MAT 3670: Lab 1
Introduction to Turing Machines

Background

For this lab, we will explore the basics of the *Turing machine*, a simple yet powerful computing device. The Turing machine—proposed in 1936 by the English mathematician Alan Turing—has been used as a theoretical abstraction of computing devices. The ingredients required for a Turing machine are as follows. Refer to Figure 1 for a specific example.

- An *input alphabet*, not containing the blank symbol □. In the example, the input alphabet is the binary alphabet, $\{0, 1\}$.
- A *tape alphabet* containing the blank symbol □, all of the input alphabet symbols, and possibly others. In the example, the tape alphabet is $\{\square, 0, 1, X\}$.
- A finite set of *states*. In the example, there are seven states: $q_0, q_1, \ldots, q_6$.
- A designated state to serve as the *initial state*, indicated by an incoming arrow. In the example machine, state q_3 is the initial state.
- Some number, possibly zero, of *final states*, indicated with a double circle. In our example, there is one final state, q_6.
- A collection of *transitions*. A transition from one state to another is labeled with a triplet of values which indicates what is to happen in one step of the computation. For example, the transition from q_3 to q_4 labeled `0; X, L` means the following: "if the read/write head is scanning a square on the tape containing a `0`, then it should overwrite it with an `X` and move one position to the left." Possible directions are `L` (move one square left), `R` (move one square right) or `S` (stay at the current position).

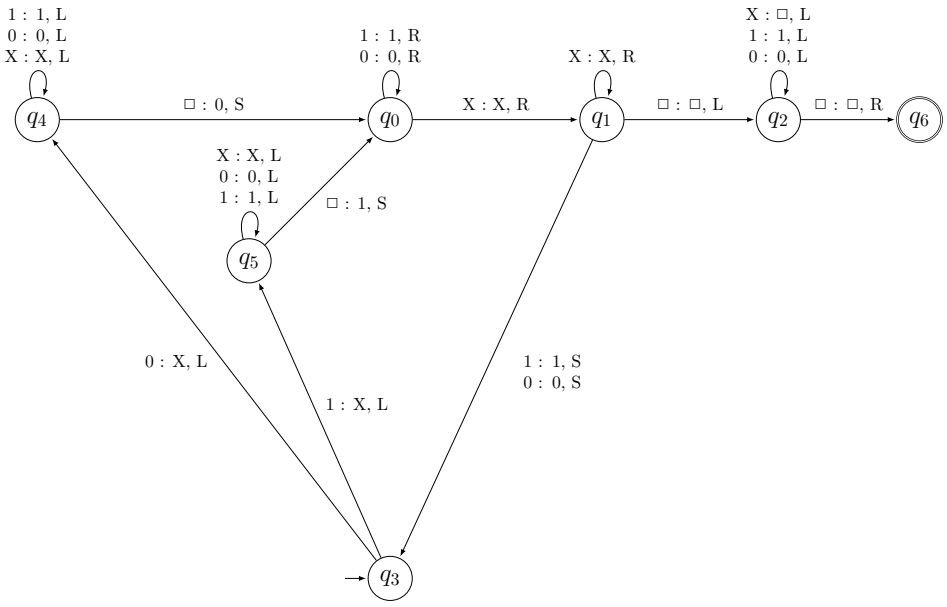

Figure 1: An example of a Turing machine. Initially, the read-write head is positioned at the leftmost bit of the input string. Upon processing, the input string is erased and replaced with an output string, leaving the read-write head at the leftmost bit of the output string. The initial state is q_3 and there is one final state, q_6.

Pre-Lab Exercises

Complete the following two exercises before arriving at this week's lab.

Exercise 1

The Turing machine shown in Figure 1 computes a well-known operation on bit strings. To get a better understanding of what it does and how it operates, we can record the computational details for a typical input string.

To begin, we write the input string `011` onto the tape of the Turing machine, position the read/write head at the first symbol, and set the current state to q_3, the initial state of the Turing machine. This is shown pictorially in Figure 2. The tape extends infinitely in both directions and all squares not shown have blanks. At each step of the computation, the machine has a *current state* and the read/write head is positioned under one of the squares of the tape.

