HW 2 Problem 1b.

the LP problem.

Max $25x_1 + 30x_2$

subject to.

$x_1 \leq 6000$

$x_2 \leq 4000$

$\frac{1}{200}x_1 + \frac{1}{140}x_2 \leq 40$

$x_1, x_2 \geq 0$

We import pulp, in the following two cells.

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

```
Collecting pulp
  Using cached PuLP-2.3-py3-none-any.whl (40.6 MB)
Processing /home/jupyter/.cache/pip/wheels/c6/62/13/fc20c94998f8157f
bd5582fa01d9d0661e5558d80ee0f11257/amply-0.1.2-py3-none-any.whl
Requirement already satisfied: pyparsing in /opt/conda/lib/python3.8
/site-packages (from amply>=0.1.2->pulp) (2.4.7)
Requirement already satisfied: docutils>=0.3 in /opt/conda/lib/pytho
n3.8/site-packages (from amply>=0.1.2->pulp) (0.15.2)
Installing collected packages: amply, pulp
Successfully installed amply-0.1.2 pulp-2.3
```

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

In [3]:
```
# Create a LP Minimization problem
Lp_prob = pulp.LpProblem('Your_LP_Problem', pulp.LpMaximize)  # We set
up theproblem using the command LpProblem in the PuLP package.
```

In [4]:
```
# Create problem Decision Variables
x_1 = pulp.LpVariable("x_1")
x_2 = pulp.LpVariable("y_1")
```

We used the LpVariable class.

We now set up our LP problem.

In [5]:
```
# Objective Function
Lp_prob += 25 * x_1 + 30 * x_2

# We put objective function first then constraints.

# Constraints:
Lp_prob +=  x_1 <= 6000
Lp_prob +=  x_2 <= 4000
Lp_prob += 1/200 * x_1 + 1/140 * x_2  <= 40
Lp_prob += x_1>= 0
Lp_prob += x_2>= 0
```

The objective function and constraints are added using the += operator to our model. The objective function is added first, then the individual constraints.

In [6]:
```
# Display the problem
print(Lp_prob)
```

```
Your_LP_Problem:
MAXIMIZE
25*x_1 + 30*y_1 + 0
SUBJECT TO
_C1: x_1 <= 6000

_C2: y_1 <= 4000

_C3: 0.005 x_1 + 0.00714285714286 y_1 <= 40

_C4: x_1 >= 0

_C5: y_1 >= 0

VARIABLES
x_1 free Continuous
y_1 free Continuous
```

You realize that the inequalities are rearranged to put numbers only in the right hand side.

We can solve this LP using the `solve` function. `Lp_prob.solve` means apply `solve` to the `Lp_prob` we defined.

In [7]:
```
Lp_prob.solve()
pulp.LpStatus[Lp_prob.status]
```

Out[7]: 'Optimal'

It solved the LP problem and gave the result: There are 5 status codes:

- Not Solved: Status prior to solving the problem.
- Optimal: An optimal solution has been found.
- Infeasible: There are no feasible solutions.
- Unbounded: The constraints are not bounded, maximising the solution will tend towards infinity.
- Undefined: The optimal solution may exist but may not have been found.

We can now view our optimal variable values and the optimal value of Z.

In [8]:
```
# Printing the final solution
print("x_1=", pulp.value(x_1), "x_2=", pulp.value(x_2), "z=", pulp.val
ue(Lp_prob.objective))
```

```
x_1= 6000.0 x_2= 1400.0 z= 192000.0
```

Another way to show the solutions:

In [9]:
```
for a in Lp_prob.variables():
        print(a.name, "=", a.varValue)
print("Optimal value is z = ", pulp.value(Lp_prob.objective))
```

```
x_1 = 6000.0
y_1 = 1400.0
Optimal value is z =  192000.0
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: