**MP-1 PRACTICAL-2**

1. **Develop a python program to demonstrate Two Phase Simplex method in Linear Programming.**

**QUESTION:**

Minimize: z = x1 + x2 + x3 + x4 + x5  
Subject to:

3x1 + 2x2 + x3               = 1  
                 5x1 +  x2 +  x3 + x4        = 3  
                 2x1 + 5x2 + x3 +       x5  = 4

Solve using two-phase simplex method in python.

**Code**

def printTableu(tableu):

 print ('----------------------')

 for row in tableu:

  print (row)

 print ('----------------------')

 return

def pivotOn(tableu, row, col):

 j = 0

 pivot = tableu[row][col]

 for x in tableu[row]:

  tableu[row][j] = tableu[row][j] / pivot

  j += 1

 i = 0

 for xi in tableu:

  if i != row:

   ratio = xi[col]

   j = 0

   for xij in xi:

    xij -= ratio \* tableu[row][j]

    tableu[i][j] = xij

    j += 1

  i += 1

 return tableu

# assuming tablue in standard form with basis formed in last m columns

def phase\_1\_simplex(tableu):

 THETA\_INFINITE = -1

 opt   = False

 unbounded  = False

 n = len(tableu[0])

 m = len(tableu) - 2

 while ((not opt) and (not unbounded)):

  min = 0.0

  pivotCol = j = 1

  while(j < (n-m)):

   cj = tableu[1][j]

   if (cj < min):

    min = cj

    pivotCol = j

   j += 1

  if min == 0.0:

   opt = True

   continue

  pivotRow = i = 0

  minTheta = THETA\_INFINITE

  for xi in tableu:

   if (i > 1):

    xij = xi[pivotCol]

    if xij > 0:

     theta = (xi[0] / xij)

     if (theta < minTheta) or (minTheta == THETA\_INFINITE):

      minTheta = theta

      pivotRow = i

   i += 1

  if minTheta == THETA\_INFINITE:

   unbounded = True

   continue

  tableu = pivotOn(tableu, pivotRow, pivotCol)

 return tableu

def simplex(tableu):

 THETA\_INFINITE = -1

 opt   = False

 unbounded  = False

 n = len(tableu[0])

 m = len(tableu) - 1

 while ((not opt) and (not unbounded)):

  min = 0.0

  pivotCol = j = 0

  while(j < (n-m)):

   cj = tableu[0][j]

   if (cj < min) and (j > 0):

    min = cj

    pivotCol = j

   j += 1

  if min == 0.0:

   opt = True

   continue

  pivotRow = i = 0

  minTheta = THETA\_INFINITE

  for xi in tableu:

   if (i > 0):

    xij = xi[pivotCol]

    if xij > 0:

     theta = (xi[0] / xij)

     if (theta < minTheta) or (minTheta == THETA\_INFINITE):

      minTheta = theta

      pivotRow = i

   i += 1

  if minTheta == THETA\_INFINITE:

   unbounded = True

   continue

  tableu = pivotOn(tableu, pivotRow, pivotCol)

 return tableu

def drive\_out\_artificial\_basis(tableu):

 n = len(tableu[0])

 j = n - 1

 isbasis = True

 while(j > 0):

  found = False

  i = -1

  row = 0

  for xi in tableu:

   i += 1

   if (xi[j] == 1):

    if (found):

     isbasis = False

     continue

    elif (i > 1):

     row = i

     found = True

   elif (xi[0] != 0):

    isbasis = False

    continue

  if (isbasis and found):

   if (j >= n):

    tableu = pivotOn(tableu, row, j)

   else:

    return tableu

  j -= 1

 return tableu

def two\_phase\_simpelx(tableu):

 infeasible  = False

 tableu = phase\_1\_simplex(tableu)

 sigma = tableu[1][0]

 if (sigma > 0):

  infeasible  = True

  print ('infeasible')

 else:

  #sigma is equals to zero

  tableu = drive\_out\_artificial\_basis(tableu)

  m = len(tableu) - 2

  n = len(tableu[0])

  n -= m

  tableu.pop(1)

  i = 0

  while (i < len(tableu)):

   tableu[i] = tableu[i][:n]

   i += 1

  tableu = simplex(tableu)

 return tableu

def getTableu(c, eqs, b):

