ISE-DSA 5113 Homework 5

Group 10 Team Members:

- Orkhan Khankishiyev
- Anvitha Reddy Thummalapally
- Tahsin Tabassum

Question 1.

(a) Solution:

Formulation of the model:

Sets:

A = Set of airports, $i \in A$

Parameter:

f_i = Fuel burn from traveling from $i \in A$ to the next airport

c_i = Fuel cost at $i \in A$

r_i = Ramp fee at $i \in A$

w_i = Minimum gallon of fuel required to buy for waiving ramp fee at $i \in A$

p_i = Number of passengers flying from $i \in A$ to the next airport

pw = Passenger weight (per person)

tc = Tank capacity of the aircraft

rw = Maximum ramp weight of the aircraft

lw = Maximum landing weight of the aircraft

ow = Basic operating weight (BOW) of the aircraft

lf = Minimum fuel required of the aircraft during landing

uf = Fuel up level of the aircraft

sf = Starting fuel of the aircraft from the 1st airport

Decision Variable:

x_i = Amount of fuel to purchase (in gallons) at $i \in A$

y_i = Binary variable, indicating whether ramp fee is to be paid at $i \in A$

z_i = Remaining fuel amount (in gallons) after a trip is completed at $i \in A$

Objective Function:

$$\text{Minimize} \sum_{i \in A} 6.7 * c_i * x_i + \sum_{i \in A} r_i * y_i$$

Constraints:

The amount of fuel remaining after traveling from airport $i \in A$ to airport $i + 1$ is equal to the fuel purchased at airport i plus the remaining fuel at airport i, minus the fuel burned during the journey.

$$z_{i+1} = x_i + z_i - f_i \quad , \forall\, i \in A$$

The total weight of the aircraft (including basic operating weight, passenger weight, remaining fuel, and purchased fuel) does not exceed the maximum ramp weight at each airport i.

$$ow + pw * p_i + z_i + x_i \leq rw \quad , \forall\, i \in A$$

The total weight of the aircraft (including basic operating weight, passenger weight, and remaining fuel) does not exceed the maximum landing weight at each airport i after refueling.

$$ow + pw * p_i + z_{i+1} \leq lw \quad , \forall\, i \in A$$

The initial amount of remaining fuel at the first airport is equal to the starting fuel amount.

$$z_1 = sf$$

The amount of remaining fuel plus the fuel purchased at the last airport is greater than or equal to the starting fuel amount.

$$z_6 + x_6 \geq sf$$

The total amount of fuel (purchased plus remaining) does not exceed the tank capacity of the aircraft at each airport i.

$$x_i + z_i \leq tc \quad , \forall\, i \in A$$

The amount of remaining fuel at each airport i is greater than or equal to the minimum fuel required for landing. (*Note: For a no-tankering solution, the amount of remaining fuel at each airport i is equal to the minimum fuel required for landing.*)

$$z_i \geq lf \quad , \forall\, i \in A$$

If the binary variable y_i is 0 (indicating that the ramp fee is not paid), then the amount of fuel purchased at airport i must be greater than or equal to the minimum gallon of fuel required to buy for waiving the ramp fee. Otherwise, if y_i is 1, no fuel purchase is necessary.

$$x_i \geq 6.7 * w_i * (1 - y_i) \quad , \forall\, i \in A$$

The decision variables x_i and z_i are continuous, and can take any values more than 0, whereas y_i are binary integer variables.

$$x_i \geq 0 \quad , \forall\, i \in A$$

$$y_i \in (0,1) \quad , \forall\, i \in A$$

$$z_i \geq 0 \quad , \forall\, i \in A$$

AMPL code:

```
reset;
option solver cplex;

set A;  # Set of airports

param f{A};  # Fuel burn from airport i to the next airport
param C{A};  # Fuel cost at airport i
param r{A};  # Ramp fee at airport i
param w{A};  # Minimum gallon of fuel required to buy for waiving ramp fee at airport
i
param p{A};  # Number of passengers flying from airport i to the next airport
param pw;    # Passenger weight (per person)
param tc;    # Tank capacity of the aircraft
param rw;    # Maximum ramp weight of the aircraft
param lw;    # Maximum landing weight of the aircraft
param ow;    # Basic operating weight (BOW) of the aircraft
param lf;    # Minimum fuel required of the aircraft during landing
param uf;    # Fuel up level of the aircraft
param sf;    # Starting fuel of the aircraft from the 1st airport
param bf;    # (for 1b) Minimum amount of fuel that must be bought in case fuel is
needed
var x{i in A} >= 0;  # Amount of fuel to purchase (in gallons) at airport i
var y{i in A} binary;# Binary variable, indicating whether ramp fee is to be paid at
airport i
var z{i in A} >= 0;  # Remaining fuel amount (in gallons) after a trip completed at
airport i
var v{i in A} binary;# (for 1b) Binary variable, indicating whether fuel should be
brought from airport i
minimize TotalCost: sum{i in A} (6.7 * C[i] * x[i]) + sum{i in A} (r[i] * y[i]);
subject to RemainingFuel{i in A}: z[i] = x[i] + z[i] - f[i];
subject to MaxRampWeight{i in A}: ow + pw * p[i] + z[i] + x[i] <= rw;
subject to MaxLandingWeight{i in A}: ow + pw * p[i] + z[i] <= lw;
subject to InitialFuel: z['KCID'] = sf;
subject to FuelAtLastAirport: z['KTUL'] + x['KTUL'] >= sf;
subject to TankCapacity{i in A}: x[i] + z[i] <= tc;
subject to MinFuelForLanding{i in A}: z[i] >= lf;
subject to FuelPurchase{i in A}: x[i] >= 6.7 * w[i] * (1 - y[i]);
data Group10_HW5_Q1.dat;

solve;
display x;
display y;
display z;
display v;
```

Data file:

```
data;

set A := KCID KACK KMMU KBNA KTUL;

param:    f        C      r       w     p:=
KCID      5100     4      0       0      2
KACK      2200     8.32   800     600    4
```

```
KMMU      4700     8.99   750     500   8
KBNA      3800     6.48   600     650   8
KTUL      3600     9.27   800     500   8;

param  pw := 200;
param  tc := 14000;
param  rw := 36400;
param  lw := 31800;
param  ow := 22800;
param  lf := 2500;
param  uf := 4500;
param  sf := 7000;
```

Solution obtained from AMPL:

Considering “tankering”, the solution is:

ampl: model HW5_P1.mod;

CPLEX 20.1.0.0: optimal integer solution; objective 932385.1

0 MIP simplex iterations

0 branch-and-bound nodes

x [*] :=

KACK 2200

KBNA 3800

KCID 5100

KMMU 4700

KTUL 3600 ;

y [*] :=

KACK 1

KBNA 1

KCID 0

KMMU 0

KTUL 0 ;

z [*] :=

KACK 2500

KBNA 2500

KCID 7000

KMMU 2500

KTUL 3400 ;

Considering no “tankering”, the solution is:

x [*] :=

KACK 2200

KBNA 3800

KCID 5100

KMMU 4700

KTUL 0;

y [*] :=

KACK 1

KBNA 1

KCID 0

KMMU 0

KTUL 0;

z [*] :=

KACK 2500

KBNA 2500

KCID 2500

KMMU 2500

KTUL 2500 ;

(b) Solution:

Formulation of the model:

Extra parts with the formulation of 1(a):

Parameter:

bf = Minimum amount of fuel that must be bought in case fuel is needed

Decision Variables:

v_i = Binary variable, indicating whether fuel should be brought from $i \in A$

Constraints:

If the binary variable w_i is 1 (indicating that fuel should be brought from airport i), then the amount of fuel purchased at airport i must be less than or equal to the tank capacity of the aircraft.

$$x_i \leq tc * v_i \quad , \forall\, i \in A$$

If the binary variable w_i is 1, then the amount of fuel purchased at airport i must be greater than or equal to the minimum amount of fuel that must be bought in case fuel is needed.

$$x_i \geq 6.7 * bf * v_i \quad , \forall\, i \in A$$

AMPL code:

```
reset;
option solver cplex;

set A;  # Set of airports

param f{A};  # Fuel burn from airport i to the next airport
param C{A};  # Fuel cost at airport i
param r{A};  # Ramp fee at airport i
param w{A};  # Minimum gallon of fuel required to buy for waiving ramp fee at airport
i
param p{A};  # Number of passengers flying from airport i to the next airport
param pw;    # Passenger weight (per person)
param tc;    # Tank capacity of the aircraft
param rw;    # Maximum ramp weight of the aircraft
param lw;    # Maximum landing weight of the aircraft
param ow;    # Basic operating weight (BOW) of the aircraft
param lf;    # Minimum fuel required of the aircraft during landing
param uf;    # Fuel up level of the aircraft
param sf;    # Starting fuel of the aircraft from the 1st airport
param bf;    # (for 1b) Minimum amount of fuel that must be bought in case fuel is
needed

var x{i in A} >= 0;  # Amount of fuel to purchase (in gallons) at airport i
var y{i in A} binary;# Binary variable, indicating whether ramp fee is to be paid at
airport i
var z{i in A} >= 0;  # Remaining fuel amount (in gallons) after a trip completed at
airport i
var v{i in A} binary;# (for 1b) Binary variable, indicating whether fuel should be
brought from airport i

minimize TotalCost: sum{i in A} (6.7 * C[i] * x[i]) + sum{i in A} (r[i] * y[i]);

subject to RemainingFuel{i in A}: z[i] = x[i] + z[i] - f[i];
subject to MaxRampWeight{i in A}: ow + pw * p[i] + z[i] + x[i] <= rw;
subject to MaxLandingWeight{i in A}: ow + pw * p[i] + z[i] <= lw;
```

```
subject to InitialFuel: z['KCID'] = sf;
subject to FuelAtLastAirport: z['KTUL'] + x['KTUL'] >= sf;
subject to TankCapacity{i in A}: x[i] + z[i] <= tc;
subject to MinFuelForLanding{i in A}: z[i] >= lf;
subject to FuelPurchase{i in A}: x[i] >= 6.7 * w[i] * (1 - y[i]);
# for 1(b)
subject to extraConstr1{i in A}: x[i] <= tc * v[i];
subject to extraConstr2{i in A}: x[i] >= 6.7 * bf * v[i];

data Group10_HW5_Q1.dat;

solve;
display x;
display y;
display z;
display v;
```

Data file:

```
data;

set A := KCID KACK KMMU KBNA KTUL;

param:    f        C      r       w     p:=
KCID      5100     4      0       0      2
KACK      2200     8.32   800     600    4
KMMU      4700     8.99   750     500    8
KBNA      3800     6.48   600     650    8
KTUL      3600     9.27   800     500    8;

param  pw := 200;
param  tc := 14000;
param  rw := 36400;
param  lw := 31800;
param  ow := 22800;
param  lf := 2500;
param  uf := 4500;
param  sf := 7000;
param  bf := 200; # for 1b
```

Solution obtained from AMPL:

x [*] :=

KACK 2200

KBNA 3800

KCID 5100

KMMU 4700

KTUL 3600 ;

y [*] :=

KACK 1

KBNA 1

KCID 0

KMMU 0

KTUL 0 ;

z [*] :=

KACK 2500

KBNA 2500

KCID 7000

KMMU 2500

KTUL 3400 ;

v [*] :=

KACK 1

KBNA 1

KCID 1

KMMU 1

KTUL 1;

Question 2.

Some pairs of animals cannot be housed together due to natural aggression or incompatibility. The goal is to assign animals to enclosures such that incompatible animals are never in the same enclosure, and the total number of enclosures used is minimized.

This problem is modeled as a graph coloring problem, where:

- Each animal is a node.
- An edge connects any two animals that cannot be housed together.
- A color represents an enclosure.

The objective is to find the minimum number of colors (enclosures) needed so that no two adjacent nodes (conflicting animals) share the same color (enclosure).

Model Formulation

Let:

I be the set of animals.

K be the set of potential enclosures (max 10).

h[i,j] = 1 if animal i and j cannot be housed together.

X[i,k] = 1 if animal i is assigned to enclosure k.

E[k] = 1 if enclosure k is used.

Objective: Minimize the number of enclosures used: minimize sum over k of E[k]

Constraints:

1. Each animal must be assigned to exactly one enclosure:
   $\sum X[i,k] = 1 \ \forall \ i \in I$
2. If any animal is assigned to enclosure *k*, then that enclosure is marked as used:
   $\sum X[i,k] \leq |I| * E[k] \ \forall \ k \in K$
3. Incompatible animals cannot be assigned to the same enclosure:
   $X[i,k] + X[j,k] \leq 1$ for all (i,j) with h[i,j] = 1

The model is implemented using binary decision variables and solved using the CPLEX optimizer in AMPL. The model found an optimal solution using only 3 enclosures, meaning the minimum number of groups needed to safely house all animals is three. Below is the assignment of animals to enclosures:

Enclosure	Animals Assigned
1	a, c, d, g, i
2	b, e, f, h
3	j

Code:

```
reset;
option solver cplex;

# ----------------------------
# SETS
# ----------------------------

set I ordered;          # Set of animals
set K;                  # Set of possible enclosures (colors)
set S := {I, I};        # All possible animal pairs (used for constraints)

# ----------------------------
# PARAMETERS
# ----------------------------
```

```
param h{I, I} >= 0;     # Compatibility matrix: h[i,j] = 1 means i and j CANNOT be
together
param M >= 0;           # A large constant for big-M constraints (optional use if
needed)

# ----------------------------
# VARIABLES
# ----------------------------

var E{K} binary;        # E[k] = 1 if enclosure k is used
var X{I, K} binary;     # X[i,k] = 1 if animal i is assigned to enclosure k

# ----------------------------
# OBJECTIVE: Minimize number of enclosures used
# ----------------------------

minimize enc: sum {k in K} E[k];

# ----------------------------
# CONSTRAINTS
# ----------------------------

# Each animal must be assigned to exactly one enclosure
subject to assign_one_enclosure {i in I}:
    sum {k in K} X[i,k] = 1;

# If any animal is assigned to enclosure k, then E[k] must be 1
subject to activate_enclosure {k in K}:
    sum {i in I} X[i,k] <= card(I) * E[k];

# Animals that cannot be housed together (h[i,j] = 1) must not share the same
enclosure
subject to incompatible_animals {(i,j) in S, k in K}: X[i,k] + X[j,k] <=  1 + M * (1-
h[i,j]);

#DATA ----------------------------------------
data Group10_HW5_Q2.dat;


#COMMANDS -------------------------------------
solve;
display X;
```

Datafile:

```
set I := a b c d e f g h i j;
set K := 1 2 3 4 5 6 7 8 9 10;


param h:
        a b c d e f g h i j :=
      a 0 1 0 0 1 0 0 0 0 1
      b 1 0 0 1 0 0 1 0 0 0
      c 0 0 0 0 0 0 0 1 0 1
      d 0 1 0 0 0 1 0 0 0 0
      e 1 0 0 0 0 0 0 0 1 0
```

```
f 0 0 0 1 0 0 0 0 0 1
g 0 1 0 0 0 0 0 0 0 0
h 0 0 1 0 0 0 0 0 1 0
i 0 0 0 0 1 0 0 1 0 1
j 1 0 1 0 0 1 0 0 1 0;

param M := 20;
```

Result:

```
X [*,*]
:   1   2   3   4   5   6   7   8   9  10    :=
a   1   0   0   0   0   0   0   0   0   0
b   0   1   0   0   0   0   0   0   0   0
c   1   0   0   0   0   0   0   0   0   0
d   1   0   0   0   0   0   0   0   0   0
e   0   1   0   0   0   0   0   0   0   0
f   0   1   0   0   0   0   0   0   0   0
g   1   0   0   0   0   0   0   0   0   0
h   0   1   0   0   0   0   0   0   0   0
i   1   0   0   0   0   0   0   0   0   0
j   0   0   1   0   0   0   0   0   0   0
;
```

Question 3.

Decision Variables:

- x[g,t] – Amount of gasoline of type g stored in tank t
- b[g,t] – Binary Variable (indicates whether the gasoline of type g is in tank t (if no, 0)

Objective Function:

To minimize the total pumping cost (minimum cost storage plan)

$$\min \sum_{g \epsilon gastype,\ t \epsilon tanks} \frac{cost[g,t].x[g,t]}{1000}$$

Constraints:

- All the gasoline types need to be stored fully across all tanks.
- Total gasoline stored in each tank should not exceed its capacity.
- Each tank can only store only one type of gasoline.
- Ensuring that a gasoline type is assigned to a tank or not (Binary Decision). Binary variable determines if gasoline is stored in tank or not.

