CSC 404 - Activity/Project 7 - Name: Chris Glanzer

Problem 1 (Copy and Paste!). In this problem, we look at the problem of copying a string to another location - Wee!

a. Describe how a Turing machine, *COPY* would copy the string 1011. (You can simply give the general idea here and leave the technical movements for the state diagram). That is, we begin with a tape

⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯

and need to produce a tape with two copies of 1011 i.e.,

⋯	⊔	1	0	1	1	#	1	0	1	1	⊔	⋯

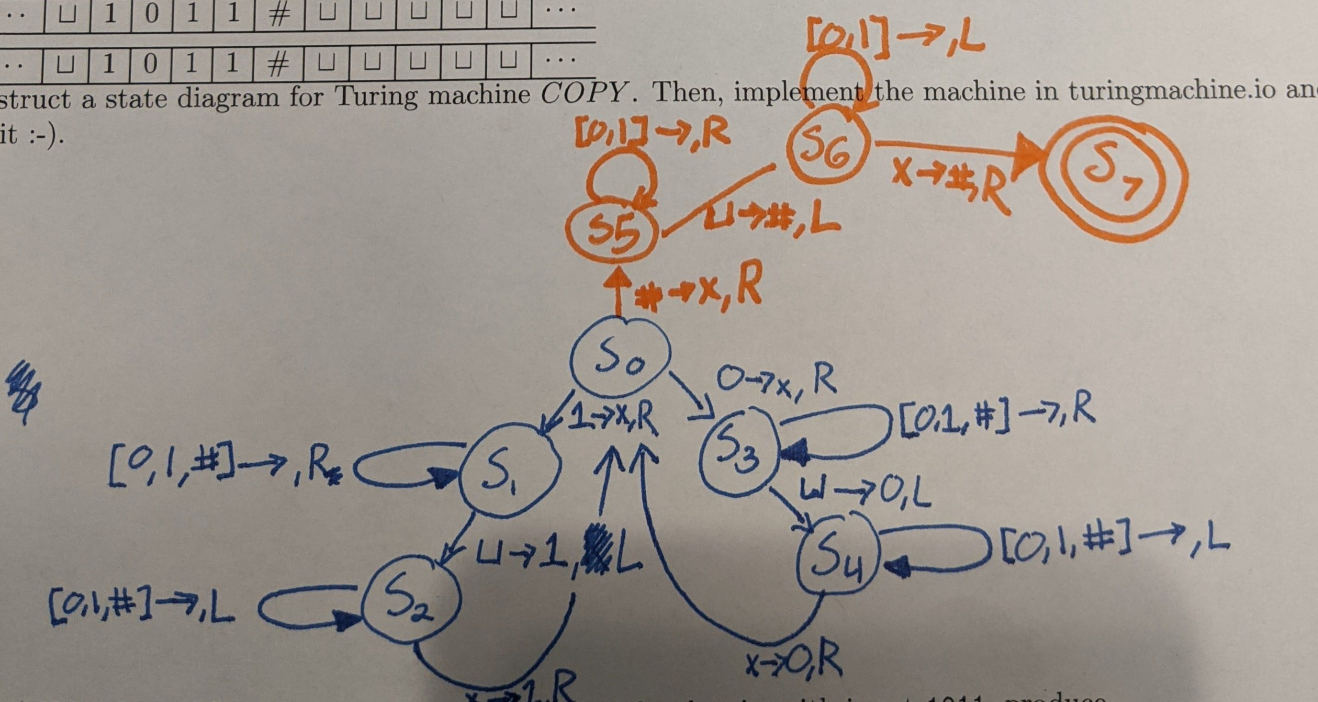
(If you want you can paste another # at the end of the string - see Bonus below)
Hint: We want to keep the original list on the tape. So, you will need to keep track of where in the initial list you left off. You can 'X' out the copied symbol while keeping track of what symbol you have (e.g., after have 1 you go back and paste the 1 into the X) or you can leave an 'X' for 0 and 'Y' for 1. Either way works - or explore other ideas :-).

first thought was use other symbols to indicate read [1,0] like [3,4] but that seems kinda iffy

⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯
⋯	⊔	1	0	1	1	#	⊔	⊔	⊔	⊔	⊔	⋯

- branch [1,0]
- write x at place being read
- write [1,0] at next blank
- return to x
- write [1,0] over x
- shift right
- repeat until read "#"

