HW 4

Steps for installing PuLP

In [1]:
```
import sys
!{sys.executable} -m pip install pulp
```

```
Collecting pulp
  Using cached PuLP-2.3-py3-none-any.whl (40.6 MB)
Collecting amply>=0.1.2
  Downloading amply-0.1.4-py3-none-any.whl (16 kB)
Requirement already satisfied: pyparsing in /opt/conda/lib/python3.8/site-pa
ckages (from amply>=0.1.2->pulp) (2.4.7)
Requirement already satisfied: docutils>=0.3 in /opt/conda/lib/python3.8/sit
e-packages (from amply>=0.1.2->pulp) (0.15.2)
Installing collected packages: amply, pulp
Successfully installed amply-0.1.4 pulp-2.3
```

In [2]:
```
import pulp
```

In [3]:
```
# Import PuLP modeler functions
from pulp import *    # Here because of * we will not put `pulp' before eac
```

Steps for Decision Variables.

Set up the ranges and lists for rows and columns.

In [4]:
```
M=10
N=10
```

In [5]:
```
column=[j+1 for j in range(N)]
```

In [6]:
```
row=[i+1 for i in range(M)]
```

Define the vectors, matrices, vector variables using dictionary.

In [7]:
```
a= {i:{j:(-1)**(j+i)*(i-j) for j in column} for i in row}
```

In [8]:
```
b = {i: (-1)**(i)  for i in row}
```

In [9]:
```
# create index for columns
my_var = LpVariable.dicts("var", column, lowBound=0)
```

Define the Auxiliary variable

In [10]:
```
# Auxliliaryy variable
aux_var=LpVariable("aux_var", lowBound=0)
```

Define the auxiliary LP problem

In [11]:
```
probA = LpProblem("contraint_problem", LpMaximize)
```

Objective function for the auxiliary LP.

In [12]:
```
probA += -aux_var
```

Constraints for the auxiliary LP.

In [13]:
```
for i in row:
    probA += lpSum([a[i][j]*my_var[j] for j in column]) - aux_var<=b[i]
```

In [14]:
```
print(probA)
```

```
contraint_problem:
MAXIMIZE
-1*aux_var + 0
SUBJECT TO
_C1: - aux_var + 9 var_10 + var_2 - 2 var_3 + 3 var_4 - 4 var_5 + 5 var_6
 - 6 var_7 + 7 var_8 - 8 var_9 <= -1

_C2: - aux_var - var_1 - 8 var_10 + var_3 - 2 var_4 + 3 var_5 - 4 var_6
 + 5 var_7 - 6 var_8 + 7 var_9 <= 1

_C3: - aux_var + 2 var_1 + 7 var_10 - var_2 + var_4 - 2 var_5 + 3 var_6
 - 4 var_7 + 5 var_8 - 6 var_9 <= -1

_C4: - aux_var - 3 var_1 - 6 var_10 + 2 var_2 - var_3 + var_5 - 2 var_6
 + 3 var_7 - 4 var_8 + 5 var_9 <= 1

_C5: - aux_var + 4 var_1 + 5 var_10 - 3 var_2 + 2 var_3 - var_4 + var_6
 - 2 var_7 + 3 var_8 - 4 var_9 <= -1

_C6: - aux_var - 5 var_1 - 4 var_10 + 4 var_2 - 3 var_3 + 2 var_4 - var_5
 + var_7 - 2 var_8 + 3 var_9 <= 1

_C7: - aux_var + 6 var_1 + 3 var_10 - 5 var_2 + 4 var_3 - 3 var_4 + 2 var_5
 - var_6 + var_8 - 2 var_9 <= -1

_C8: - aux_var - 7 var_1 - 2 var_10 + 6 var_2 - 5 var_3 + 4 var_4 - 3 var_5
 + 2 var_6 - var_7 + var_9 <= 1

_C9: - aux_var + 8 var_1 + var_10 - 7 var_2 + 6 var_3 - 5 var_4 + 4 var_5
 - 3 var_6 + 2 var_7 - var_8 <= -1

_C10: - aux_var - 9 var_1 + 8 var_2 - 7 var_3 + 6 var_4 - 5 var_5 + 4 var_6
 - 3 var_7 + 2 var_8 - var_9 <= 1

VARIABLES
aux_var Continuous
var_1 Continuous
var_10 Continuous
var_2 Continuous
var_3 Continuous
var_4 Continuous
var_5 Continuous
var_6 Continuous
var_7 Continuous
var_8 Continuous
var_9 Continuous
```

Solve the auxiliary LP.

In [15]:

```python
# The problem is solved using PuLP's choice of Solver
probA.solve()
# The status of the solution is printed to the screen
print("Status:", LpStatus[probA.status])
```

```
Status: Optimal
```

Show the optimal value and determine feasiblity of the original constraints.

In [16]:
```
print("The optimal value = ", value(probA.objective))
```

```
The optimal value =  0.0
```

In [17]:
```
if value(probA.objective)==0:
    print("The original constraints are feasible")
else: print("The original constraints are not feasible")
```

```
The original constraints are feasible
```

Show the values for the optimal solution

In [18]:
```
# Each of the variables is printed with it's resolved optimum value
for a in probA.variables():
    print(a.name, "=", a.varValue)
```

```
aux_var = 0.0
var_1 = 0.0
var_10 = 0.0
var_2 = 0.14285714
var_3 = 0.0
var_4 = 0.0
var_5 = 0.0
var_6 = 0.0
var_7 = 0.0
var_8 = 0.0
var_9 = 0.14285714
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: