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

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

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

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

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

Hint: You can use the S (stay put) movement to achieve the effect of having one head move faster than the other.

Note: One-way infinite tape means the tape extends to the right with infinitely many blanks. In a two-way infinite tape, the tape extends both ways with infinitely many blanks. In both cases the heads may move right or left or (in this problem) stay-put. We say the machine keeps the head in the same place if it tries to move left from the leftmost square.

Solution:

In brief, the idea is to find the middle by having one head move twice as fast as the other until the faster head reaches the end of the input string. At this point, the faster head will be at the end of the second half of the input string and the slower head at the end of the first half of the input string. Then equality is checked as both heads move left at equal speed.

Here is a more detailed description of the program. Note that both heads start at the left end of the tape.

1. If the first square is blank (i.e. the input is the empty string), accept.
2. Without moving the heads put a special marker on the first square. (This requires having a corresponding marked symbol in the tape alphabet for every symbol in the input alphabet. It is also possible to know where the first square is by executing a right-shift and writing a blank to the first symbol. This latter strategy does not require expanding the alphabet.)
3. While the second head is over a non-blank square
    a. Keep the first head still, move the second head right.
    b. If the second head sees a blank square, then reject (input length is odd);
    c. Otherwise, move both heads right.
4. Move both heads one square left.
5. While the first head is not on the first tape square
    a. If there are different characters under the heads, then reject;
    b. Otherwise, move both heads left
6. Accept if the first tape head is the marked version of the character under the second tape head, otherwise reject.

There are more complicated but still correct solutions that really need only one head, basically finding the middle of the string by marking outside in, and then checking each character of w one by one by marking them off.

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.

Solution:

Initially set $\delta' = \delta$.

If M accepts or rejects while staying put then we can just accept or reject while moving right. Formally if $\delta'(q,\gamma) = (q_{accept}, \xi, S)$then redefine $\delta'(q,\gamma) = (q_{accept}, \xi, R)$and if $\delta'(q,\gamma) = (q_{reject}, \xi, S)$then redefine $\delta'(q,\gamma) = (q_{reject}, \xi, R)$.

If M executes a stay-put in a non-halting state and then moves right or left then we can combine these operations. So if $\delta'(q,\gamma) = (p, \xi, S)$ and $\delta'(p,\xi) = (r, \varrho, R)$ redefine $\delta'(q,\gamma) = (r, \varrho, R)$. Similarly if $\delta'(q,\gamma) = (p, \xi, S)$ and $\delta'(p,\xi) = (r, \varrho, L)$ redefine $\delta'(q,\gamma) = (r, \varrho, L)$. Keep repeating until we can make no more changes.

Any stay-put moves that remain will just stay-put indefinitely and the machine will run forever (in non-halting states). By definition the input will not be in the language so we can get the same language by just making those transitions immediately reject while moving right. Formally if $\delta'(q,\gamma) = (p,\xi,S)$redefine $\delta'(q,\gamma) = (q_{reject},\xi,R)$.

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.

Hint: Use lots of tape and do a lot of copying.

Solution:

Given a single-tape Turing machine M, we can construct another single tape machine M-prime that writes at most twice to each square on the tape as follows. In short, the idea for M-prime to simulate each step of M by appending a '#' followed by the entire tape contents of M to the tape, with a special symbol 'H' inserted before the square under the head to keep track of the head position. (This should be reminiscent of the configuration notation employed in the videos and Michael Sipser's textbook.)

Starting from M's transition function, we construct M-prime's with these steps.

1. Add an initialization phase which appends #Hw to the end of the tape, where w is the input string.

2. Replace every transition in the transition function of M with a submachine that

- letting z be the string of symbols after the last # on the tape, appends #y to the tape, where y is the string z modified so as to account for the effect of the replaced transition of M.
- moves the head over the square immediately to the right of the ‘H’ in the string y.

Note that most of the work of writing y based on z is simply a matter of copying. Which symbols have been copied and which symbols have not is recorded on the tape by replacing symbols that have been read for copying with an 'X'.

Note that in the case of a transition that moves the head left, the copying phases must read ahead two symbols before copying so as to know where to place the head.

With this strategy, each square is written to twice: once when the blank symbol is overwritten and once when 'X' is written during the copying phase.

Bonus (hard): Describe how you would modify your machine so that it only changes the symbol once.

Solution:
Modify the above solution by using two tape cells for each tape cell of M. Use one of those cells for the copy-in and the other as the marker for the copy-from.

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.

A 2D Turing machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, where $Q, \Sigma, \Gamma$are all finite sets and

1. $Q$is the set of states
2. $\Sigma$is the input alphabet not containing the blank symbol
3. $\Gamma$is the tape alphabet containing the blank symbols and where $\Sigma \subset \Gamma$.
4. $\delta: Q \times \Gamma \quad \rightarrow Q \times \Gamma \times \{L, R, U, D\}$ is the transition function
5. $q_0 \in Q$is the start state
6. $q_a \in Q$is the accept state
7. $q_r \in Q$is the reject state, where $q_a \neq q_r$.

Note: this is the usual definition with $\delta: Q \times \Gamma \quad \rightarrow Q \times \Gamma \times \{L, R, U, D\}$.