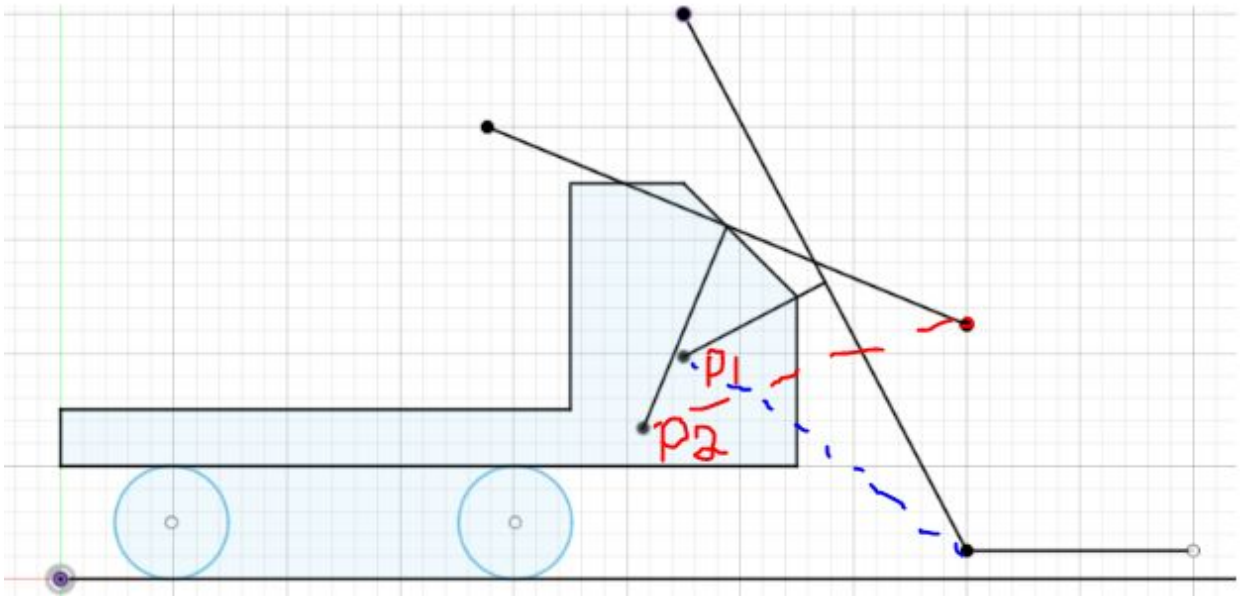
Peter Nikopoulos

Mechanical Question: Solve four bar mechanism

My first step was to draw the diagram in fusion 360.I drew lines from both the bottom and top points of each bucket position. Then I drew a perpendicular bisector from each of the previous lines. I selected the points at the following positions.

Point 1: (2.75, 0.984) with an arm that is 1.516 ft long. This arm is unpowered.
Point 2: (2.572,0.667) with an arm that is 1.5 ft long. This arm is powered.

I chose the position of point 1 so that arm1 does not cross over the toggle point. I chose arm 2 as the crank because it is shorter and at the lowest position it will always be angled slightly up, so as the arm raises the payload, the motor torque demands will only get smaller.



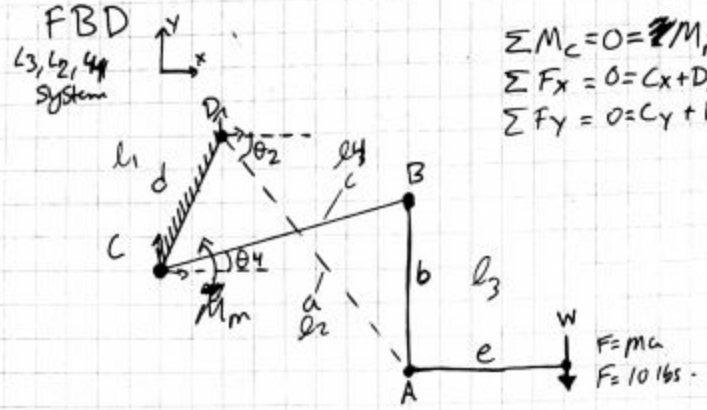
I used these measurements to set up a system of equations, my work is shown below.

