CS375 Homework Assignment 9 Solution Set (40 points)

Due date: 4/15/2024

1. (6 points)

A Turing Machine (TM) that accepts the language $\{ a^n b^n c^n \mid n \in \mathbf{N} \}$ can be built with 6 states and 17 instructions. The instruction set of the TM is shown below.

(0, Λ, Λ, S, halt) accept Λ
(0, a, X, R, 1) mark a with X
(0, Y, Y, R, 4) no more a's

Scan right looking for b to pair with a:
(1, a, a, R, 1) b not found yet
(1, Y, Y, R, 1) b not found yet
(1, b, Y, R , 2) mark b with Y

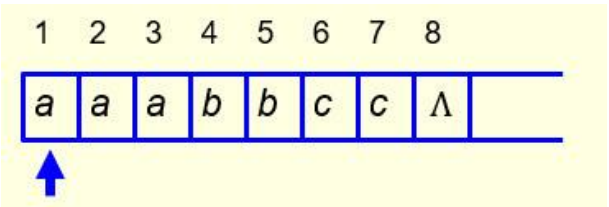
Scan right looking for c to pair with a:
(2, b, b, R, 2) c not found yet
(2, Z, Z, R, 2) c not found yet
(2, c, Z, L , 3) mark c with Z

Scan back left looking for X:
(3, Z, Z, L, 3) X not found yet
(3, b, b, L, 3) X not found yet
(3, Y, Y, L, 3) X not found yet
(3, a, a, L, 3) X not found yet
(3, X, X, R, 0) X found, turn around

Scan right looking for Λ and halt:
(4, Y, Y, R, 4)
(4, Z, Z, R, 4)
(4, Λ, Λ, S, halt)

The TM works as follows: for each 'a' in the given string, it marks that 'a' with an 'X' and then moves right to find a 'b' in the string to pair with this 'a'. Once a 'b' is found, the TM marks that 'b' with an 'Y' and then moves right again to find a 'c' in the string to pair with this 'a' as well. Once a 'c' is found, the TM marks that 'c' with a 'Z' and then turns around to repeat the same process until all the a's in the string are marked.

(a) The TM does not accept the following input string:

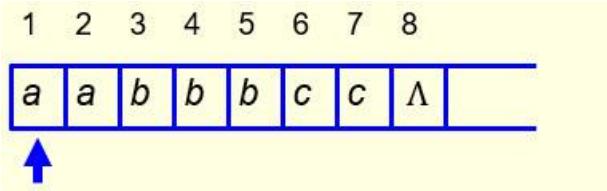


When the TM stops execution of this string, what is the location of the read/write head of the TM and what is the state of the TM?

Location = 6 (index of the entry where the TM stops execution, such as 5 or 7)

State = 1

(b) The TM does not accept the following input string:

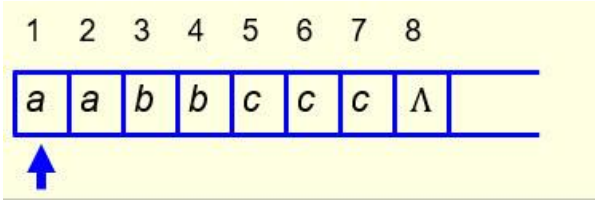


When the TM stops execution of this string, what is the location of the read/write head of the TM and what is the state of the TM?

Location = 5

State = 4

(c) The TM does not accept the following input string:



When the TM stops execution of this string, what is the location of the read/write head of the TM and what is the state of the TM?

Location = 7

State = 4

2. (10 points)

Given a non-empty string, we can move the string two units to the left using three

different approaches. The first approach is to use the TM introduced in slides 44-51 of the notes “Turing Machines and Equivalent Models-I” twice; the second approach is to move each letter of the string two units to the left directly; the third approach is to use a stack to assist the moving process.

