Group Conference 1 – Turing Machines

1. Go to the class web page under Links, and click on Turing Machine simulator from IronPhoenix. Spend a few minutes reading through the instructions. When you're ready, try out the three Turing machines shown below on the sample inputs. Type the rules for a machine into the Programming window, exactly as shown, and the desired input string into the box labeled "Initial characters on tape", and then press the *Install Program* button. This will initialize the machine. To run it on the input string, press the *Start* button.

Inverter

```
1 0  1 1 >
1 1  1 0 >
1 _  H _
```

Example inputs

```
1100101
00000
111000
```

Eraser

```
1 0  1 _ >
1 1  1 _ >
1 _  H _
```

Example inputs

```
01010101
111
```
(blank tape)

Looper

```
1 0  1 0 >
1 1  1 1 <
1 _  H _ >
```

Example inputs

```
00000
0001111
111
```

2. Now see if you can figure out what each of the "mystery" Turing machines TM1 through TM6 does. A data file containing these machines is available under Labs/lab01 on the class web page. Cut-and-paste the rules for TM1 into the simulator, then try the following inputs and record the output produced in each case. Do you see a pattern? How would you describe in words what this Turing machine does?

TM1 Input	**Output**	**TM1 Description:**
1. `011001#`	1.	
2. `111#`	2.	
3. `00#`	3.	
4. `0101#`	4.	
5. `1101#`	5.	

Next, try TM2 and record the output produced. Create more tests of your own to fill out the table, using inputs of the same form as shown in the table. Describe TM2's function to the right of its table.

TM2 Input	**Output**	**TM2 Description:**
1. `111`	1.	
2. `1`	2.	
3. `11111`	3.	
4.	4.	
5.	5.	

Now do the same thing for TM3 through TM6. Try to discover each Turing machine's intended purpose by experimenting with test inputs. Record at least five of your tests per machine in the tables below. Describe each Turing machine's function, if you can. IMPORTANT: The input format for TM3 and TM4 is two blocks of `1`s containing no `0`s, separated by a single `#`.

TM3 Input	**Output**	**TM3 Description:**
1. `11#111`	1.	
2. `1111#1`	2.	
3. `1#1`	3.	
4.	4.	
5.	5.	

TM4 Input	Output	TM4 Description:
1. `11111#11`	1.	
2. `111#11`	2.	
3. `111#111`	3.	
4.	4.	
5.	5.	

IMPORTANT: Input for TM5 should be a block of `1`s containing no `0`s.

TM5 Input	Output	TM5 Description:
1. `11111`	1.	
2. `1111`	2.	
3. `111`	3.	
4.	4.	
5.	5.	

IMPORTANT: Input for TM6 should be a string of `0`s and `1`s.

TM6 Input	Output	TM6 Description:
1. `10011001`	1.	
2. `11011001`	2.	
3. `1111`	3.	
4.	4.	
5.	5.	

3. Try out the more complicated machines. The Multiplier TM takes two numbers in unary notation, separated by a single #, and computes their product. For example, try the input `111#1111`. The Divides TM takes two numbers and determines whether the first number divides evenly (with no remainder left over) into the second number. The Primality Tester TM takes a number and determines whether it is prime.

4. Modify the Inverter machine so that the read/write head returns to the beginning of the input before halting.

5. Create a Turing machine that will move to the right until it finds a $. Then it will erase everything on the tape between that $ and the next $ to the right. It will halt when it gets to the second $. The $'s themselves should not be erased. You can do this with a fairly simple machine that uses only two states. (Note that if the machine is started on a tape that does not contain two $'s to the right of the machine's starting position, then the machine will never halt.)

6. Create a machine that will move to the right until it finds three X's in a row. It should halt when it finds a group of three consecutive X's. For an added touch, see if you can make it move back to the first of the three X's and halt there.

7. Construct a Turing machine to do the following: Assume that the machine is started on a tape that contains nothing but a string of $'s. The machine is started on the left end of this string. The purpose of the machine is to multiply the length of the string by 3. For example, if it is started on a string of seven $'s, it should halt with twenty-one $'s on the tape. If it is started on a string that contains just one $, it should halt with three $'s on the tape. Here is one way that the machine might operate: Change one of the $'s to an x, then go to the end of the string and write two more x's. Go back and process the next $ in the same way. Continue until all the $'s have been processed. Then change all the x's to $'s.