Rho Beta Fourbar question.

$P_1 = (2.75, 0.984)$ $P_2 = (2.572, 0.667)$

Arm 1 = 1.516 ft = l_2 Arm 2 = 1.5 ft. = l_4 Crank arm

FBD L_3, L_2, L_4 System



$\Sigma M_C = 0 = M_m - W(e + c\cos\theta_4)$

$\Sigma F_x = 0 = C_x + D_x$

$\Sigma F_y = 0 = C_y + D_y - W$

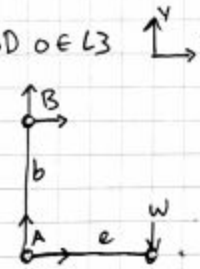
$\theta_2 = 22.5$ deg

$\theta_4 = 17.8$ deg

F = ma

F = 10 lbs.

FBD of L_3

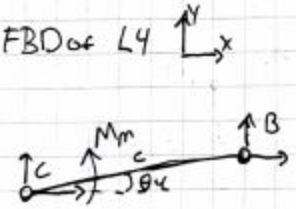


$\Sigma M_A = 0 = B_x \cdot b - W \cdot e$

$\Sigma F_x = 0 = B_x + A_x$

$\Sigma F_y = 0 = B_y + A_y - W$

FBD of L_4

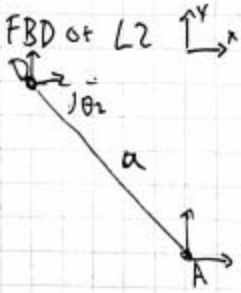


$\Sigma M_C = 0 = M_m + c\cos(\theta_4) \cdot B_x$

$\sum F_x = 0 = C_x + B_x$

$\sum F_y = 0 = C_y + B_y$

FBD of L_2



$\Sigma M_D = 0 = A_x \cdot a\sin(\theta_2) + A_y \cdot a\cos(\theta_2)$

$\Sigma F_x = 0 = D_x + A_x$

$\Sigma F_y = 0 = D_y + A_y$

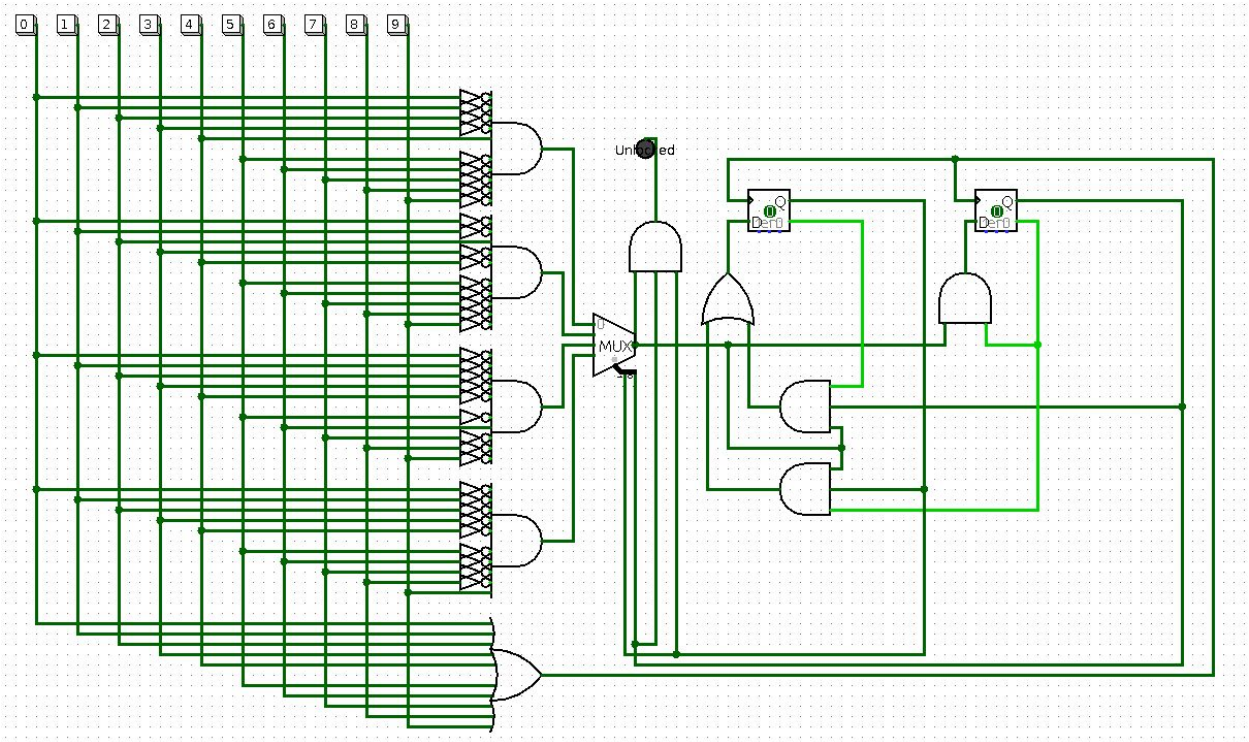
Finally I plugged these equations into a matlab script that solved for all unknown variables.

```
w = 10; %lbs
a = 1.516;
b = 1;
c = 1.5;
d = 0.363;
e = 1;
theta4 = deg2rad(17.8);
theta2 = deg2rad(22.5);

syms Mm Ax Ay Bx By Cx Cy Dx Dy
eq1  = Mm == w* (e+c*cos(theta4));
eq2  = Cx + Dx == 0;
eq3  = Cy + Dy == w;
eq4  = b * Bx == w * e;
eq5  = Bx + Ax == 0;
eq6  = By + Ay == w;
eq7  = Mm + c * cos(theta4) * Bx == 0;
eq8  = Cx + Bx == 0;
eq9  = Cy + By == 0;
%eq10 =
%eq11 =

[A,B] = equationsToMatrix([eq1,eq2,eq3,eq4,eq5,eq6,eq7,eq8,eq9], [Mm, Ax, Ay, Bx, By, Cx, Cy, Dx, Dy])

double(B(1)) % motor torque
```

```
A =

[ 1, 0, 0,                                  0, 0, 0, 0, 0, 0]
[ 0, 0, 0,                                  0, 0, 1, 0, 1, 0]
[ 0, 0, 0,                                  0, 0, 0, 1, 0, 1]
[ 0, 0, 0,                                  1, 0, 0, 0, 0, 0]