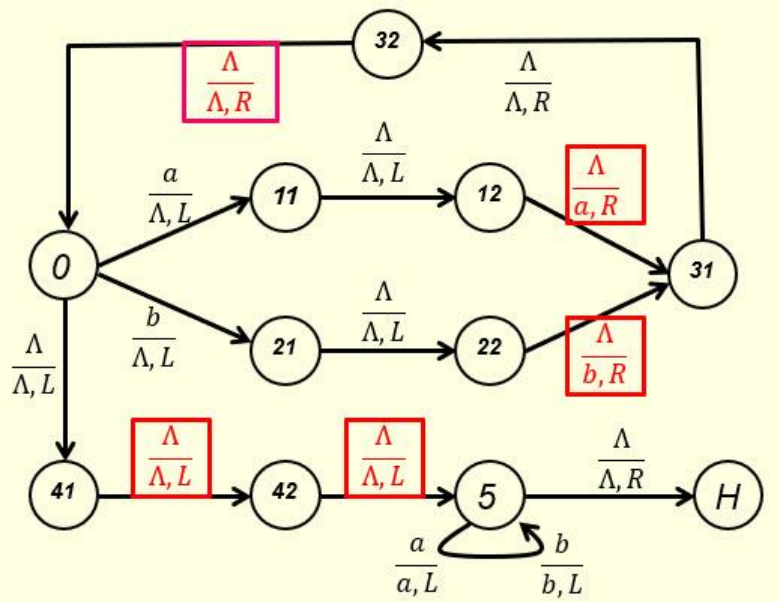
A TM that can move the string two units to the left directly requires 14 instructions and 11 states. In the following, eight instructions for such a TM are given in the tables. Fill out the remaining blanks to make the resulting instruction set the instruction set for such a TM and then fill out the blanks in the next chart to make it a complete state transition diagram of this TM. In the diagram, state H means the halt state. (6 points)

Find a or b to move:	
(0, a, Λ, L, 11)	found a
(0, b, Λ, L, 21)	found b
(0, Λ, Λ, L, 41)	no more a’s or b’s

Move to left end of output:	
(41, Λ, Λ, L, 42)	skip Λ
(42, Λ, Λ, L, 5)	skip Λ
(5, a, a, L, 5)	skip a
(5, b, b, L, 5)	skip b
(5, Λ, Λ, R, halt)	Done

Write a or b:	
(11, Λ, Λ, L, 12)	skip Λ
(12, Λ, a, R, 31)	write a
(21, Λ, Λ, L, 22)	skip Λ
(22, Λ, b, R, 31)	write b
(31, Λ, Λ, R, 32)	skip Λ
(32, Λ, Λ, R, 0)	skip Λ

Note that there are several ways to define the remaining instructions of the TM, but make sure the instructions you choose fit into the following diagram naturally and logically. (6 points)



