Questions 1-4 should be completed on the Udacity site. The graders will download and evaluate your answers. The others should be typed and uploaded to T-Square (on-campus students may upload all answers to T-square if they prefer).

For problems 5-7 we are not looking for a formal transition function and inductive proof, but rather a detailed argument that convinces the reader of its correctness.

1. Complete the exercise of tracing through the configuration sequence for the given Turing machine .
https://www.udacity.com/course/viewer#!/c-ud557/l-1740638623/e-1740278552/m-1740278553
Completed

2. Program a Turing machine the shifts its input one square to the right and places a $ sign at the beginning of the tape. See
https://www.udacity.com/course/viewer#!/c-ud557/l-1740638623/e-1732309017/m-1740278556
Completed

3. Program a Turing machine that tests if the input string has an equal number of zeros and ones. See
https://www.udacity.com/course/viewer#!/c-ud557/l-1740638623/e-1740278557/m-1740278558
Completed

4. Program a two-tape Turing machine to perform substring search. See
https://www.udacity.com/course/viewer#!/c-ud557/l-1728138752/e-1718598811/m-1751158600
Completed

5. Another alternative Turing machine model has a single, one-way infinite tape, but two read-write heads. The transition function has the form $\delta: Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, R, S\}^2$, the same as a multi-tape machine. Describe how you would program such a machine to decide the language $L = \{ww \mid w \in \{0,1\}^*\}$.

- Determine if the string is even in length (because otherwise you can't have a string of the desired form)
    - Both heads are at the start of the tape
    - Move one of the heads two positions at a time to the right until a blank is read
    - When the head reads a blank, move left one position. If the entry is not a blank, then the string is even. Continue
    - Otherwise the string is odd. Reject.
- Shift everything right and write a blank to the first position of the tape (this was done in problem 2) so that the beginning and end of the string are marked
- Move both heads to the start of the tape
- Move both heads right one position so that they are both over the start of the input string
- Move one head (call it head2) right one position while the other (head1) stays put.
- Begin searching for first match
    - If the entries under the two heads agree, shift both heads right and move to the "found match" state
    - If the entries do not agree, then keep head1 in place and shift head2 right until a match is found
    - If head2 reads a blank, reject.
- Match found state: Continue matching the tape entries.
    - If the entries match, shift both heads right
    - If head2 reads a blank, accept
    - If the entries do not match, go to the rewind state
- Rewind state
    - Move head1 left one position. Head2 stays put
    - Move both heads left until head1 reads a blank
    - Return to "begin searching for first match" state

6. Suppose we have a one-tape Turing machine M whose head instead of having to move just left or right in each computation step, can move left or right or stay put. We called these "stay-put machine" in the lesson. Argue that it is possible to create a new standard Turing machine M' (no "stay put", just moves left and right) that recognizes the same language as M by changing only the transition function δ, keeping the same Q and Γ.

We discussed in lecture that adding a stay-put move adds no computational capability to the Turing machine because we can perform the same actions with a traditional Turing machine. However, in that case, we added a state to show this equivalence. In this problem, we don't necessarily want to show equivalence, but we do want to show that M' recognizes the same languages as M. In order to do that, we need to show that there are alternatives to the stay-put move. In other words, we need to show that the stay-put move is useless.

In order to do this, I will borrow an example presented in the discussion forum. Take these Turing machines M and M' that have a tape alphabet of {b,0,1,2} that determines if the sum of the tape digits is even or odd.

M	**M'**
q_0: b → b,R,q_a	q_0: b → b,R,q_a
0 → 0,R,q_0	0 → 0,R,q_0
1 → 1,R,q_1	1 → 1,R,q_1
2 → 2,S,q_1	2 → 2,R,q_0
q_1: b → b,R,q_r	q_1: b → b,R,q_r
0 → 0,R,q_1	0 → 0,R,q_1
1 → 1,R,q_0	1 → 1,R,q_0
2 → 1,S,q_0	2 → 2,R,q_1

At the end of computation, M and M' will have different tape contents, so they are not equivalent. However, they end in the same state, so M' is able to recognize the same languages as M.

So we have shown that the stay-put action is useless for one machine. Now we need to show that it is useless for all machines. When you use a stay-put move, you have three options to avoid going into an infinite loop:

1. Write a different symbol to the tape and go to a different state
2. Write the same symbol to the tape and go to a different state
3. Write a different symbol to the tape and stay in the same state

In the example above, we used options 1 and 2 for machine M. We were able to remove the stay-put move in machine M' by skipping this state transition. I will state that a similar change is possible for all such Turing machines that use a stay-put operation. We can use a "shortcut" to go directly from the state before the stay-put to the state after the stay-put. The tape contents may have to be different to account for this, but they can recognize the same languages.

7. Suppose that we constrained the standard 1-tape Turing machine to only be able to change the symbol on each square two times (clarification: overwriting a symbol with itself doesn't +count; overwriting a blank symbol with a non-blank symbol does). Prove that this model can decide every language that a standard Turing machine can.

- Shift everything right and put a marker at the beginning of the tape (something not in the original alphabet. Call this symbol the input divider).
- Do your processing
- If you need to write to a position, write a super special symbol in this place, remember the character you wanted to write, and go to the copy state
    - (Again, just to clarify, this writing is when you want to write a character different from what is currently on the tape)
    - Note: the position where the super special character is written has now been written to twice. DON'T WRITE HERE AGAIN!!!!
- Copy state
    - Go right until a blank is read. This is the end of the input. Write the input divider character.
    - Go left until the preceding input divider character is reached
    - Shift right one position
    - Read the entry under the head.
        - If this is the super special symbol, don't write anything
        - Otherwise write a blank symbol in its place
    - Go right to the end of the tape (i.e., a blank is read).
        - If the symbol you read before was the super special one, write what you wanted to write before, but put a dot over it so you know where you left off (Clarification: If the symbol already has a dot over top of it, don't put a dot on top of a dot. Just leave one dot. The dot is just so you know where you left off.)
        - Otherwise, write the entry you previously read. If the entry has a dot overtop of it, remove the dot. Otherwise you might get confused later. There should be only one character with a dot overtop of it at once.
    - Go Left until a blank or the super special symbol is encountered. Go back to the "Shift right one position" step
    - Repeat until the input divider character is read. Go to "Find where you left off"
- Find where you left off
    - Now that you have copied everything over, you need to go back to where you were before
    - You should be at the start of the input string you just copied over
    - Go Right until you encounter a symbol with a dot over it.
    - Now you are exactly where you left off before, but you have written the symbol you wanted to write
    - Return to "Do your processing"

8. Consider a Turing machine with a two-dimensional tape where the head can move up and down as well as right and left. Assume that the paper is infinite in the right and downward directions. Give a formal definition of this machine.

The formal definition of a Turing machine is a 7-tuple. The only major difference between this two-dimensional tape and the traditional one-dimensional tape is the transition function. Specifically, for the one-dimension tape, the transition function is

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

For the case of the two-dimensional Turing machine, the transition function is now

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

Where U and D stand for up and down respectively, and up is defined the same where left is when the end of the tape is reached. In other words, when up is performed along the top of the two-dimensional tape, the head stays where it is.

The other parts of the definition of a turning machine are listed below. They do not differ from the one dimension Turing machine.

1. Finite set of states Q
2. Input alphabet ∑ (does not include the blank symbol)
3. Tape alphabet Γ (includes the blank symbol and potentially other characters)
4. Start state q_0
5. Accept state q_{accept}
6. Reject state q_{reject}