 #assume b >= 0 so if there is any b[i] negative make sure to enter

 #it possitive by multiplying (-1 \* eqs[i]) and (-1 \* b[i]) for all i

 tableu = []

 m = len(eqs)

 n = len(c)

 c.insert(0, 0.0)

 artificial = []

 sigma = [0.0]

 i = 0

 while (i < n):

  sigma.append(0.0)

  i += 1

 i = 0

 while (i < m):

  artificial.append(0.0)

  sigma.append(1.0)

  i += 1

 c.extend(artificial)

 tableu.append(c)

 tableu.append(sigma)

 i = 0

 for eq in eqs:

  eq.insert(0, b[i])

  eq.extend(artificial)

  eq[n+1+i] = 1.0

  tableu.append(eq)

  i += 1

 i = 0

 for xi in tableu:

  if (i > 1):

   j = 0

   for xij in xi:

    tableu[1][j] -= xij

    j += 1

  i += 1

 return tableu

c = [ 1.0, 1.0, 1.0, 1.0, 1.0,]

eq1 = [ 3.0 ,  2.0 , 1.0 ,  0.0,  0.0]

eq2 = [ 5.0 ,  1.0 , 1.0 ,  1.0,  0.0]

eq3 = [ 2.0 ,  5.0 , 1.0 ,  0.0,  1.0]

b = [1.0 , 3.0 , 4.0]

eqs = []

eqs.append(eq1)

eqs.append(eq2)

eqs.append(eq3)

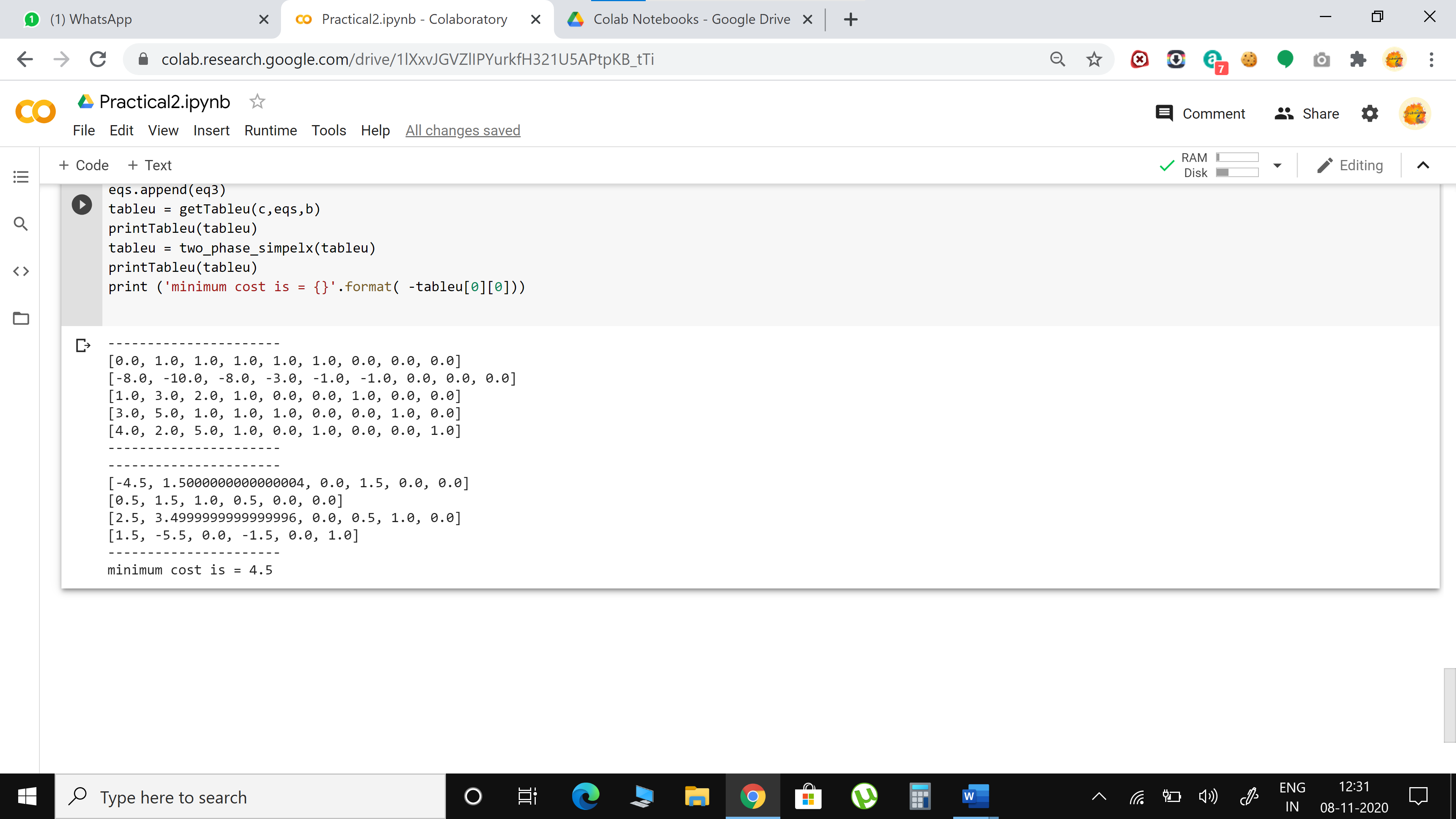
tableu = getTableu(c,eqs,b)

printTableu(tableu)

tableu = two\_phase\_simpelx(tableu)

printTableu(tableu)

print ('minimum cost is = {}'.format( -tableu[0][0]))

****