AMPL CODE:

```
reset;

option solver cplex;
```

```
# Defining sets
set gastype;
set tanks;

# Defining parameters
param gasCapacity{gastype};   #each gasoline type requirement
param tankCapacity{tanks};    #capacity of each tank
param cost{gastype, tanks};   #pumping cost per 1000 liters

# Decision Variables
var x{gastype, tanks} >= 0;   #Quantity of gasoline stored in a tank
var b{gastype, tanks} binary; #Binary Variable - 1 if gasoline is stored in tank,
otherwise 0

# Objective Function
minimize TotalCost:
    sum {g in gastype, t in tanks} (cost[g, t] * x[g, t] / 1000);

#Constraints
#each gasoline type is fully stored
subject to GasStorage {g in gastype}:
    sum {t in tanks} x[g, t] = gasCapacity[g];

#tank capacity constraint - total gas stored in tank should not exceed max capacity
subject to TankCap {t in tanks}:
    sum {g in gastype} x[g, t] <= tankCapacity[t];

#each tank can only store one type of gas
subject to OnegasinTank {t in tanks}:
    sum {g in gastype} b[g, t] <= 1;

#linking - used tanks are marked
subject to Link {g in gastype, t in tanks}:
    x[g, t] <= tankCapacity[t] * b[g, t];

data Group10_HW5_Q3.dat;

solve;
display x;
display TotalCost;
```

DAT FILE:

```
data;

#defining sets
set gastype := A B C D E;
set tanks := t1 t2 t3 t4 t5 t6 t7 t8;

#gas capacity
param gasCapacity :=
    A 75000
    B 50000
```

```
    C 25000
    D 80000
    E 20000;

#tank capacity
param tankCapacity :=
    t1 25000
    t2 25000
    t3 30000
    t4 60000
    t5 80000
    t6 85000
    t7 100000
    t8 50000 ;

#pumping cost
param cost:
    t1  t2  t3  t4  t5  t6  t7  t8 :=
A   1   2   2   1   4   4   5   3
B   2   3   3   3   1   4   5   2
C   3   4   1   2   1   4   5   1
D   1   1   2   2   3   4   5   2
E   1   1   1   1   1   1   5   5;
```

OUTPUT:

```
ampl: model '/Users/anuu/Documents/Group10_HW5_P3.mod';
CPLEX 20.1.0.0: optimal integer solution; objective 320
56 MIP simplex iterations
0 branch-and-bound nodes
x [*,*] (tr)
:      A       B       C       D       E      :=
t1   25000       0       0       0       0
t2       0       0       0   25000       0
t3       0       0   25000       0       0
t4   50000       0       0       0       0
t5       0   50000       0       0       0
t6       0       0       0       0   20000
t7       0       0       0    5000       0
t8       0       0       0   50000       0
;

TotalCost = 320

ampl:
```

The computed optimal total pumping cost is 320. This means gasoline is allocated to tanks in such a way that the total cost of transferring fuel is minimized.

The output provides values for x[g,t], which indicate how much gasoline of type g stored in tank t.

This determines an efficient storage plan for We Got Gas! by minimizing the total pumping cost while ensuring all the gasoline types are stored within the given tank capacities

Question 4.

Galaxy Industries produces two products: Space Rays and Zappers. Each product yields revenue, but their production costs follow piecewise linear structures based on the number of units produced. The company must decide how many units of each to produce in order to maximize profit, given limited resources of plastic and labor, a total production cap, and a product mix restriction.

Objective: Maximize total profit, calculated as:

Profit = Revenue − Production Costs

The cost per unit decreases or increases depending on production volume segments (blocks). For each block, a binary decision variable determines whether the block is active, allowing modeling of piecewise costs.

