Assessment Task 4: Problem solving task 3

Q1)

a) Linear programming is used for obtaining the most optimal solution for a problem with given constraints. In linear programming, we formulate our real-life problems into a mathematical model. It involves an objective function, linear inequalities with is subjected to constraints. In our current model factories can apply linear programming to their work by determining the number of dresses and coats to be produced that would maximize the profit. In Big corporates linear programming provides a powerful tool to aid in planning for the number of resources to be utilized to generate an optimum amount of result in our case being the dresses and coats the factory produces for a chain of departmental stores in Victoria. The constraints here may include number of workers, hours of production, minimum demand, time required for production of each unit and units' profits for each unit. Mathematical modelling provides assistance to calculate the quantity of each item in order to maximize the company profit.

b) For formulating a linear programming model, we have: -

Let us assume that the factory produces x number of dresses and y number of coats.

Now since the factory works 8 hours a day, i.e. the amount of production hours in the factory is 8 hours which is 480 minutes. (since the production times in the table are given in minutes.

Now for the maximum number of workers, we have:

- The factory employs 25 workers in the cutting department and the factory operates for 8 hours so, the total amount of hours cutting factory works in minutes are: -
  25 x 480(8 hours) = 12,000
  So, the equation for cutting department becomes,
  →25x + 12y<=12000
- The factory employs 52 workers in the sewing department and the factory operates for 8 hours so, the total amount of hours cutting factory works in minutes are: -
  52 x 480(8 hours) = 24,960
  So, the equation for cutting department becomes,
  →25x + 55y<=24960
- The factory employs 14 workers in the packaging department and the factory operates for 8 hours so, the total amount of hours cutting factory works in minutes are: -
  14 x 480(8 hours) = 6720
  So, the equation for cutting department becomes,
  →15x + 15y<=6720

Now there is a daily demand for at least 120 dresses,

So, x>=120.

And as the number of coats and dresses cannot be less than 0, so the nonnegative constraints are:-

y>=0.

Now, for maximizing the profits we have, the objective function as:

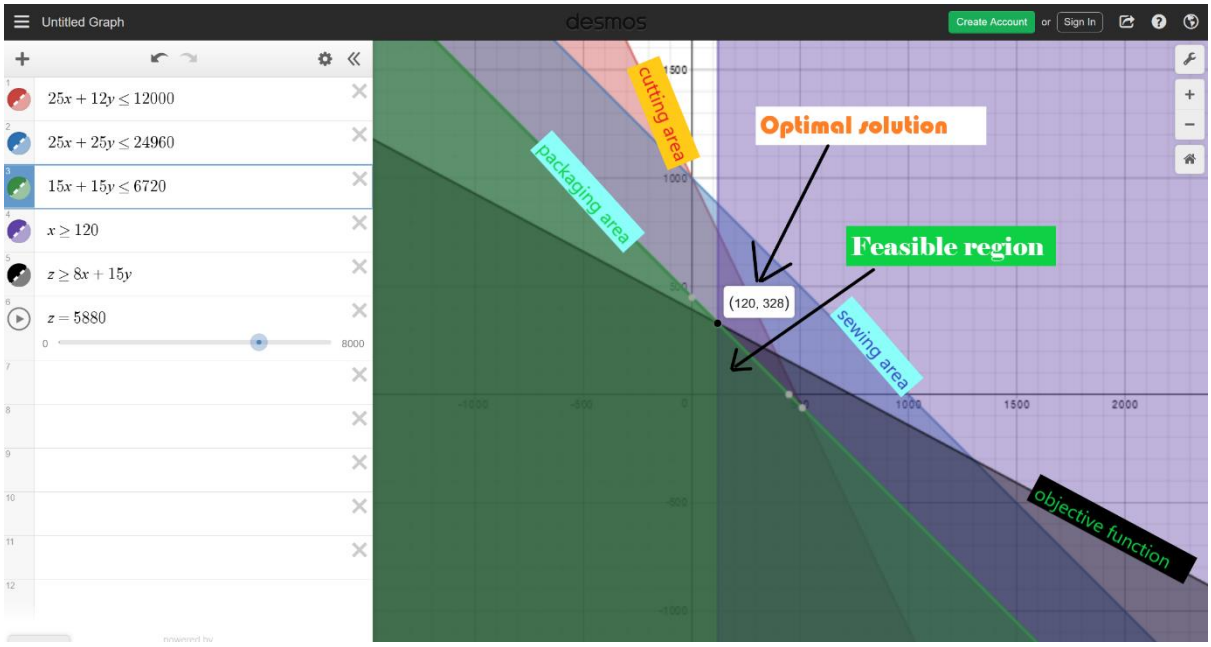
→Z(max)>= 8x + 15y

So, after evaluating these equations we have 120 dresses and 328 coats that would maximize the profit value that would be ($) 5880 as the optimum solution.

