Practice Final: Linear?

```
param T;
param Demand{1..T};
var InitialLevel;
var GrowthRate;
var Estimate{1..T};
minimize TotalError:
   sum{t = 1..T} (Estimate[t] - Demand[t])*
                 (Estimate[t] - Demand[t]);
s.t. DefineEstimate {t in 1..T}:
   Estimate[t] = InitialLevel + GrowthRate*t;
```

Linear?

```
Set QUESTIONS;                              /* The set of questions */
Param AvgPctCorrect{QUESTIONS};
Param MinAverage;                           /* minimum average score */
Param MaxAverage;                           /* maximum average score */
Var Points{QUESTIONS} >= 0; Var TooHigh >= 0;Var TooLow >= 0;
Minimize Error:
    TooHigh +  TooLow;
s.t. Nearly100TotalPoints:
    sum{q in QUESTIONS} Points[q] = 100+TooHigh - TooLow;
s.t. MeetMin:
    (sum{q in QUESTIONS} AvgPctCorrect[q]*Points[q])/
   (100+TooHigh - TooLow) >= MinAverage;
s.t. MeetMax:
    (sum{q in QUESTIONS} AvgPctCorrect[q]*Points[q])/
    (100+TooHigh - TooLow) <= MaxAverage;
```

Linear?

```
Set POOLS; Set STORES; Set DCS; Set PRODUCTS;
Param Demand{PRODUCTS, STORES}:
Param CubicCapacity; Param WeightLimit;
Param MaxFillTime{DCS, POOLS};
Param Weight{PRODUCTS}; Param Cube{PRODUCTS};
var WeighOut{DCS, POOLS} binary;
var UseEdge{DCS, POOLS} binary;
var Flow[PRODUCTS, DCS, POOLS} >= 0;
var Assign{POOLS, STORES} binary;
```

Linear Continued

s.t. ImposeTrailerFillbyWeight{dc in DCS, pool in POOLS}:

MaxFillTime[dc, pool]*sum{prd in PRODUCTS} Weight[prd]*Flow[prd, dc, pool]

>= WeightLimit*WeighOut[dc, pool]*UseEdge[dc, pool];

s.t. ImposeTrailerFillbyCube{dc in DCS, pool in POOLS}:

MaxFillTime[dc, pool]*sum{prd in PRODUCTS} Cube[prd]*Flow[prd, dc, pool]

>= CubicCapacity*CubeOut[dc, pool]*UseEdge[dc, pool];

s.t. DefineUseEdge{prd in PRODUCTS, dc in DCS, pool in POOLS}:

Flow[prd, dc, pool] <= (sum{store in STORES} Demand[prd, store])*UseEdge[dc, pool];

s.t. WeightOrCube{dc in DCS, pool in POOLS}:

WeighOut[dc, pool] + CubeOut[dc, pool] = 1;

Linear "And"

CubeOut[dc, pool]*UseEdge[dc, pool] equivalent to

CubeOut[dc, pool] AND

UseEdge[dc, pool]

Linear Version:

(CubeOut[dc, pool] + UseEdge[dc, pool] -1)

Or more precisely:

Var Both[dc, pool] binary

Both[dc, pool] >= CubeOut[dc, pool] + UseEdge[dc, pool] -1

Both[dc, pool] <= CubeOut[dc, pool]

Both[dc, pool] <= UseEdge[dc, pool]

Sensitivity Analysis

Decisions	Blend A	Blend B	Blend C	Total	Supply
Vintage 1	180.00	-	-	180	180
Vintage 2	246.71	3.29	-	250	250
Vintage 3	-	200.00	-	200	200
Vintage 4	22.46	377.54	-	400	400
Total Produced	449.17	580.83	-		
Total Sold	449.17	580.83	-	Profit	
Sales Price	$ 70.00	$ 40.00	$ 30.00	$ 54,675	

Min Blend %	Blend A	Blend B	Blend C
Vintage 1	180.00		-
Vintage 2	246.71	3.29	
Vintage 3		200.00	-
Vintage 4			
Least % of Total	75%	35%	50%
Balance	89.83	-	-
Must Be	0	0	0

Max Blend %	Blend A	Blend B	Blend C
Vintage 1			
Vintage 2			
Vintage 3			
Vintage 4	22.46		-
Least % of Total	5%	0%	40%
Balance	0	0	0
Must Be	0	0	0

Questions

- What is the minimum amount by which the selling price of C would have to change before it would be attractive to produce Blend C?