[ 0, 1, 0,                                  1, 0, 0, 0, 0, 0]
[ 0, 0, 1,                                  0, 1, 0, 0, 0, 0]
[ 1, 0, 0, 6432014367542989/4503599627370496, 0, 0, 0, 0, 0]
[ 0, 0, 0,                                  1, 0, 1, 0, 0, 0]
[ 0, 0, 0,                                  0, 1, 0, 1, 0, 0]


B =

 106793105419077/4398046511104
                              0
                             10
                             10
                              0
                             10
                              0
                              0
                              0


ans =

   24.2819
```

The script produces an output saying that the torque required to begin lifting the arm off the ground is **24.28 ft lbs.**

Electrical Question: Designing a lockbox
I designed my circuit using LOGISIM, I have attached the logism circuit for you to try.

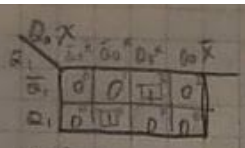


To create this circuit I used a D flip flop state machine that is connected to a LED to indicate the unlocking signal. The state machine uses any high signal on a clock pulse to proceed through the states. The state machine is designed so that the password must be inputted in the correct order, if not then it will drop back to the initial 00 state. Clock pulses are created anytime a button is pressed, this is done though the OR gate at the bottom left. The other gates on the left side will only produce a high signal if the proper button is pressed. The multiplexer selects the AND gate to listen to based on the current value of the state machine, which is piped directly from the D flip flops to the multiplexers selector pins. Inputting the correct code will cause the state machine to proceed though the 00, 01, 10, and 11 states, briefly pulse the unlock signal and return to the 00 state.

