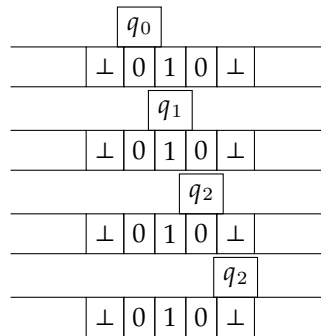
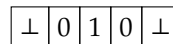


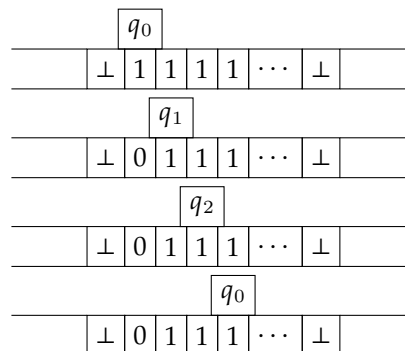
iii. 010

Solution:

And again, we will switch states to q_3 and halt, leaving us with the final tape:



(b) Describe the general behavior of M when the input is of the form 1^k for some $k \in \mathbb{N}$.

Solution:

Now, we can see that we are in the same starting state, only now we are working with the string 1^{k-3} . So we repeat the above process on this substring, and continue until we find the character \perp , at which point we will be stuck in q_3 , and halt. So we can say that the machine takes a string 1^k and converts every third 1 to a zero, starting with the first 1.

(c) Construct a Turing machine $M' = (Q', \Sigma, T, \delta', q'_0, q'_{\text{accept}})$, where $T = \{0, 1, \perp\}$, that satisfies each of the following:

- Replaces the first occurrence of the substring 01 in the input with 10, and leaves the rest unchanged.
- Accepts if and only if the input contains the substring 010.

Specify only the state transitions relevant to this task (you may assume the rest lead to a rejecting state or halt).

Solution: Let $M = (Q, \Sigma, T, \delta, q_0, q_{\text{accept}})$ be a Turing machine, where:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_{\text{accept}}, q_{\text{reject}}\}$,

$\Sigma = \{0, 1\}$ is the input alphabet

$T = \{0, 1, \perp\}$ is the tape alphabet (with \perp denoting the blank symbol).

The transition function δ is defined by the following table

δ	0	1	\perp
q_0	$(q_1, 0, R)$	$(q_0, 1, R)$	(q_7, \perp, L)
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	(q_7, \perp, L)
q_2	$(q_3, 0, L)$	$(q_0, 1, R)$	(q_7, \perp, L)
q_3	$(q_3, 0, L)$	$(q_0, 1, L)$	(q_4, \perp, R)
q_4	$(q_5, 0, R)$	$(q_4, 1, R)$	(q_A, \perp, R)
q_5	$(q_5, 0, R)$	$(q_6, 0, L)$	(q_A, \perp, R)
q_6	$(q_A, 1, R)$	—	—
q_7	$(q_7, 0, L)$	$(q_7, 1, L)$	—
q_8	$(q_9, 0, R)$	$(q_8, 1, R)$	—
q_9	$(q_9, 0, R)$	$(q_{10}, 0, L)$	—
q_{10}	$(q_A, 1, R)$	—	—

Summary: q_0 to q_2 will search for the 010 in the input tape.

If 010 is found, q_3 will loop back to the start, and q_4 to q_6 will scan for 01 and replace it with 10 and accept.

If 010 is not found, we are at the right end of the tape, and we go to q_7 . q_7 scans back to the start of the tape, and moves to q_8 through q_{10} which perform identically to q_4 through q_6 , however these will reject after completion (or if a blank space is found while scanning).

2. Let $\Sigma = \{0, 1\}$. Define the language:

$$L' = \{0^n 1^n 0^n 1^n \mid n \in \mathbb{N}_0\}.$$

(a) Design a Turing machine that accepts the language L' .

Solution: Let $M = (Q, \Sigma, T, \delta, q_0, q_{\text{accept}})$ be a Turing machine, where:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_{\text{accept}}, q_{\text{reject}}\}$,

$\Sigma = \{0, 1\}$ is the input alphabet

$T = \{0, 1, X, Y, \perp\}$ is the tape alphabet (with \perp denoting the blank symbol).

The transition function δ is defined by the following (it is understood that "search right for \perp " means to move right and leave characters unchanged until the specified character is found):

q_0 : Search right for a 0. When one is found, mark it X and go to q_1 . If \perp is found, go to q_5 .

q_1 : Search for 1. When one is found, mark it Y and go to q_2 . Reject if \perp is found.

q_2 : Search right for a 0. When one is found, mark it X and go to q_3 . If \perp is found, reject.

q_3 : Search for 1. When one is found, mark it Y and go to q_4 . Reject if \perp is found.

q_4 : Search left for \perp . Then move right and go to q_0 .

q_5 : The final check. Search left. If a 0 or 1 is found, reject. If a \perp is found, accept.

- (b) Prove that if the Turing machine accepts a string x , then $x \in L'$.

Could not finish this :(

Proof: Suppose that M accepts x . Then after q_0 through q_4 have finished processing, there will be no 0,1 on the tape (In order for M to accept a string, it must have reached q_5 at the right end of the tape and q_5 must find no 0,1 in the tape). ■

- (c) Modify the Turing machine so that it replaces the input $x \in L'$ with the string xx (i.e., it duplicates the input).

Solution: Leave the machine M unchanged, however add states, and change q_5 :

q_5 : Sweep left until \perp , then switch to q_6 . If a 0 or 1 is found, reject.

q_6 : Switch an X to a 0 and switch to q_7 . Or, switch a Y to a 1 and switch to q_8 . If \perp is found, accept.

Search right for \perp , then change it to 0 and switch to q_9 .

Search right for \perp , then change it to 1 and switch to q_9 .

Search left for a 0 or 1. Leave it unchanged, then move right and switch to q_6

- (d) Prove that the modified machine correctly duplicates the input only if $x \in L'$.

Could not finish this :(

3. Consider an infinite collection of mirrors $\{M_i\}_{i \in \mathbb{N}}$ in an art gallery. Each mirror M_i displays a unique digital artwork titled Art_i , generated by a python function P_i . The artwork may or may not visually include the string "Art_i" within its display. Define the language:

$$L = \{i \in \mathbb{N} \mid \text{Python function } P_i \text{ does not include the string } \text{Art}_i \text{ in its output}\}.$$

- (a) Is the language L Python recognizable? Justify your answer.

Could not finish this :(

Solution: This language is not recognizable.

- (b) Is L_{Halting} Python reducible to L ? Justify your answer.

Could not finish this :(

4. Let us define four new languages over Python functions. Determine whether they are Python-decidable.

- (a) Let $L_{\text{SyntaxCheck}}$ contains all strings that define a syntactically correct Python function.

$$L_{\text{SyntaxCheck}} = \{\text{code} \mid \text{code is a syntactically valid Python function definition}\}.$$

Solution: Since the Python interpreter can detect syntax errors, it recognizes $L_{\text{SyntaxCheck}}$, and the language is Python decidable. (In fact, there is a Python interpreter Byterun, written only in Python).

- (b) Let $L_{\text{EventuallyTrue}}$ be the set of all Python functions func such that for at least one argument arg , $\text{func}(\text{arg})$ returns true. Hence,

$$L_{\text{EventuallyTrue}} = \{\text{func} \mid \exists \text{arg}, \text{func}(\text{arg}) = \text{True}\}.$$

Solution: Could not finish this :(

- (c) Let $L_{\text{FalseOnSelf}}$ be the set of all Python functions `func` such that `func(func)` returns `False`. Hence,

$$L_{\text{FalseOnSelf}} = \{\text{func} \mid \text{func}(\text{func}) = \text{False}\}.$$

Solution: Let D be a decider, for the sake of contradiction. Consider $D(D)$. If $D(D)$ is true, then $D \in L_{\text{FalseOnSelf}}$, and therefore $D(D)$ must be false. A contradiction!

On the other hand, if $D(D)$ is false, then $D \notin L_{\text{FalseOnSelf}}$, and therefore $D(D)$ must be true. A contradiction! Therefore no such decider can exist and $L_{\text{FalseOnSelf}}$ cannot be decidable.

- (d) Let $L_{\text{HaltOnEmpty}}$ contains all Python functions that halt when run with no arguments.

$$L_{\text{HaltOnEmpty}} = \{\text{func} \mid \text{func}() \text{halts (terminates execution)}\}.$$

Solution: The halting problem reduces to $L_{\text{HaltOnEmpty}}$, and is as such undecidable.