A Turing Machine (TM) is a form of finite state machine that differs from ones we have seen so far in some respects.

(i) A TM can rewrite the symbol on the cell it reads.

(ii) A TM can move its head both to the right and to the left.

(iii) A TM does not have final states but halting states. Whenever a TM enters a halting state the computation ends and whatever is written on the tape is the output of the TM. For instance, if the task to be performed by the TM is to recognize a language, then it should indicate acceptance and rejection by writing its decision on the tape, rather than signalling the decision by the state it halted, as was the case with finite automata.

(iv) Every blank cell in the input tape carries the special blank symbol (we will use '#') and the tape is assumed to be unbounded in both directions.

Now, the formal definition:

Definition 0.1 (Turing Machine)
A Turing Machine is a 7-tuple $\langle Q, \Sigma, s, H, D, B, \delta \rangle$ where,

Q is a finite set of states;

Σ is the input alphabet;

$s \in Q$ is the start state;

$H \subseteq Q$ is the set of halting states;

D is a two element set of directions for left and right, we require $D \cap \Sigma = \emptyset$;

B is the singleton set of blank symbol, we require $B \cap \Sigma = \emptyset$.

$\delta : (Q - H) \times \Sigma \cup B \mapsto Q \times \Sigma \cup B \times D$ is the transition function.

□

We will designate the transition function δ as a set of 4-tuples (p, x, q, y), where p and q are states, x is a symbol from $\Sigma \cup B$, and y is either a symbol from $\Sigma \cup B$ or a direction instruction from D. Therefore, (p, x, q, y) means: if you read x in state p, change to state q and rewrite x as y (if y is a symbol) or move the head to another cell indicated by y (if y is a direction). In other words, there are two types of transitions, one rewrites the symbol on the current cell, the other leaves the content of the cell unchanged and moves the head to left or right. The transition function does not have any transitions for the halting states; whenever the machine enters a halting state the computation ends.

A **situation** of a TM is a 4-tuple (q, x, u, w), where q is the current state, x is the symbol on the current cell, u is the string to the left of the current cell up to the start of the input, and w is the string to the right of the current cell up to the end of the input. We can talk of the start and end of the input, since the input comes sandwiched between non-finite series of blank symbols. For example, an input like *aabbaa* is represented as $\ldots$###*aabba*###$\ldots$, where '$\ldots$' represent non-finite sequences of blank symbols.

Example 0.2
Here is a TM $\langle \{q_0, q_1, q_2\}, \{a, b\}, q_0, \{q_2\}, \{\leftarrow, \rightarrow\}, \{\#\}, \delta \rangle$, where

$$\begin{aligned} \delta = & (q_0, a, q_1, b) \\ & (q_0, b, q_1, a) \\ & (q_1, a, q_0, \rightarrow) \\ & (q_1, b, q_0, \rightarrow) \\ & (q_0, \#, q_2, \#) \end{aligned}$$

The following is the sequence of situations that the machine runs through given the input *abab* on its tape. TM's start at the initial cell of the input and in the start state; ε designates the empty string; you can think of it as

an infinite sequence consisting of only the blank character '#'. (Be careful not to confuse the representation of a transition with that of a situation)

$$
\begin{array}{c}
(q_0, a, \varepsilon, bab) \\
(q_1, b, \varepsilon, bab) \\
(q_0, b, b, ab) \\
(q_1, a, b, ab) \\
(q_0, a, ba, b) \\
(q_1, b, ba, b) \\
(q_0, b, bab, \varepsilon) \\
(q_1, a, bab, \varepsilon) \\
(q_0, \#, baba, \varepsilon) \\
(q_2, \#, baba, \varepsilon)
\end{array}
$$

□

Question 1

(a) Describe the task the TM in the previous example is designed to perform.

(b) Give the sequence of situations for the input *baab*.

Question 2

Given the TM $\langle \{q_0, \ldots, q_6, s_1, \ldots, s_6, r_0\}, \{a, b\}, q_0, \{r_0\}, \{\leftarrow, \rightarrow\}, \{\#\}, \delta \rangle$, where δ is as follows:

$(q_0, a, q_1, \#)$	$(q_1, \#, q_2, \rightarrow)$	$(s_1, \#, s_2, \rightarrow)$
$(q_0, b, s_1, \#)$	$(q_2, a, q_2, \rightarrow)$	$(s_2, a, s_2, \rightarrow)$
$(q_0, \#, r_0, \#)$	$(q_2, b, q_2, \rightarrow)$	$(s_2, b, s_2, \rightarrow)$
	$(q_2, \#, q_3, \rightarrow)$	$(s_2, \#, s_3, \rightarrow)$
	$(q_3, a, q_3, \rightarrow)$	$(s_3, a, s_3, \rightarrow)$
	$(q_3, b, q_3, \rightarrow)$	$(s_3, b, s_3, \rightarrow)$
	$(q_3, \#, q_4, a)$	$(s_3, \#, s_4, b)$
	$(q_4, a, q_4, \leftarrow)$	$(s_4, a, s_4, \leftarrow)$
	$(q_4, b, q_4, \leftarrow)$	$(s_4, b, s_4, \leftarrow)$
	$(q_4, \#, q_5, \leftarrow)$	$(s_4, \#, s_5, \leftarrow)$
	$(q_5, a, q_5, \leftarrow)$	$(s_5, a, s_5, \leftarrow)$
	$(q_5, b, q_5, \leftarrow)$	$(s_5, b, s_5, \leftarrow)$
	$(q_5, \#, q_6, a)$	$(s_5, \#, s_6, b)$
	$(q_6, a, q_0, \rightarrow)$	$(s_6, b, q_0, \rightarrow)$

What is the task this machine is designed to perform? (Try it on some inputs, but you do not need to submit your trials.)

Question 3

Design a TM that takes an input from $\{a, b\}^*$ and removes all the b's from the input without leaving blanks after the deletion. For instance *abba* becomes *aa*; *aaa* stays *aaa*.

Please submit your answer to me in a txt file carrying your name.