum,Product,Smax,Smin,Ord,Card,SpecialValues,
)

options = Options(variable_listing_limit=0)
m = Container(options=options)
```

In [32]:
```
# WRITE YOUR PRIMAL LP MODEL HERE
A = Set(m, 'A', records=['molding', 'packaging', 'glass'])
P = Set(m, 'P', records=['wine', 'beer', 'champagne', 'whisky'])

actions = Parameter(m, 'actions', domain=[A, P], records=np.array([[4, 9, 7,
                                                                    [1, 1, 3,
                                                                    [3, 4, 2,


cost = Parameter(m, 'cost', domain=P, records=np.array([6, 10, 9, 20]))
time = Parameter(m, 'time', domain=A, records=np.array([950, 600, 200]))

x = Variable(m, 'x', 'positive', domain=P)
```

```
time_c = Equation(m, 'time_c', domain=A)
time_c[A] = Sum(P, actions[A, P] * x[P]) <= time[A]

primal = Model(m,
    name="primal",
    equations=m.getEquations(),
    problem=Problem.LP,
    sense=Sense.MAX,
    objective=Sum(P, cost[P] * x[P]),
)
```

In [33]:
```
primal.solve()
```

Out[33]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	1024	4	5	LP	CPLEX	

Write down (and solve) the dual of this LP problem

You should set up a separate model and include just those equations needed in each model in the model statement.

In [39]:
```
# WRITE YOUR DUAL LP MODEL HERE
pi = Variable(m, 'pi', type='positive', domain=A)
dualcons = Equation(m, 'dcons', domain=P)
dualcons[P] = Sum(A, actions[A, P]*pi[A]) >= cost[P]

dual = Model(m,
    name="dual",
    equations=[dualcons],
    problem=Problem.LP,
    sense=Sense.MIN,
    objective=Sum(A, time[A]*pi[A]),
)

dual.solve()
```

Out[39]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	1024	5	4	LP	CPLEX	0.00

In [55]:
```
## Marginal values of Primal vs Values of Dual
reduced = Parameter(m, 'reduced', domain=P)
reduced[P] = x.m[P]

display("x:", x.records, 'reduced costs:', reduced.records, "require:", time

print("Notice how the marginal values of the primal solution are equal to th
```

```
'x:'
```

	P	level	marginal	lower	upper	scale
0	wine	0.0	-6.053333	0.0	inf	1.0
1	beer	0.0	-5.933333	0.0	inf	1.0
2	champagne	96.0	0.000000	0.0	inf	1.0
3	whisky	8.0	0.000000	0.0	inf	1.0

```
'reduced costs:'
```

	P	value
0	wine	-6.053333
1	beer	-5.933333

```
'require:'
```

	A	level	marginal	lower	upper	scale
0	molding	752.0	0.000000	-inf	950.0	1.0
1	packaging	600.0	0.413333	-inf	600.0	1.0
2	glass	200.0	3.880000	-inf	200.0	1.0

```
'pi:'
```

	A	level	marginal	lower	upper	scale
0	molding	0.000000	198.0	0.0	inf	1.0
1	packaging	0.413333	0.0	0.0	inf	1.0
2	glass	3.880000	0.0	0.0	inf	1.0

```
Notice how the marginal values of the primal solution are equal to the varia
ble values of the dual.
```

What is the solution of the dual problem? Can you show how the multipliers on the primal problem are related to the dual solution?

In [53]:
```
# UPDATE ALL INSTANCES OF XXXX in this cell.
# Quantities labelled 'primal' must only involve quantities associated with
# Quantities labelled 'dual' must only involve quantities associated with yo

obj = {}
obj['primal'] = 1024
obj['dual'] = 1024

xsolution = {}
xsolution['primal'] = x.records
xsolution['dual'] = dualcons.records['marginal']

usolution = {}
usolution['primal'] = time_c.records
usolution['dual'] = pi.records
```

```
display('obj=',obj,'x=',xsolution,'u=',usolution)
```

```
'obj='
{'primal': 1024, 'dual': 1024}
'x='
{'primal':            P  level  marginal  lower  upper  scale
 0       wine    0.0 -6.053333    0.0    inf    1.0
 1       beer    0.0 -5.933333    0.0    inf    1.0
 2  champagne   96.0  0.000000    0.0    inf    1.0
 3     whisky    8.0  0.000000    0.0    inf    1.0,
 'dual': 0     0.0
 1     0.0
 2    96.0
 3     8.0
 Name: marginal, dtype: float64}
'u='
{'primal':            A  level  marginal  lower  upper  scale
 0    molding  752.0  0.000000   -inf  950.0    1.0
 1  packaging  600.0  0.413333   -inf  600.0    1.0
 2      glass  200.0  3.880000   -inf  200.0    1.0,
 'dual':            A     level  marginal  lower  upper  scale
 0    molding  0.000000     198.0    0.0    inf    1.0
 1  packaging  0.413333       0.0    0.0    inf    1.0
 2      glass  3.880000       0.0    0.0    inf    1.0}
```

In [54]:
```
# Replace YYYY with an expression that predicts the change in revenue when g
# You cannot solve another model but may use results you obtained from the a

RevIncrease = Parameter(m,'RevIncrease')
RevIncrease[:] = (25) * time_c.records.loc[time_c.records['A'] == 'glass', '

print(f"The revenue is expected to increase ${RevIncrease.toValue().round()}
```

```
The revenue is expected to increase $97.0 if 225 oz of glass is available
```