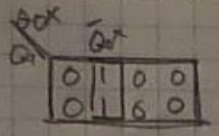
The password to unlock is 4269

My math for this problem is shown below.

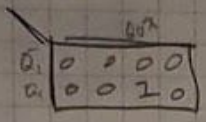
Q_1	Q_0	x	Q_1^*	Q_0^*	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1



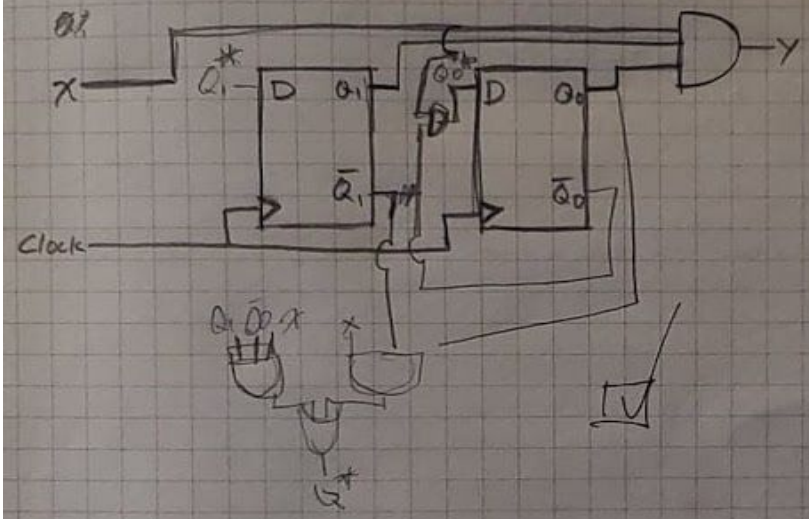
$$Q_1^* = Q_1 \cdot \bar{Q}_0 \cdot x + \bar{Q}_1 \cdot Q_0 \cdot x$$



$$Q_0^* = \bar{Q}_0 \cdot x$$



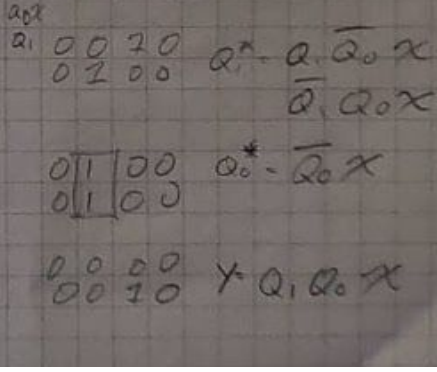
$$Y = Q_1 \cdot Q_0 \cdot x$$



Q_1	Q_0	x	Q_1^*	Q_0^*	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	1

0 1 3 2

4 5 7 6



$$Q_1^* = Q_1 \bar{Q}_0 x + \bar{Q}_1 Q_0 x$$

$$Q_0^* = \bar{Q}_0 x$$

$$Y = Q_1 Q_0 x$$

Software Question: Racket Interpreter

I programmed this interpreter in python and it is attached in this folder. I used a polling loop to get the user inputs and defined a recursive function to evaluate each expression that was inputted. This function is built to identify the operator and extract the values of the expression string into variables. The function takes in a string expression that starts with an open parentheses and ends with an equal number of closed parentheses. Knowing this we are able to build a method that calls itself on only a section of the original expression so that we can recursively solve each layer of the expression. Each time the method returns it passes on its answer the expression it was given, and eventually the program completes the recursive loop and the last method returns the final answer. For storing variables I used a dictionary to store the key and values, because of this implementation my interpreter can use any single word variable, not just single letters.