PART B

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1.1

a.)

```
var TV;

var MAG;

maximize audience: 1800000* TV + MAG * 1000000;

subject to cost: (20000)* TV + (10000) * MAG = 1000000;

subject to TV_limit:  TV >= 10;

subject to MAG_limit:  MAG >=1;
```

Answer:

```
optimal solution; objective 9.8e+07
```

= 98 mil

```
ampl: display MAG;

MAG = 80
```

b)

```
var TV;

var MAG;

maximize audience: 1800000* TV + MAG * 1000000;

subject to cost: (20000)* TV + (10000) * MAG = 1000000;

subject to TV_limit:  TV >= 10;

subject to MAG_limit:  MAG >=1;

subject to staff_cost: MAG * (3) + TV <=100; # added constraint
```

Answer:

optimal solution; objective 9.2e+07

=92mil

TV = 40

MAG = 20

c.)

```
var TV;

var MAG;

var RADIO;

maximize audience: 1800000* TV + MAG * 1000000 + 250000 * RADIO;

subject to cost: (20000)* TV + (10000) * MAG + (2000) * RADIO = 1000000;

subject to TV_limit:  TV >= 10;

subject to MAG_limit:  MAG >=1;

subject to staff_cost: MAG * (3) + TV + RADIO <=100;
```

Answer:

```
optimal solution; objective 93794444.44

RADIO = 52.7778

TV = 44.2222

MAG = 1
```

d.)

```
var TV;

var MAG;

var RADIO;

maximize audience: 1800000* TV + MAG * 1000000 + 250000 * RADIO;

subject to cost: (20000)* TV + (10000) * MAG + (2000) * RADIO = 1000000;

subject to TV_limit:  TV >= 10;

subject to MAG_limit:  MAG >=2;

subject to Radio_sub:  RADIO >=120;

subject to staff_cost: MAG * (3) + TV + RADIO <=100;
```

Answer:

```
presolve: constraint staff_cost cannot hold:

        body <= -20 cannot be >= 16; difference = -36
```

1.5

a.)

```
set Items;

param Value {Items}; # value of one item

param Weight {Items}; # weight of one item
param Volume {Items}; # volume of one item
param Available {Items}; # number of items available
param MaxWeight; # maximum weight that can be handled per day
param MaxVolume; # maximum volume that can be handled per day
var ItemCount {Items};
maximize TotalValue: sum {i in Items} (ItemCount[i] * Value[i]);
subject to WeightLimit: sum {i in Items} (ItemCount[i] * Weight[i]) <= MaxWeight;
subject to VolumeLimit: sum {i in Items} (ItemCount[i] * Volume[i]) <= MaxVolume;
subject to Availability {i in Items}: 0 <= ItemCount[i] <= Available[i];
```

b.)

results:

Profit: 4800

Number of Items:

'CD player' 30

TV 0

VCR 2

camcorder 15

camera 20

radio 0

c.)

Add parameters for minimum and respectively maximum stock per item. Add another constraint for quantity

```
set Items;
param Value {Items}; # value of one item
param Weight {Items}; # weight of one item
param Volume {Items}; # volume of one item

param Available {Items}; # number of items available

param MinGet {Items}; # minimum number to acquire

param MaxSell {Items}; # maximum number that can sell

param MaxWeight; # maximum weight that can be handled per day

param MaxVolume; # maximum volume that can be handled per day

var ItemCount {Items};

maximize TotalValue: sum {i in Items} (ItemCount[i] * Value[i]);

subject to WeightLimit: sum {i in Items} (ItemCount[i] * Weight[i]) <= MaxWeight;

subject to VolumeLimit: sum {i in Items} (ItemCount[i] * Volume[i]) <= MaxVolume;

subject to Availability {i in Items}: 0 <= ItemCount[i] <= Available[i];

subject to StockLimits {i in Items}: MinGet[i] <= ItemCount[i] <= MaxSell[i];
```

d.)

Partners Evaluation: the group would ask how much they would be able to handle in terms of weight and volume. The limit values for Weight and volume would be increased by the amounts corresponding to the new partner. The group would then assess the extra profit generated together with the cost of the new partner handling the extra items.

e.)

The optimum solution from AMPL indicates a profit of 212.5 by acquiring 2.5 cameras. The limiting value is the volume limit of 5 cubic feet. For an all integer solution, the 0.5 camera can be replaced with either a radio or a CD player, as each of them requires only one cubic foot. Since the value of a CD player is higher, the optimal all integer solution is to acquire 2 cameras and 1 CD player - for a total profit of 210.

2.1

```
set Foods;

param calories{Foods}; #no of calories in food

param protein{Foods};   #no of protein in food
param calcium{Foods}; #no of calcium in food

param vitaminA{Foods}; #no of vitamin in food

param pound{Foods}; #no of cost of food

var Foodcount {Foods}  >=0; #food quantity



minimize cost: sum{i in Foods} pound[i]*Foodcount[i];

subject to calconstraint: sum{i in Foods} (Foodcount[i]*calories[i])>=3000;

subject to proteinconstraint: sum{i in Foods} (Foodcount[i]*protein[i])>=70;

subject to calciumconstraint: sum{i in Foods} (Foodcount[i]*calcium[i])>=800;

subject to vitconstraint: sum{i in Foods} (Foodcount[i]*vitaminA[i])>=500;
```

Result:

```
TotalCost 0.6196227853

Foodcount [*] :=

   bread  0

 cabbage  0

 gelatin  0

    meat  0

    milk  0.815393

potatoes  8.64165

;
```

2.3

```
set Sugars;

set Suppliers;
```

```
param Percentage {Sugars, Suppliers}; # percentage sugar in a supplier's mix

param Cost {Suppliers}; # cost of one ton of supplier's mix

param BlendQuantity {Sugars}; # quantity of a sugar in the target blend

var Quantity {Suppliers}; # quantity of a supplier's mix to buy

minimize TotalCost: sum {i in Suppliers} (Quantity[i] * Cost[i]);

subject to QuantityBounds {i in Suppliers}: Quantity[i] >= 0;

subject to SugarQuantity {i in Sugars}: sum {j in Suppliers} (Percentage[i, j] *
Quantity[j] / 100) = BlendQuantity[i];
```

Results:

```
TotalCost = 2068.5

Quantity [*] :=

A 60

B 0

C 0

D 0

E 0

F 45.5

G 61.5

;
```

b)

```
set Sugars;

set Suppliers;
```

```
param Percentage {Sugars, Suppliers}; # percentage of a sugar in a supplier's mix

param Cost {Suppliers}; # cost of one ton of supplier's mix

param BlendQuantity {Sugars}; # the quantity of a sugar in the target blend

param MinQuantity {Suppliers}; # minimum quantity to buy

var Quantity {Suppliers}; # quantity of a supplier's mix to buy

minimize TotalCost: sum {i in Suppliers} (Quantity[i] * Cost[i]);

subject to QuantityBounds {i in Suppliers}: Quantity[i] >= MinQuantity[i];

subject to SugarQuantity {i in Sugars}: sum {j in Suppliers} (Percentage[i, j] *
Quantity[j] / 100) = BlendQuantity[i];
```

Results:

```
TotalCost = 2093.954545

Quantity [*] :=

A 43.6364

B 10

C 10

D 10

E 10

F 30.9545

G 52.4091
```

Analysis: it can be observed that the total cost has increased by a small amount.

c)

```
set Sugars;
set Suppliers;

param Percentage {Sugars, Suppliers}; # percentage of a sugar in a supplier's mix

param Cost {Suppliers}; # cost of one ton of supplier's mix

param MinBlend; # the lowest percentage of each sugar in the target blend

param MaxBlend; # the highest percentage of each sugar in the target blend

param TotalQuantity; # total quantity of blend to produce

var Quantity {Suppliers}; # quantity of a supplier's mix to buy

minimize TotalCost: sum {i in Suppliers} (Quantity[i] * Cost[i]);

subject to QuantityBounds {i in Suppliers}: Quantity[i] >= 0;

subject to QuantitySum: sum {i in Suppliers} Quantity[i] = TotalQuantity;

subject to BlendRatio {i in Sugars}: (MinBlend * TotalQuantity) <= (sum {j in
Suppliers} (Percentage[i, j] * Quantity[j])) <= (MaxBlend * TotalQuantity);
```

Results:

```
TotalCost = 12.25

Quantity [*] :=

A 0.4

B 0

C 0

D 0

E 0

F 0.25

G 0.35
```

Sugar percentage in final blend:

Cane 30%

Corn 33%

Beet 37%

3.1

```
set Orig;# orgin of supply
set Dest; #dest of supply
param supply{Orig}; #no of supply
param demand{Dest};    #no of demand
check: sum{i in Orig} supply[i]>=sum{j in Dest} demand[j]; #sum supplies > or equal to demand

param miles{Orig,Dest}>=0; #miles asscoiated with origin and dest
var Trans{Orig,Dest}>=0; # #amt transported

minimize Total_Cost: sum{i in Orig, j in Dest} (Trans[i,j]*miles[i,j]);
subject to Supply {i in Orig}:
sum {j in Dest} Trans[i,j] = supply[i];
subject to Demand {j in Dest}:
sum {i in Orig}  Trans[i,j] = demand[j];
subject to Mile: sum{i in Orig,j in Dest} (miles[i,j]/1000)*90;  # represent per mile constraint
```

Result

```
display {i in Orig, j in Dest} (Trans[i,j]*miles[i,j]);
Trans[i,j]*miles[i,j] :=
SanDiego Chicago   405000
SanDiego Newyork   937500
Seattle  Chicago   127500
Seattle  Topeka    495000
;

ampl: display Trans;
Trans :=
SanDiego Chicago   225
SanDiego Newyork   375
Seattle  Chicago    75
Seattle  Topeka    275
```