Decisions	Blend A	Blend B	Blend C	Total	Supply
Vintage 1	180.00	-	-	180	180
Vintage 2	246.71	3.29	-	250	250
Vintage 3	-	200.00	-	200	200
Vintage 4	22.46	377.54	-	400	400
Total Produced	449.17	580.83	-		
Total Sold	449.17	580.83	-	Profit	
Sales Price	$ 70.00	$ 40.00	$ 30.00	$ 54,675	

Min Blend %	Blend A	Blend B	Blend C
Vintage 1	180.00		-
Vintage 2	246.71	3.29	
Vintage 3		200.00	-
Vintage 4			
Least % of Total	75%	35%	50%
Balance	89.83	-	-
Must Be	0	0	0

Max Blend %	Blend A	Blend B	Blend C
Vintage 1			
Vintage 2			
Vintage 3			
Vintage 4	22.46		-
Least % of Total	5%	0%	40%
Balance	0	0	0
Must Be	0	0	0

- What are the Shadow Prices of the 4 vintages?
- What are the units of these prices?

Decisions	Blend A	Blend B	Blend C	Total	Supply
Vintage 1	180.00	-	-	180	180
Vintage 2	246.71	3.29	-	250	250
Vintage 3	-	200.00	-	200	200
Vintage 4	22.46	377.54	-	400	400
Total Produced	449.17	580.83	-		
Total Sold	449.17	580.83	-	Profit	
Sales Price	$ 70.00	$ 40.00	$ 30.00	$ 54,675	

Min Blend %	Blend A	Blend B	Blend C
Vintage 1	180.00		-
Vintage 2	246.71	3.29	
Vintage 3		200.00	-
Vintage 4			
Least % of Total	75%	35%	50%
Balance	89.83	-	-
Must Be	0	0	0

Max Blend %	Blend A	Blend B	Blend C
Vintage 1			
Vintage 2			
Vintage 3			
Vintage 4	22.46		-
Least % of Total	5%	0%	40%
Balance	0	0	0
Must Be	0	0	0

- What would be the impact of losing 100 gals of Vintage 3?

True or False

- In solving an integer programming problem, Solver and CPLEX employ genetic algorithms.
- In solving an integer programming problem, Solver and CPLEX employ the simplex method.
- The set of optimal solutions to an integer program describes a convex set.
- The function $f(x) = x^2$ (that's x*x) is a convex function

True or False

- The function $f(x,y) = x^2 + y^2$ (that's x*x + y*y) a convex function
- All polynomials are convex functions.
- If a real valued function f is a convex function then the function $g(x) = -f(x)$ is a convex function

Restrictions & Relaxations

- ______ ≤ ______ ≤ ______ ≤ ______

A. Optimal objective function value of the linear programming relaxation with integrality restrictions on some, but not all, variables.

B. Optimal objective function value of the integer program.

C. Optimal objective function value of the complete linear programming relaxation (no integrality restrictions).

D. Optimal objective function value of the integer program if we fix the values of some of the variables.

Restrictions & Relaxations

- Minimizing: __C___ ≤ __A__ ≤ ___B__ ≤ ___D__

A. Optimal objective function value of the linear programming relaxation with integrality restrictions on some, but not all, variables.

B. Optimal objective function value of the integer program.

C. Optimal objective function value of the complete linear programming relaxation (no integrality restrictions).

D. Optimal objective function value of the integer program if we fix the values of some of the variables.

IP Modeling

- There are N items numbered 1, 2, …, N. If we choose item i, we earn r[i]. If we select exactly one of the items, we must pay b. You may introduce other variables if you wish. You will certainly need to add an objective (minimize total cost) and constraints.

- param N; # the number of items.
- param r{1..N}; # the revenue on each item
- param b; # the cost incurred if we choose exactly one item

- var x{1..N} binary; # x[i] is one if we choose item i and 0 otherwise.

IP Modeling

- param N; # the number of items.
- param r{1..N}; # the revenue on each item
- param b; # the cost incurred if we choose exactly one item
- var x{1..N} binary; # x[i] is one if we choose item i and 0 otherwise.
- Var All binary; # is 1 if we choose all
- maximize Revenue : sum{k = 1..N} r[k]*x[k] - b*All;
- s.t. NotNecessaryButForCompleteness{k in 1..N}:
  - All <= x[k];
- s.t. DidWeSelectAll:
  - N*All >= sum{k in 1..N} x[k];