c)I have used the graphical method to evaluate the following linear programming model and derived an optimal solution on the graph which is as follows:

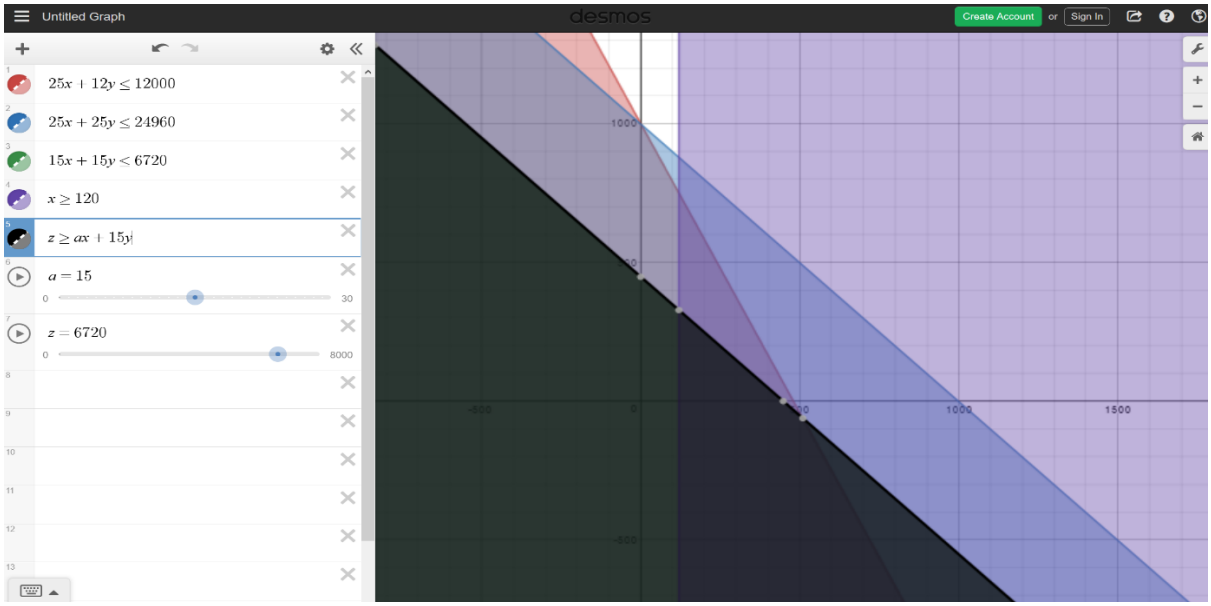
The graph is shown below, and the feasible region is shown with an arrow i.e. the region below the green line.

As we can see the maximum value of z i.e. the maximum profit here is ($)5880 and the solution for number of dresses and coats are 120 and 328 respectively.



d)So, the maximum profit margin we can obtain for a dress is if we charge $15 per dress which would give a profit margin of $6720 which could be changed while keeping the optimal solution obtained above that is 120 dresses and 328 coats.

The graph for this is Shown below:



Q2)

a) LP model

Decision variable be x_{ij} where i is the various ingredients used to produce j Cereals.

For i=1,2,3,4 ; j=1,2,3

With these variables in mind we have,

Formulating equations for Ingredients: -

Oates x11+x12+x13

Raisins x21+x22+x23

Apricots x31+x32+x33

Hazelnuts x41+x42+x43

Equations for Cereals can be formulated as: -

Cereal A x11+x21+x31+x41

Cereal B x12+x22+x32+x42

Cereal C x13+x23+x33+x43

Revenue generated through sales per ton

1300(x11+x21+x31+x41) +1150(x12+x22+x32+x42) +1600(x13+x23+x33+x43)

Purchase price of ingredients

100(x11+x12+x13) +90(x21+x22+x23) +110(x31+x32+x33) +200(x41+x42+x43)

Production cost of cereals

4.20(x11+x21+x31+x41) +2.60(x12+x22+x32+x42) +3.0(x13+x23+x33+x43)

Now, in order to calculate the objective function, we have: -

→**Max profit = The revenue generated-Purchase price of ingredients- production cost of cereals.**

1295.8(x11+x21+x31+x41)+1147.4(x12+x22+x32+x42)+1597(x13+x23+x33+x43) –{ 100(x11+x12+x13) +90(x21+x22+x23) +110(x31+x32+x33) +200(x41+x42+x43)}

Maxprofit=>1195.8x11+1205.8x21+1185.8x31+1095.8x41+1047.4x12+1057.4x22+1037.4x32+947.4x42+1497x13+1507x23+1487x33+1397x43

The other equation constraints here are as:

→0.8x11+0.1 x21+ 0.05 x31+ 0.05 x41 >=1000

→0.6 x12+0.25 x22+0.05 x32+0.1 x42>=800

→0.45 x13+0.15 x23+0.1 x33+0.3 x43>=750

The optimal mix of oats, raisins, Apricots and hazelnuts in terms of boxes is 5000,2500,1000 and 1000 Boxes respectively.