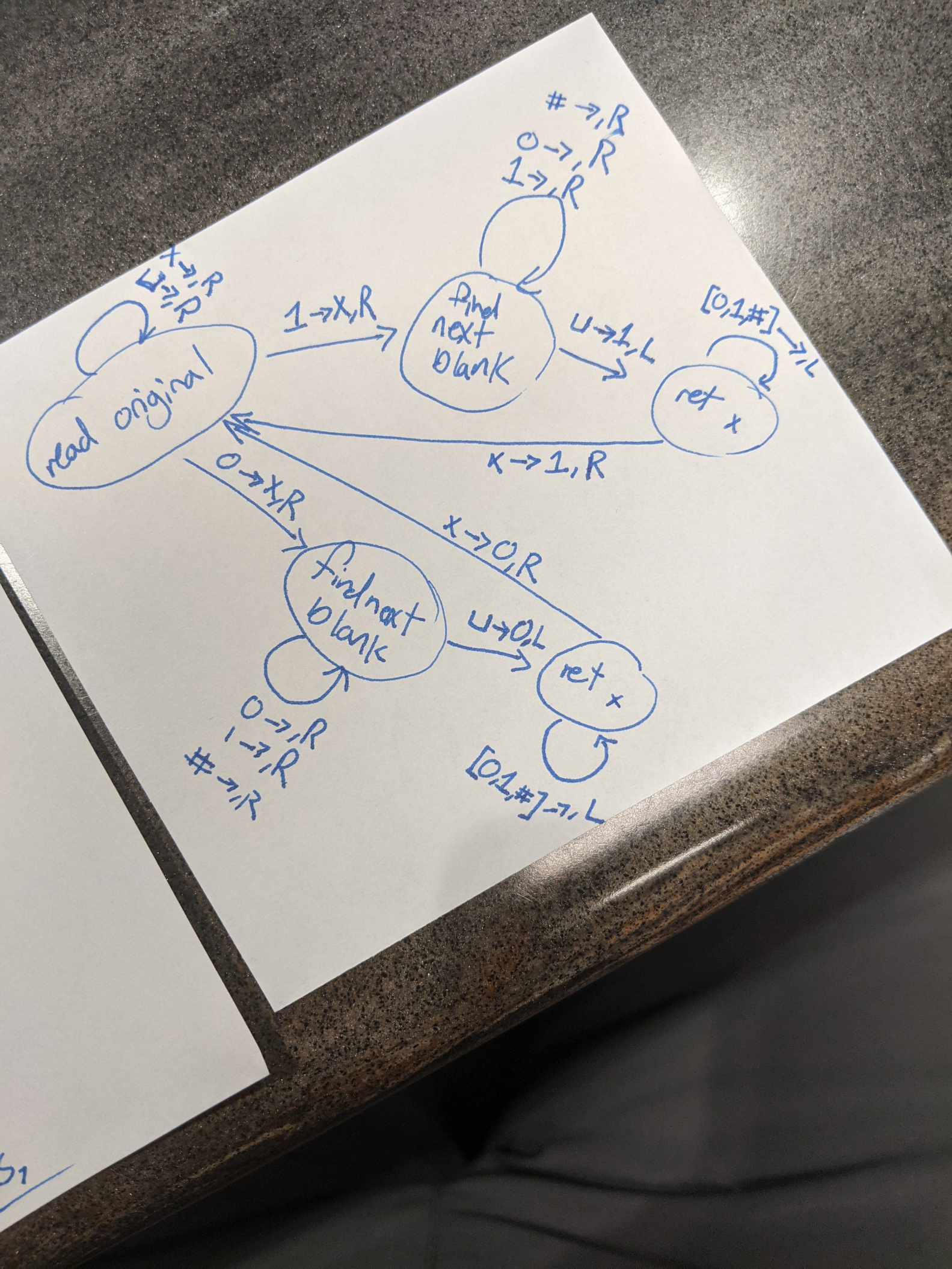
b. Construct a state diagram for Turing machine *COPY*. Then, implement the machine in turingmachine.io and test it :-).



c. (Bonus) Edit the machine to copy the string indefinitely, that is, with input 1011, produce

1011#1011#1011#1011#1011#1011#...

↓ on the orange branch, return to S0 instead of S7


read original
find next blank
next x
find next blank
next x
1 → X, R
U → 1, L
[0,1,#] → →, L
X → 1, R
0 → X, R
X → 0, R
U → 0, L
0 → →, R
1 → →, R
→ →, R
[0,1,#] → →, L

Problem 2 ($f(n_1, n_2) = \min(n_1, n_2)$). In this exercise, we consider the problem of finding the minimum of two nonnegative integers. That is, we work towards constructing a Turing machine that computes the function $f(n_1, n_2) = \min(n_1, n_2)$. As with addition we will work with the unary representation of integers. That is, we represent integer n by a string of n 1s (e.g., we represent 5 as 11111 and 3 as 111.) To represent the tuple (n_1, n_2) we use a string of n_1 1s, followed by an $*$, followed by a string of n_2 1s. For example, we represent the tuple $(5, 3)$ as $11111 * 111$.

a. Describe how a Turing machine, MIN, would compute the function $f(5, 3) = \min(5, 3) = 3$. That is, we begin with a tape

⋯	⊔	1	1	1	1	1	*	1	1	1	⊔	⋯

and need to produce a tape with 3 1s, i.e.,

⋯	⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	1	1	1	⊔	⋯

(You can simply give the general idea here and leave the technical movements for the state diagram.)

start from the *, moving outward flipping one bit at a time on each side. the first to hit a blank is the min, so wipe the other. then wipe the *. then flip the winning bits back to 1.

b. Describe how a Turing machine, MIN, would compute the function $f(3, 5) = \min(3, 5) = 3$. That is, we begin with a tape

⋯	⊔	1	1	1	*	1	1	1	1	1	⊔	⋯

and need to produce a tape with 3 1s, i.e.,

⋯	⊔	1	1	1	⊔	⊔	⊔	⊔	⊔	⊔	⊔	⋯

(You can simply give the general idea here and leave the technical movements for the state diagram.)

same as above

c. Construct a state diagram for Turing machine, MIN. Then, implement the machine in turingmachine.io and test it against $(5, 3)$ and $(3, 5)$.