Engineering and Applied Science Programs for Professionals
Whiting School of Engineering
Johns Hopkins University
685.621 Foundations of Algorithms
Programming Assignment 1
Assigned with Module 1
Due at the end of Module 5

Total Points 100/100

One of the following two problems is to be implemented by the individual student (i.e., it is not collaborative). You may use any programming language you wish. Please follow the requirements provided under Module 1 under the link Programming Assignment Guideline.

Programming (non-collaborative)

Problem 1 - Data Analysis

The Iris Plants Database contains 3 classes of 50 instances each, where each class refers to a type of Iris plant. Five attributes/features were collected for each plant instance. It can be downloaded from iris.arff on the Sample Weka Data Sets webpage (https://storm.cis.fordham.edu/ gweiss/data-mining/datasets.html).

1. Implement an algorithm to read in the Iris dataset.
2. Implement an algorithm to visually see two sets of features and the class they belong to.
3. Implement your sorting algorithm from Homework 1 Problem 5.

Problem 2 - Deterministic Turing Machine

In this problem you will implement two variants of a DTM. The definition and example are provided below followed by two implementations. A deterministic Turing machine consists of the following:

- A finite state control,
- A tape consisting of a two-way sequence of tape squares, and
- A read-write head.

A given DTM is manipulated through the execution of a program utilizing the following components:

- A finite set Γ of symbols,
- A finite set Q of states (q_0 is designated as the start state and q_Y and q_N are designated as halting states), where $\{q_0, q_H\} \in Q$,
- The direction (left, right, hault) in which to move the tape head $s \in Q$
- A transition function $\delta : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \{+1, 0, -1\}$,
- $\Sigma \subset \Gamma$ as a set of input symbols, and
- $b \in \Gamma - \Sigma$ corresponds to the "blank symbol" (#)

This allows a DTM to be represented as a finite set of program lines with the form of a 6-tuple TM $M = (\Gamma, Q, s, \delta, \Sigma, b)$.

In other words, the DTM will scan the tape, and the transition function will read the symbol from Γ currently written on the table and determine what character to write in its place as well as which direction s to move the read-write head. Presumably, if the program indicates $+1$, the head moves to the right, if the program indicates -1, the head moves to the left and if the program indicates 0, the head stays still.

To see how a DTM works, suppose we have $\Sigma \subset \Gamma$ as a set of input symbols and $b \in \Gamma - \Sigma$ a "blank symbol" (#). Next suppose we have an input string $x \in \Sigma^*$ where we place the symbols of x in consecutive cells of the tape in positions $1...|x|$. All other cells on the tape are assumed to have b. The program will begin in state q_0 with the read-write head scanning square 1 and proceed according to the transition function.

If the current state of the DTM is ever q_Y or q_N, then the program terminates. On the other hand, if the current state is in $Q - \{q_Y, q_N\}$, then the program continues. When transitioning, the read-write head will first erase the symbol in the square it is examining and then write the new symbol as specified by the transition function. The read-write head then either moves one position to the right or one position to the left, and the Finite State Control updates the state to some successor q'.

Because we have limited the set of halting (or terminal) states to q_Y, q_N, we note that a DTM is only able to solve problems that return a Boolean result. In particular, we say that a DMT is used to solve decision problems where q_Y indicates the DMT has decided *yes* and q_N indicates the DTM has decided *no*.

Table 1: The set of states is defined to be $Q = q_0, q_1, q_2, q_3, q_Y, q_N$, and the transition function δ is defined by the following table

$Q - \{q_Y, q_N\}$	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

To reiterate how the DTM works, Table 1 represents the states $q = Q - \{q_Y, q_N\}$, symbols in x, and the direction the read-write head moves (s). Starting with the initial state represented with q_0 the finite controller reads the initial symbol represented by x. At each step the controller reads the symbol x on the tape at state q, enters the state $q' = Q = q_0, q_1, q_2, q_3, q_Y, q_N$, writes the symbol x' in the current cell, erasing x, and moves in the direction identified by s. This is represented as the five-tuple (q, x, q', x', s). For Table 1 the DTM is represented by the following finite-state machine in Figure 1.

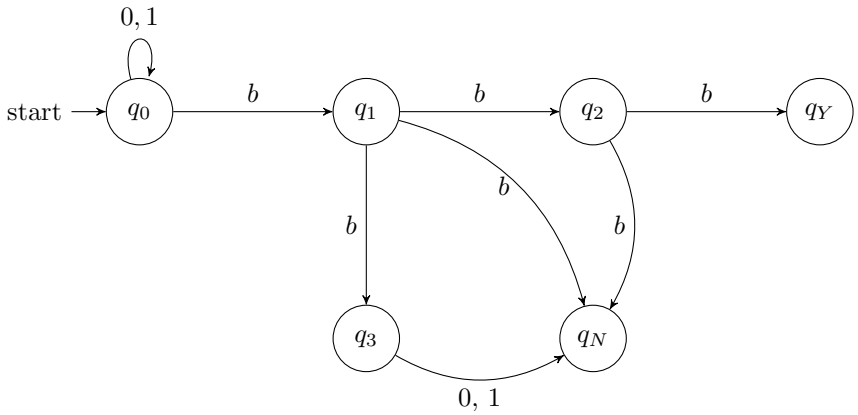

Figure 1: DTM State Diagram

Implement the DTM example provided in the course content under Module 3 (and above). Ensure that your components as listed above are clearly identified in your variable list. The finite-state machine with the individual states, state table and five-tuple TM are shown in the above example. You should implement the following methods:

1. States method, this method should have all of the operations of your TM.
2. Program line that executes the operations for each identified state, this should follow the n-tuple TM as described above for M.
3. Print method that outputs the change in tape ($x \leq 30$) after each transition function is executed.
4. Write method, for tape larger than $x > 30$ write the outputs to a file