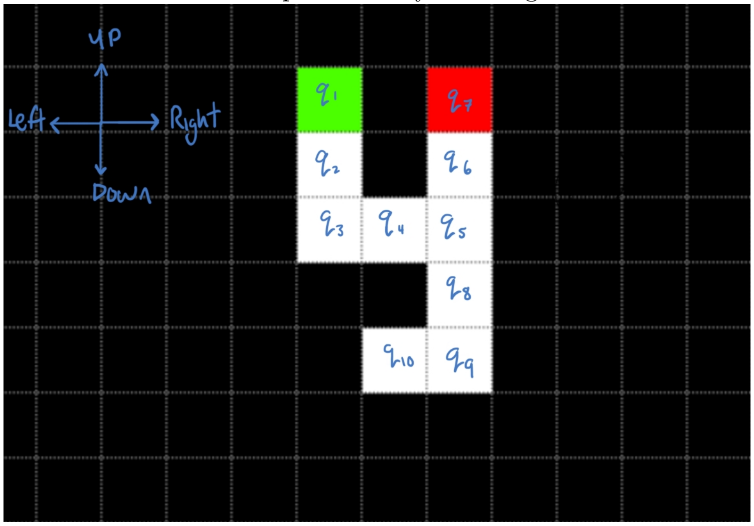
P1

Joe is a lonely farmer lost in a corn maze. Joe doesn't know how to find the end of the maze and needs a map to find his way out. Unfortunately for Joe, we are computer scientists and don't believe in maps. Instead, I have created a regular expression to represent all strings that will lead Joe to the exit of the maze, the expression is over the language Σ = L,R,D,U where L, R, D, U represents taking 1 step Left, Right, Down, or Up respectively. These directions are defined from our perspective and not Joe's.
Since it's dark out, Joe will not know if he reaches the end of the maze before the sun comes back out. Because of this, strings are only accepted if they result in Joe being at the end of the maze when the string has been fully read.

So, L = { x $\in$ (L+R+D+U)* : x results in Joe being at the end of the maze at the end of the string}

One more additional note, moving in the direction of a wall in the corn maze is a valid move, but will not change Joe's direction.
Let the corn maze be represented by this diagram:



The green square is where Joe will start.
The red square is the end of the maze (where Joe should be at the end of the string in order to be accepted).
Black squares represent walls.
White squares are valid places to move.
I have labeled the squares q_1-q_{10} so they can be referred to when constructing the regex / dsfa.
Consider the string DDRRUU, This string is in our language because the sequence of steps Down, Down, Right, Right, Up, Up leaves Joe at the end of the maze as wanted.

Proof 1:
I have defined the following smaller regexs to make this process easier to understand:

Q1 = (L+R+U)*D , represents the series of steps needed to make it from q_1 to q_2. (U+L+R)* means that Joe can move Up, Left, or Right as many times as he wants and he will not move from the q_1 D means that the move required to go from q_1 to q_2 is to go Down

Q2 = (L+R+UQ1)*D , represents the series of steps needed to make it from q_2 to q_3. (L+R+UQ1)* represents Joe's options of moving while ensuring that he remains in the square q_2. Joe can move Left or Right as many times as he wants as there are walls in both these directions. If Joe moves up, he is now and in q_1, so to make it back to q_2 he must follow the series of steps described by Q1. D means that the move required to go from q_2 to q_3 is to go Down.

Q3 = (L+D+UQ2)*R , represents the series of steps needed to make it from q_3 to q_4. Follows similar logic as the above

Q4 = (LQ3+D+U)*R , represents the series of steps needed to make it from q_4 to q_5. Follows similar logic as the above

Q5 = (LQ4+R+DQ8)*U , represents the series of steps needed to make it from q_5 to q_6. (LQ4+R+DQ8)* represents Joe's options of moving while ensuring that he ends up back in q_5. If Joe moves left, he goes back to q_4, so he must follow Q4 to get back. If Joe moves Right, he hits a wall and does not move from q_5. If Joe moves down, he goes to q_8, so he must follow Q8 to get back (see definition of Q8 below)

Q6 = (L+R+DQ5)*U , represents the series of steps needed to make it from q_6 to q_7. Follows similar logic as the above

Q7 = (L+R+DQ6+U)*, represents the series of steps needed to remain in q_7. This is the square we want to be in, so we ensure that once Joe gets here he always comes back, for this reason there is no move after making all moves that keep Joe in q_7, represented by (L+R+DQ6+U)*

Q8 = (L+R+DQ9)*U , represents the series of steps needed to make it from q_8 to q_5. Follows similar logic as the above .

Q9 = (LQ10+R+D)*U , represents the series of steps needed to make it from q_9 to q_8. Follows similar logic as the above

Q10 = (L+U+D)*R , represents the series of steps needed to move from q_{10} to q_9. Follows similar logic as the above

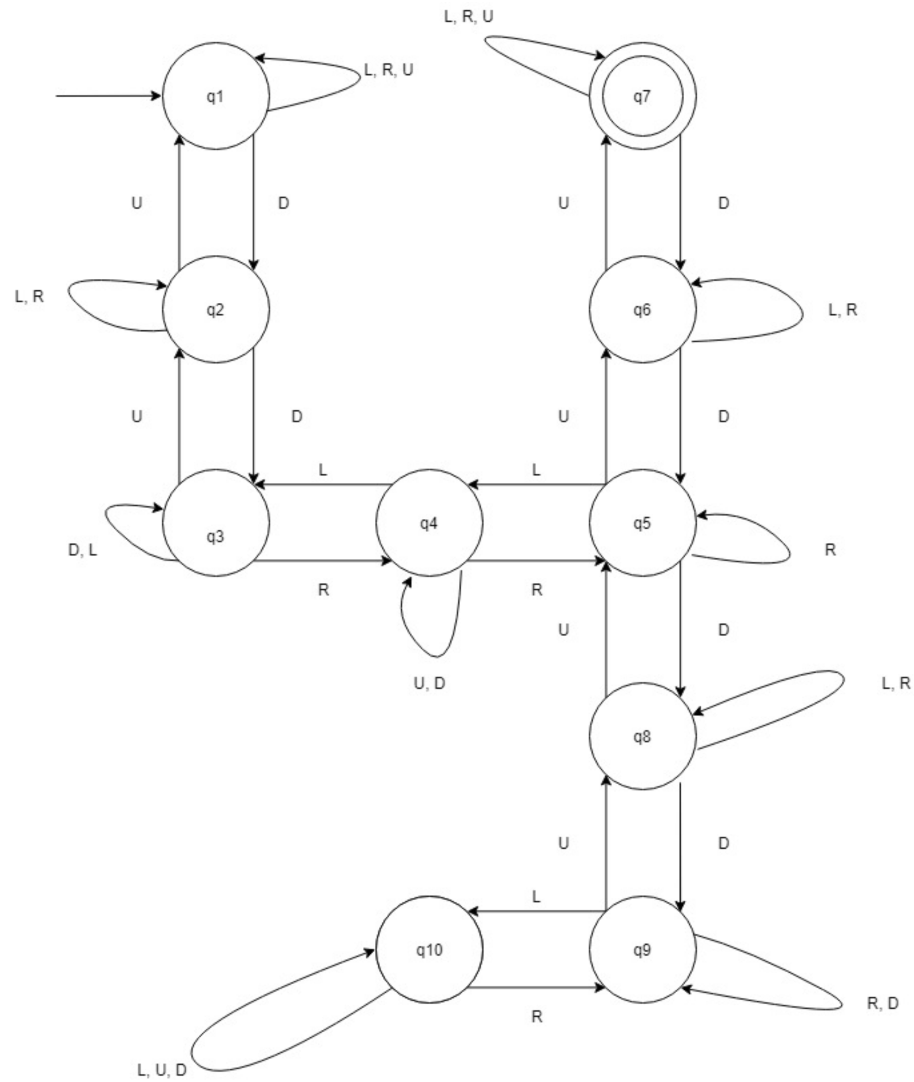
Here is the complete regex: Q1Q2Q3Q4Q5Q6Q7

Q1Q2Q3Q4Q5Q6Q7 represents the string of moves where Joe goes from square $q1 \to q2 \to ... \to q7$, the destination square, by the definition of each Qi i $\in$ [1,..,7]

Notice that the regex Q1Q2Q3Q4Q5Q6, accepts strings representing Joe reaching q7 only at the end of the string, instead of potentially reaching q7 multiple times and then finally coming back to it in the end.

This is what the full regex looks like:
(L+R+U)*D
(L+R+U(L+R+U)*D)*D
(L+D+U(L+R+U(L+R+U)*D)*D)*R
(L(L+D+U(L+R+U(L+R+U)*D)*D)*R+D+U)*R
(L(L(L+D+U(L+R+U(L+R+U)*D)*D)*R+D+U)*
R+R+D+(L+R+D+(L(L+U+D)*R+R+D)*U)*U)*U
(L+R+D(L(L(L+D+U(L+R+U(L+R+U)*D)*D)*R+D+U)*
R+R+D+(L+R+D+(L(L+U+D)*R+R+D)*U)*U)*U)*U
(L+R+D+(L+R+D(L(L(L+D+U(L+R+U(L+R+U)*D)*D)*R+D+U)*
R+R+D+(L+R+D+(L(L+U+D)*R+R+D)*U)*U)*U)*U+U)*

Proof 2:
Here is the diagram of the dsfa that represents this language:
The intuition is that we let each square that Joe can occupy represent a state, and the transitions are defined by the directions Joe can move in. So if Joe moves against a wall he stays in the same state, if Joe moves in the direction of another square he moves to that state. The only accepting state is q7 as that is the square Joe wants to be once the string terminates. When we define states this way, we are only in a specific state when Joe is in the corresponding square.



Justification:
$\delta * (q_1, x) =$
$\{$

q_1, if Joe finishes moving and ends up in square q_1
q_2, if Joe finishes moving and ends up in square q_2
q_3, if Joe finishes moving and ends up in square q_3
q_4, if Joe finishes moving and ends up in square q_4
q_5, if Joe finishes moving and ends up in square q_5
q_6, if Joe finishes moving and ends up in square q_6
q_7, if Joe finishes moving and ends up in square q_7
q_8, if Joe finishes moving and ends up in square q_8

q_9, if Joe finishes moving and ends up in square q_9
q_{10}, if Joe finishes moving and ends up in square q_{10}
}