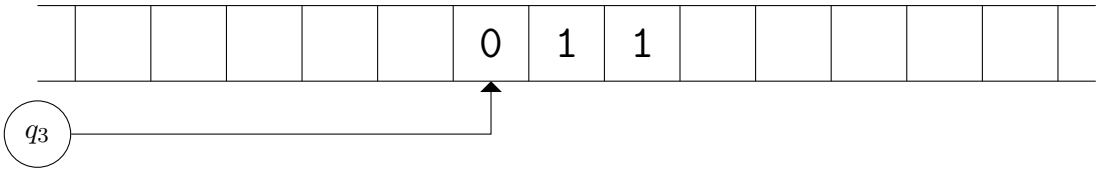

Figure 2: A snapshot of the example Turing machine, given input string `011`. The read/write head is positioned at the leftmost symbol of the input string and the current state is q_3.

From this point on, the transitions of the Turing machine dictate what happens to the current state, the read/write head, and the tape content. In this example, since the current state is q_3 and the symbol under the read/write head is 0, the transition `0; X, L` applies. This updates the current state (changing from q_3 to q_4), overwrites the `0` with `X`, and moves the read/write head one position left. This is shown in Figure 3.

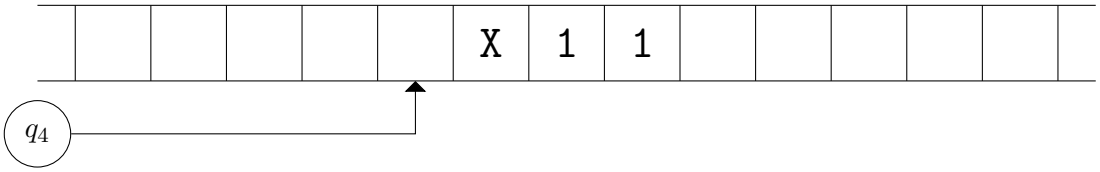

Figure 3: A snapshot of the example Turing machine after one step of the computation.

Determine the remaining sequence of actions for this input string, using the templates given in Figures 4 through 8 as a guide. Each time the Turing machine makes a transition, record the tape content, the current state, and the position of the read/write head. This will take some patience, since an input of just three bits requires more than 30 transitions of the given Turing machine. You should discover that eventually the Turing machine will reach state q_6, at which point it halts since there are no transitions which apply in that state.

Exercise 2

Design a Turing machine which, given a positive integer n, determines the value $\lfloor n/2 \rfloor$. The *unary* system can be used to represent the value of n: for the input, we write n consecutive 0's on the input tape (surrounded by blanks) and position the read/write head at the leftmost 0. When your Turing machine halts, the result should appear on the tape—also in unary—with the read/write head positioned at the leftmost 0 of the result.

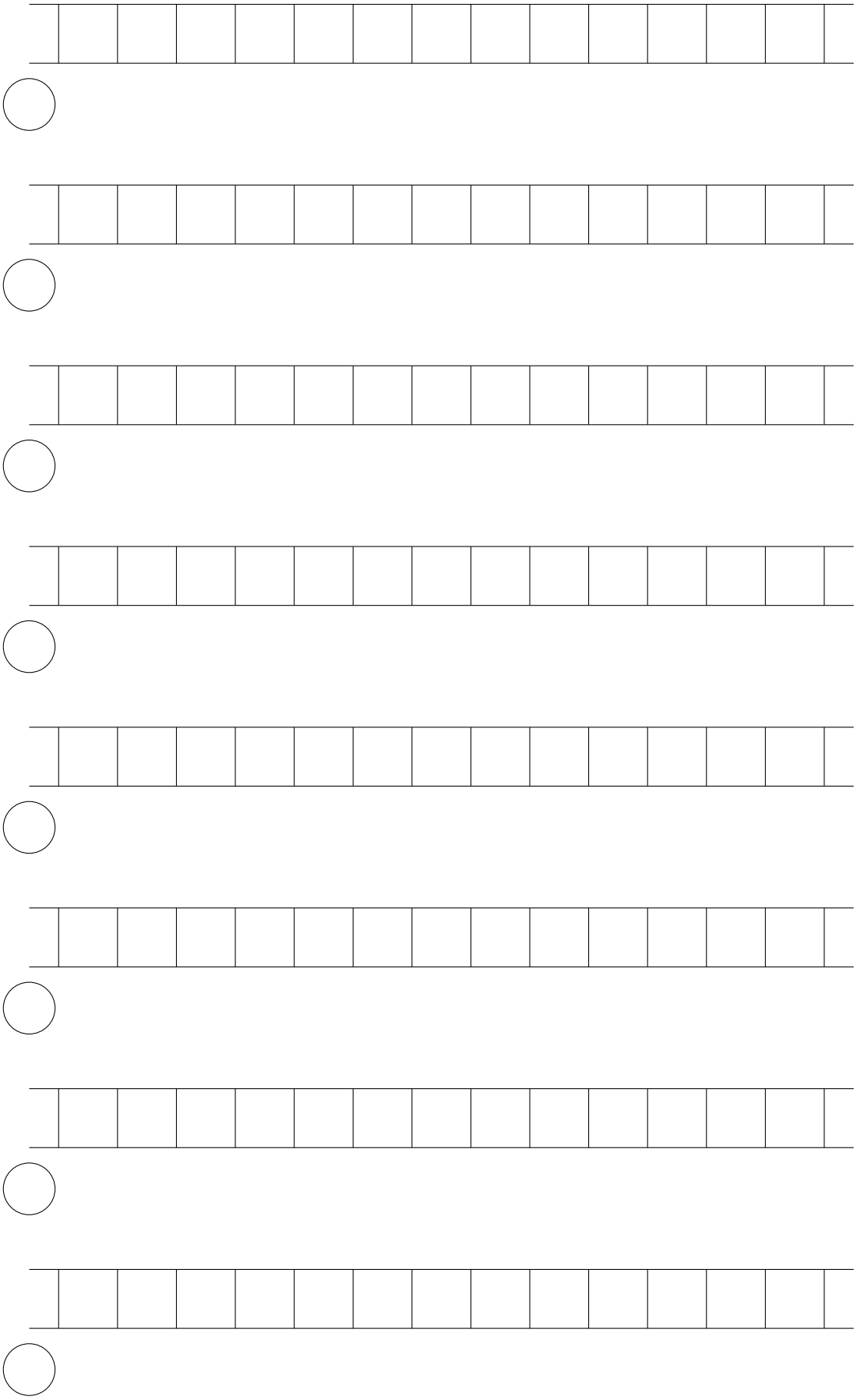

Figure 4: Additional snapshots of a Turing machine.

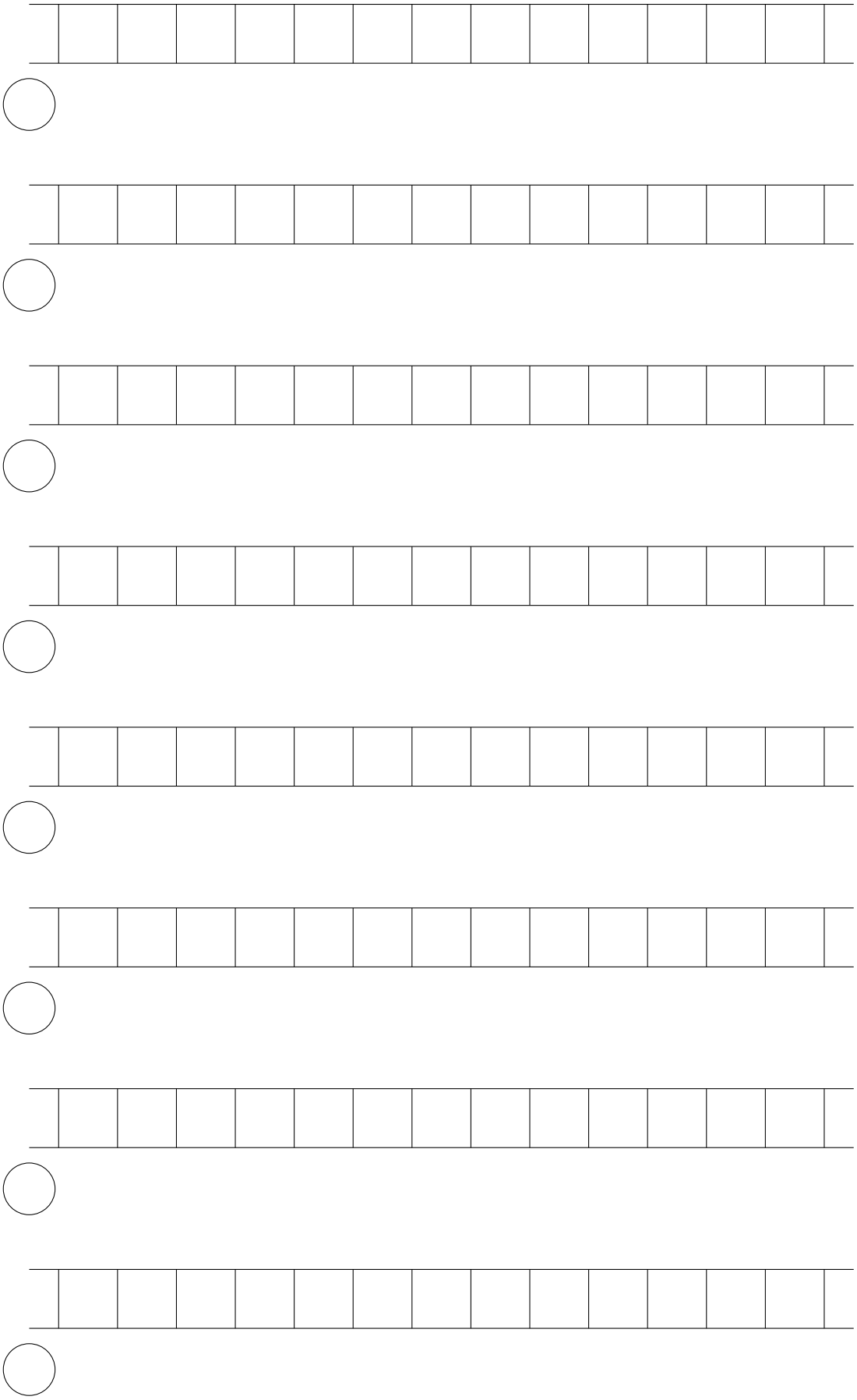

Figure 5: Additional snapshots of a Turing machine.

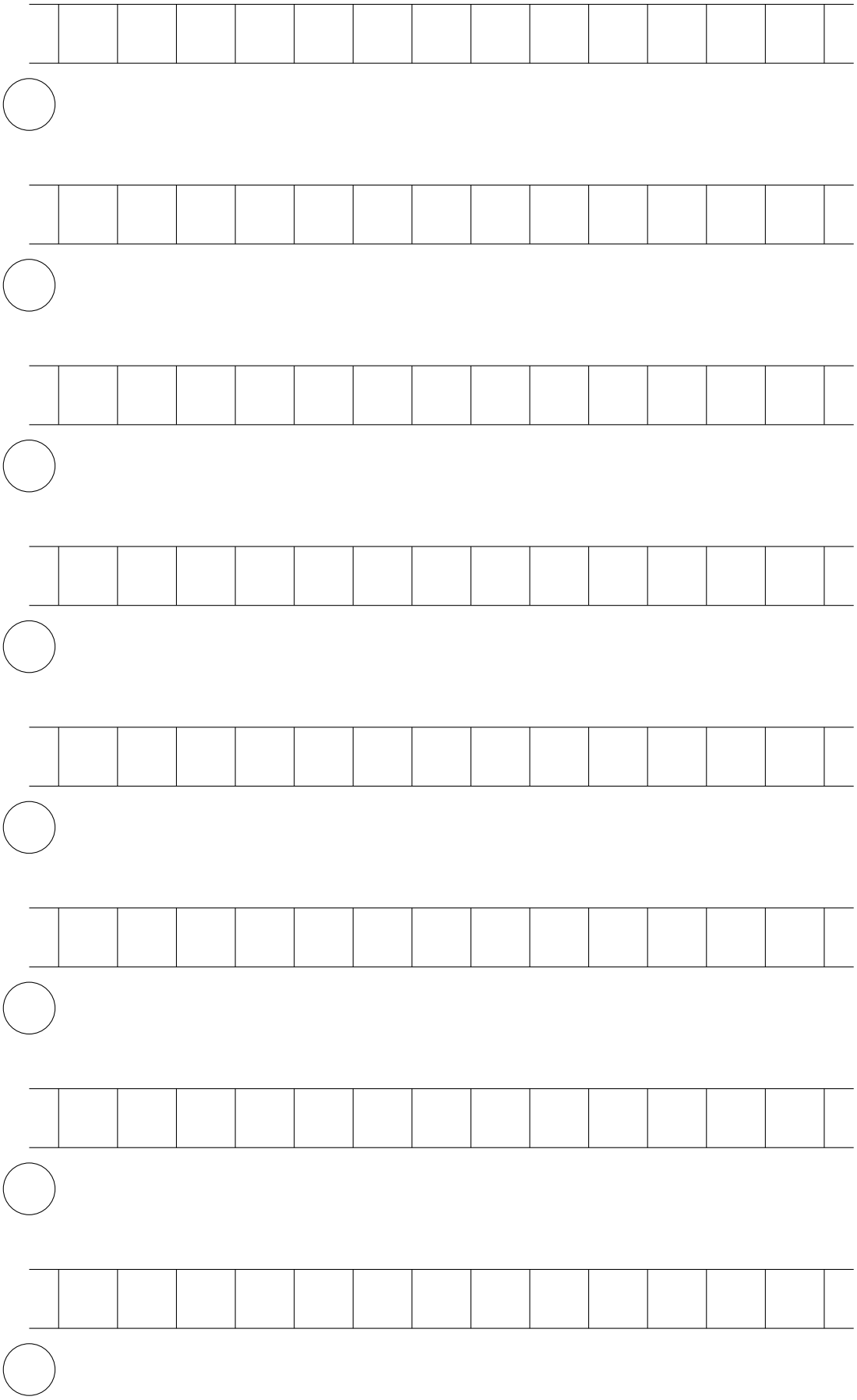

Figure 6: Additional snapshots of a Turing machine.