Cost and Production Structure

Space Rays

- Revenue: $8 per unit
- Cost blocks:
  - 0–125 units: $1.50
  - 126–225 units: $1.05
  - 226–375 units: $0.95
  - 375+ units: $0.75
- Resources required: 2 lb plastic and 3 min labor/unit

Zappers

- Revenue: $5 per unit
- Cost blocks:
  - 0–50 units: $1.05
  - 51–125 units: $0.75
  - 125+ units: $1.50
- Resources required: 1 lb plastic and 4 min labor/unit

Constraints

- Total plastic available: 1000 lb
- Total labor available: 40 hours = 2400 minutes
- Total units (Space Rays + Zappers) ≤ 700
- Space Rays cannot exceed Zappers by more than 350 units

Results Overview

Profit: $3688

Space Rays (block 4): 437 units

Zappers (blocks 1–3): 126 units (50 + 75 + 1)

Total Production: 563 units

Resources Used

Plastic: 1000 lb

Labor: 2400 min

- The model ignored earlier cost blocks for Space Rays and directly produced in block 4, where the unit cost is lowest ($0.75), maximizing profit.
- Zappers production filled blocks 1 and 2 completely, then added just 1 unit in the costly block 3.
- All available plastic and labor were fully utilized, achieving an optimal balance of quantity and cost.

Code:

```
reset;
option solver cplex;

# ----------------------------
# PARAMETERS
# ----------------------------

# Space Ray piecewise segments
param max_ray1 := 125;
param max_ray2 := 100;
param max_ray3 := 150;
param max_ray4 := 800;

# Zapper piecewise segments
param max_zap1 := 50;
param max_zap2 := 75;
param max_zap3 := 600;

# Resource limits
param plastic_limit := 1000;
param labor_limit := 2400;  # 40 hours × 60 minutes
param prod_limit := 700;
param ray_zap_diff := 350;

# ----------------------------
# DECISION VARIABLES
# ----------------------------

# Space Rays (prod in 4 cost blocks)
var ray1 >= 0, <= max_ray1, integer;
var ray2 >= 0, <= max_ray2, integer;
var ray3 >= 0, <= max_ray3, integer;
var ray4 >= 0, <= max_ray4, integer;

# Binary flags for block usage
var b1 binary;
```

```
var b2 binary;
var b3 binary;

# Binary flags for zapper blocks
var z1 binary;
var z2 binary;

# Zappers (prod in 3 cost blocks)
var zap1 >= 0, <= max_zap1, integer;
var zap2 >= 0, <= max_zap2, integer;
var zap3 >= 0, <= max_zap3, integer;

# ----------------------------
# OBJECTIVE: Maximize Profit
# ----------------------------

maximize Profit:
    (8 - 1.5)*ray1 + (8 - 1.05)*ray2 + (8 - 0.95)*ray3 + (8 - 0.75)*ray4 +
    (5 - 1.05)*zap1 + (5 - 0.75)*zap2 + (5 - 1.5)*zap3;

# ----------------------------
# CONSTRAINTS
# ----------------------------

# Resource usage
subject to PlasticUse:
    2*(ray1 + ray2 + ray3 + ray4) + 1*(zap1 + zap2 + zap3) <= plastic_limit;

subject to LaborUse:
    3*(ray1 + ray2 + ray3 + ray4) + 4*(zap1 + zap2 + zap3) <= labor_limit;

# Total production cap
subject to TotalProd:
    (ray1 + ray2 + ray3 + ray4) + (zap1 + zap2 + zap3) <= prod_limit;

# Ray-Zap difference
subject to ProdDifference:
    (ray1 + ray2 + ray3 + ray4) - (zap1 + zap2 + zap3) <= ray_zap_diff;

# ----------------------------
# Piecewise Control – Space Rays
# ----------------------------

subject to Ray_Block2: ray2 <= max_ray2 * b2;
subject to Ray_Block3: ray3 <= max_ray3 * b3;
subject to Ray_Block4: ray4 <= max_ray4;

# Sequential block logic
subject to Ray_Order_b2: b2 <= b1;
subject to Ray_Order_b3: b3 <= b2;

# ----------------------------
# Piecewise Control – Zappers
# ----------------------------
```

```
subject to Zap_Block2: zap2 <= max_zap2 * z2;
subject to Zap_Block3: zap3 <= max_zap3;

# Sequential block logic
subject to Zap_Order_z2: z2 <= z1;

# ----------------------------
# SOLVE + OUTPUT
# ----------------------------

solve;

display Profit;
display ray1, ray2, ray3, ray4, b1, b2, b3;
display zap1, zap2, zap3, z1, z2;
```

Results:

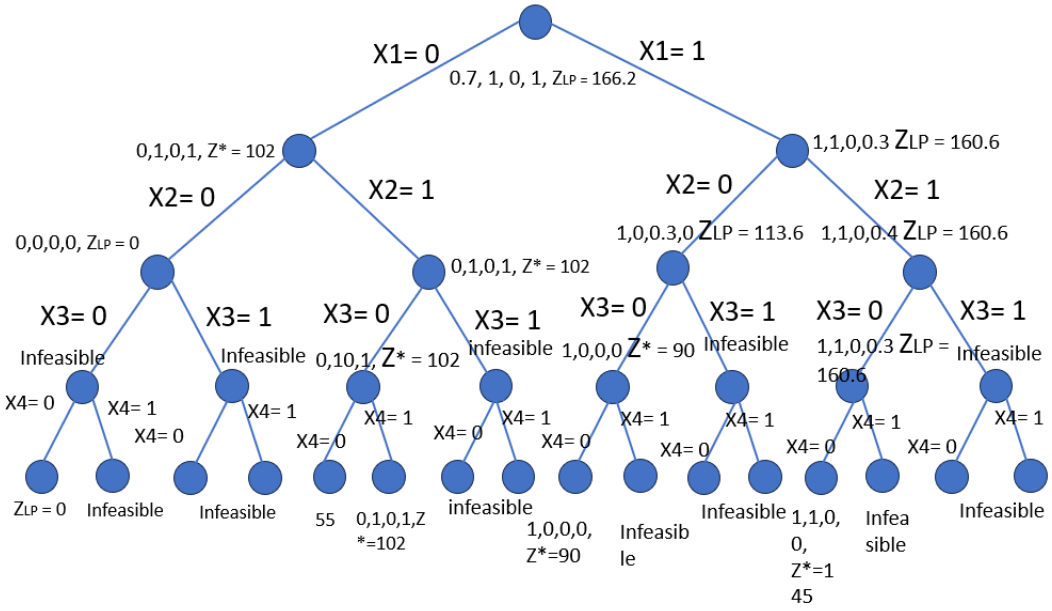
```
Profit = 3688

ray1 = 0
ray2 = 0
ray3 = 0
ray4 = 437
b1 = 1
b2 = 1
b3 = 1

zap1 = 50
zap2 = 75
zap3 = 1
z1 = 1
z2 = 1
```

Question 5.

The optimal value if we don't force integrality is 166.2. If we force integrality, the best value we find is 102.

$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1,$

$Z^* = 102$



AMPL Code:

```
reset;
option solver cplex;
# Sets
set Index;

# Parameters
param Revenue{Index} >= 0;
param RH{Index} >= 0;
param ConstraintCoeff{Index, Index};

# Binary decision variables
# var x{Index} binary;
var x{Index} >= 0 <= 1;
# Objective function
maximize TotalProfit:
    sum {i in Index} (Revenue[i] * x[i]);

# Constraints
subject to Constraint1 {i in Index}:
     sum {j in Index}(ConstraintCoeff[i,j] * x[j]) <= RH[i];

#subject to branch_1:
    # x[1]=0;
#subject to branch_1:
     #x[1]=1;
#subject to branch_2:
     #x[2]=0;
subject to branch_3:
     x[2]=1;
#subject to branch_4:
     #x[3]=0;
#subject to branch_5:
     #x[3]=1;
```

```
#subject to branch_6:
     #x[4]=0;
subject to branch_7:
     x[4]=1;

data HW5_p5.dat;

# Solve the optimization problem
solve;

# Display the optimal solution
display x;
```

Data file:

```
data;
set Index := 1 2 3 4;

param Revenue :=
    1 90
    2 55
    3 63
    4 47;

param RH :=
    1 10
    2 1
    3 0
    4 0;

param ConstraintCoeff :

       1 2 3  4:=
    1  7 2 8  3
    2  0 0 1  1
    3 -1 0 1  0
    4  0 -1 0 1   ;
```

Solution obtained from AMPL:

x [*] :=

1 0

2 1

3 0

4 1 ;