And the optimal mix of cereals is as 500 boxes of cereal A,400 boxes of cereal B and 2443 boxes of cereal C.

b)The R-code is as follows:

```
library(lpSolveAPI)

Food <- make.lp(0, 12)

lp.control(Food, sense= "maximize")

set.objfn(Food, c(1195.8,1047.4,1497,1205.8,1057.4,1507,1185.8,1037.4,1487,1095.8,947.4,1397))

add.constraint(Food, c(0.8,0.1,0.05,0.05,0,0,0,0,0,0,0,0),">=",1)

add.constraint(Food, c(0,0,0,0,0.6,0.25,0.05,0.1,0,0,0,0),">=",0.8)

add.constraint(Food, c(0,0,0,0,0,0,0,0,0.45,0.15,0.1,0.3),">=",0.75)

add.constraint(Food, c(1,1,1,0,0,0,0,0,0,0,0,0),"<=",10)

add.constraint(Food, c(0,0,0,1,1,1,0,0,0,0,0,0),"<=",5)

add.constraint(Food, c(0,0,0,0,0,0,1,1,1,0,0,0),"<=",2)

add.constraint(Food, c(0,0,0,0,0,0,0,0,0,1,1,1),"<=",2)

RowNames <- c("Constraint 1", "Constraint 2", "Constraint
3","Constraint4","Constraint5","Constraint6","Constraint7")

ColNames <- c("x11","x21","x31","x41","x12","x22","x32","x42","x13","x23","x33","x43")

dimnames(Food) <- list(RowNames, ColNames)

solve(Food)

Food

get.objective(Food)

get.variables(Food)

get.constraints(Food)
```

→the maximum profit that can be obtained in this model is $28072.2.

Q3)

(a)This is a two-player-zero-sum game as first of all it consists of two players namely John and Alice and a zero-sum game is such that one person's gain is equivalent to another's loss, so the net change in wealth or benefit is zero. So, as we can see here or for each comparison, the player with more chips in the pile will score 1 point and simultaneously the opponent will lose 1 point and so each participant's gain or loss of utility is exactly balanced by the losses or gains of the utility of the other participant. Thus, resulting in the net change of profit and loss to be zero or balanced resulting in a zero-sum game.

(b)The payoff matrix is as follows: -

Alice/John	(0,5)	(1,4)	(2,3)	(3,2)	(4,1)	(5,0)
(0,4)	-1	-1	0	0	-1	-1
(1,3)	0	-1	-1	-1	-1	0
(2,2)	0	0	-2	-2	0	0
(3,1)	0	-1	-1	-1	-1	0
(4,0)	-1	-1	0	0	-1	-1

(c) A saddle point is an element of the matrix such that it is the minimum element in its row and maximum in its column. A necessary and enough condition for a saddle point to exist is the presence of a payoff matrix element which is both a minimum of its row and a maximum of its column. A game may have more than one saddle point, but all must have the same value. Saddle point in game theory refers to a value of a function of two player which is a maximum with respect to one and a minimum with respect to the other.

No, this game does not have a saddle point as the Min of max of Alice from the table value is 0 and the Max of Min value of John from the table is -1.

Hence, L ≠ U so it is not a pure strategy will not result in an equilibrium and so, the game doesn't reach a saddle point.

(d)The linear programming model for this problem would be:

As *L*=0 and *U*=-1, the value of the game (*v*) is somewhere between [0, -1] (i.e. between -1 and 0 inclusive).

- ❖ Using Linear Programming for Player Alice's

Suppose Player I chooses the mixed strategy (x1, x2, x3, x4, x5)

- If Player II chooses strategy 1, the expected payoff is (−*x*1-X6)
- If Player II chooses strategy 2, the expected payoff is (-x1-*x*2-x4-x5)
- If Player II chooses strategy 3, the expected payoff is (-x2−2*x*3-*x*4)
- If Player II chooses strategy 4, the expected payoff is (−*x*2-2*x*3−*x*4)
- If Player II chooses strategy 5, the expected payoff is (−*x*1-*x*2−*x*4-x5)
- If Player II chooses strategy 6, the expected payoff is (−*x*1 -x5)

Since Player II wants to minimize Player I's payoff, Player II will want to choose a strategy that makes Player I's reward equal to *min*(−*x*1-X6,-x1-*x*2-x4-x5,-x2−2*x*3-*x*4,−*x*2-2*x*3−*x*4,−*x*1-*x*2−*x*4-x5,−*x*1 -x5). Then Player I should choose (*x*1,*x*2,*x*3,*x4,x5*) to make *min*(−*x*1-X6,-x1-*x*2-x4-x5,-x2−2*x*3-*x*4,−*x*2-2*x*3−*x*4,−*x*1-*x*2−*x*4-x5,−*x*1 -x5) as *large as possible*.

→And, to ensure all our proportions add up to 1, so the result can be written as a percentage.

*x*1+*x*2+*x*3_x4_x5=1 ; *xi*≥0, ∀*i*=1,2,3. - this is to ensure that all our proportions are positive (as we cannot have negative proportions).

- ❖ Using Linear Programming for Player II:

Suppose Player II chooses the mixed strategy (y1,y2,y3,y4,y5,y6):

If Player I chooses strategy 1, the expected reward is (−y1-y2−y5-y6)

If Player I chooses strategy 2, the expected reward is (−y2−y3-y4-y5)

If Player I chooses strategy 3, the expected reward is (-2y3−2y4)

If Player I chooses strategy 4, the expected reward is (-y2−y3-y4-y5)

If Player I chooses strategy 5, the expected reward is (-y1−y2-y5-y6)

Player I will choose a strategy to ensure that they obtain an expected reward of max(−y1-y2−y5-y6,−y2−y3-y4-y5,-2y3−2y4,-y2−y3-y4-y5,-y1−y2-y5-y6). Therefore Player II should choose (y1,y2,y3,y4,y5) to make max(−y1-y2−y5-y6,−y2−y3-y4-y5,-2y3−2y4,-y2−y3-y4-y5,-y1−y2-y5-y6) as small as possible.

→The two additional constraints there in the above LP which are there for similar reasons as in Player I's LP:

$y1+y2+y3=1$; $yi \geq 0, \forall i=1,2,3$

(e)Appropriate code to solve the linear programming model has been attested with this pdf document. I have also listed the R-code below:

```
# MIXED STRATEGIES #

# Player Alice's game #

library(lpSolveAPI)

lprec <- make.lp(0, 6)

lp.control(lprec, sense= "maximize")

set.objfn(lprec, c(0, 0, 0,0,0, 1))

add.constraint(lprec, c(1,0,0,0,1, 1), "<=", 0)

add.constraint(lprec, c(1,1,0,1,1, 1), "<=", 0)

add.constraint(lprec, c(0,1,2,1,0, 1), "<=", 0)

add.constraint(lprec, c(0,1,2,1,0, 1), "<=", 0)

add.constraint(lprec, c(1,1,0,1,1, 1), "<=", 0)

add.constraint(lprec, c(1,0,0,0,1, 1), "<=", 0)

add.constraint(lprec, c(1,1,1,1,1,0), "=", 1)

set.bounds(lprec, lower = c(0, 0, 0, 0, 0, -Inf))
```

```
RowNames <- c("Row1", "Row2", "Row3","Row4","Row5","Row6","Row7")
ColNames <- c("x1", "x2", "x3","x4","x5", "v")
dimnames(lprec) <- list(RowNames, ColNames)
solve(lprec)
get.objective(lprec)
get.variables(lprec)
get.constraints(lprec)
lprec
```

```
# Player John's's game #
lprec <- make.lp(0, 7)
lp.control(lprec, sense= "minimize")
set.objfn(lprec, c(0, 0, 0,0,0,0, 1))
add.constraint(lprec, c(1,1,0,0,1,1, 1), ">=", 0)
add.constraint(lprec, c(0,1,1,1,1,0, 1), ">=", 0)
add.constraint(lprec, c(0,0,2,2,0,0, 1), ">=", 0)
add.constraint(lprec, c(00,1,1,1,1,0, 1), ">=", 0)
add.constraint(lprec, c(01,1,0,0,1,1, 1), ">=", 0)
add.constraint(lprec, c(1,1,1,1,1,1,0), "=", 1)
set.bounds(lprec, lower = c(0, 0, 0,0,0,0, -Inf))
RowNames <- c("Row1", "Row2", "Row3","Row4","Row5","Row6")
ColNames <- c("y1", "y2", "y3","y4","y5","y6","v")
dimnames(lprec) <- list(RowNames, ColNames)
solve(lprec)
get.objective(lprec)
get.variables(lprec)
get.constraints(lprec)
lprec
```

(f) The game is solved for Alice has been laid out in the R-code.