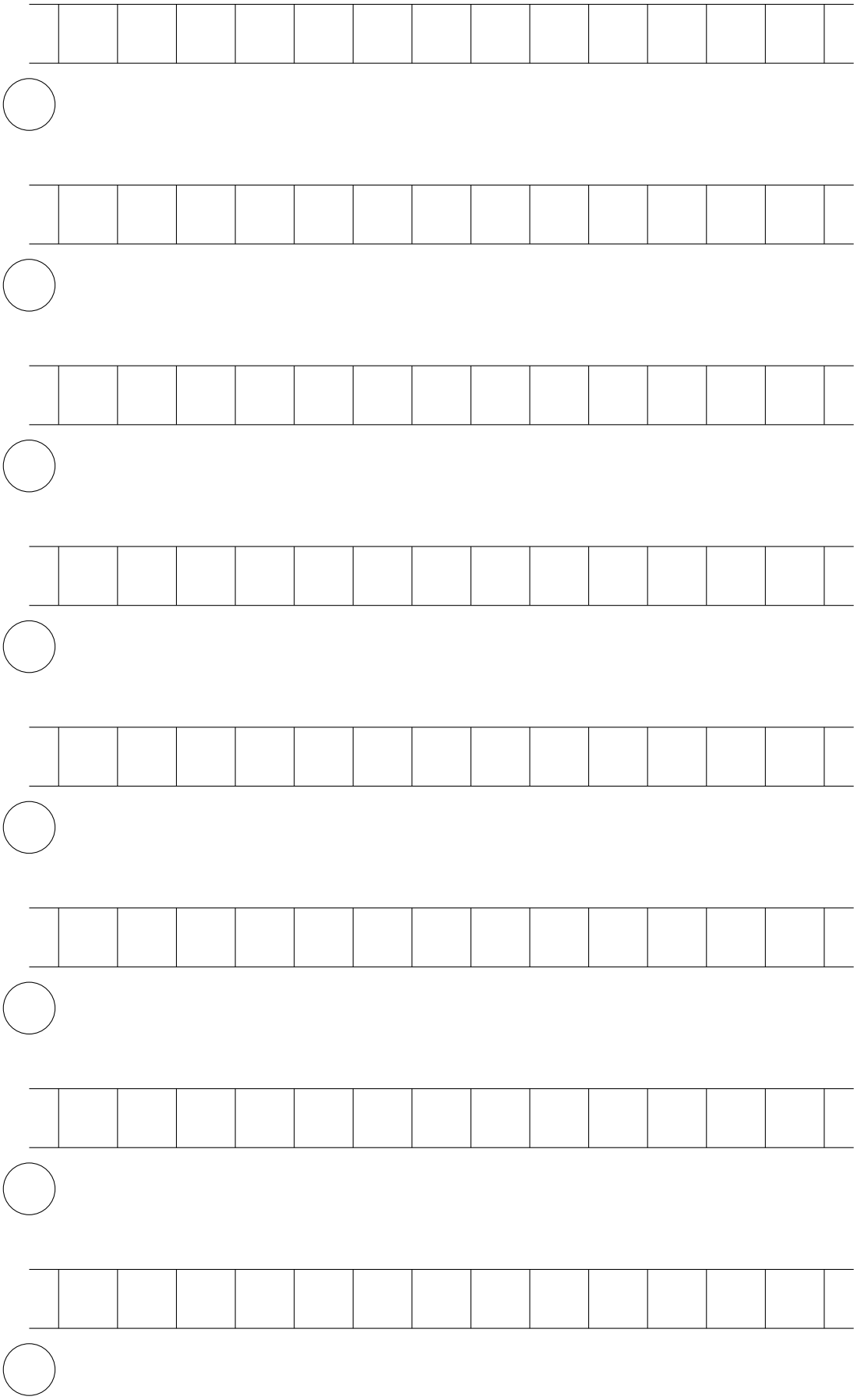

Figure 7: Additional snapshots of a Turing machine.

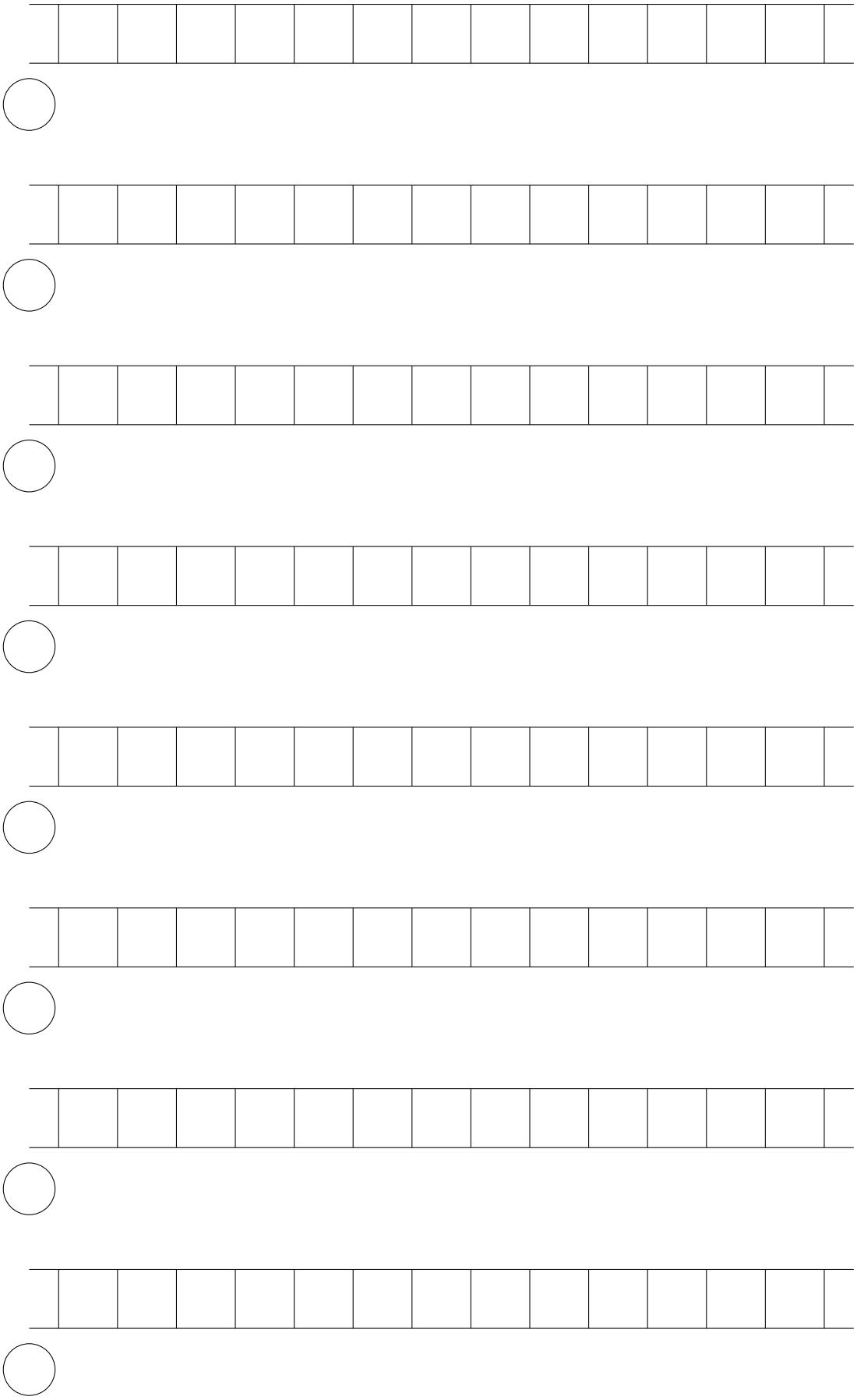

Figure 8: Additional snapshots of a Turing machine.

Laboratory Exercises

1. Using your EIU account, login at one of the Macintosh computers. Create a directory named `3670` within your home directory to store all MAT 3670-related materials. Within this `3670` directory, create a `lab1` directory and put any files you create for this lab in that directory.

2. We are going to use JFLAP[1] to simulate the computational aspects of Turing machines. An on-line tutorial on JFLAP can be found at:

   `http://www.cs.duke.edu/csed/jflap/tutorial/`

   Peruse this tutorial to get a sense of how to interact with JFLAP, especially the part dealing with Turing machines. A live demonstration of JFLAP will be given in class and/or lab to orient you to its features.

3. Start JFLAP and enter the example Turing machine of Figure 1. When you have everything the way you want it, adjust the position of the states so that the transition diagram is pleasing to the eye.

4. Save your Turing machine in `lab1a.jff`.

5. Choose `Input | Fast Run` and use `011` for the input. Scroll through the resulting window and see how your prediction from the pre-lab exercise matches reality.

6. Choose `Input | Multiple Run (Transducer)`, which will give you a chance to simulate the Turing machine on many different input strings. Unlike `Fast Run`, you won't see all the intermediate computation steps. Enter the 16 possible bit strings using 4 bits, as shown in Figure 9. Click on the `Run Inputs` button and record the results.

7. Given an arbitrary bit string, what does this Turing machine produce? Using the multiple run feature of JFLAP, choose some additional input strings and run the Turing machine to see if your prediction holds up for these strings.

8. After spending some time thinking about how the states and transitions of this Turing machine are related, give a thorough explanation of how this machine goes about the task of converting an input string to its respective output string. Using the `Aquamacs` editor, place your explanation in a file named `README`.

9. Using your design from the pre-lab exercises, create a Turing machine which computes $\lfloor n/2 \rfloor$, saving it in `lab1b.jff`. Using `Input | Multiple Run (Transducer)`, test your machine for $1 \leq n \leq 16$ and verify the correct results are produced. Debug as needed. Explain your design by adding comments to your `README` file.

Submissions

Submit your work by dragging your `lab1` folder onto the EIU submission application. (More details will be given when we meet in the lab.)

[1] JFLAP is free software. If you want a copy for your own computer, visit `http://www.cs.duke.edu/csed/jflap/`.

Input string	Output string
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Figure 9: Test data for the sample Turing machine.