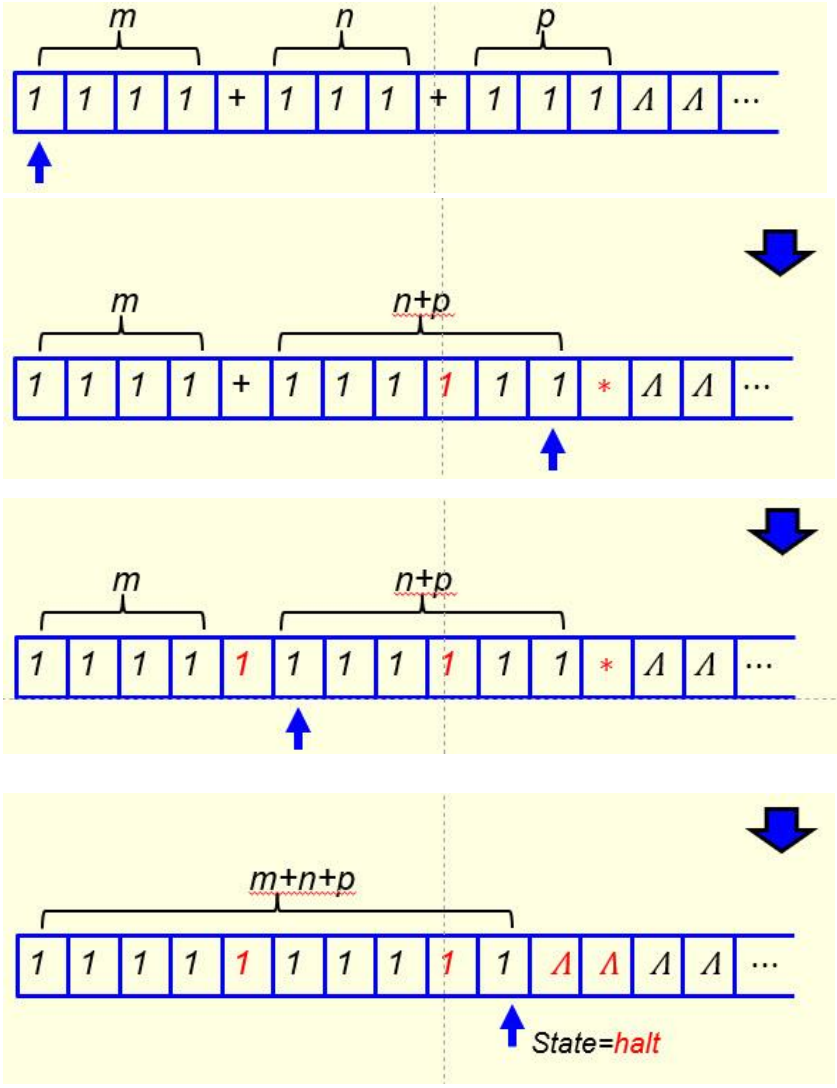
3. (2 points)

The 'P versus NP' problem is a major unsolved problem in computer science. It is an important problem because if we can prove that P = NP (i.e., all problems can be solved in polynomial time) then there will be no need to build quantum computers. Why?

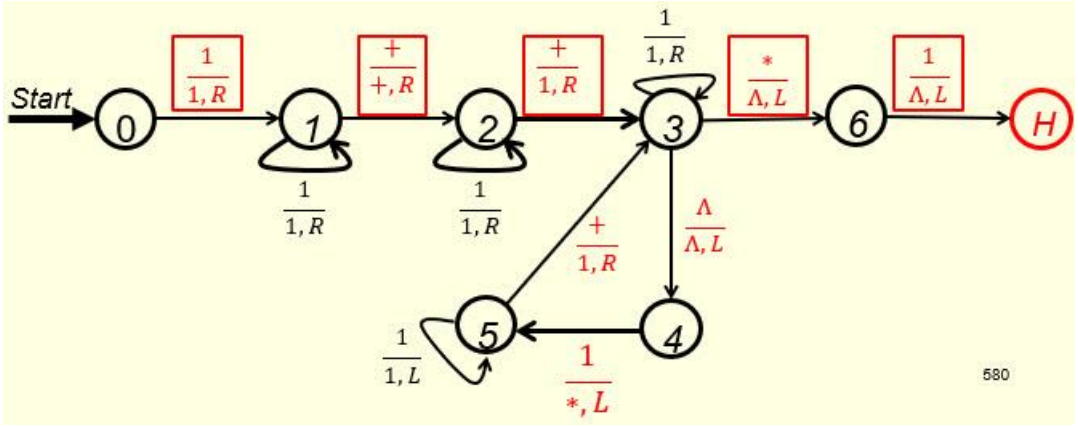
because standard computers would be fast enough (for all questions), and easier to build.

4. (10 points)

To build a TM to perform addition on three non-zero positive numbers *m*, *n*, and *p* in unary form (see the first figure below for the case *m=4*, *n=3* and *p=3*), a better approach is to perform the addition (*n+p*) first, and then perform *m+(n+p)*. To perform the addition *(n+p)* first, on its way moving right, the TM will ignore the first '+' sign (not change it to '1'), only change the second '+' to '1', then look for a 'Λ'. When a 'Λ' is reached, the TM keeps that 'Λ', turns left and changes the '1' in the next cell to '∗' (instead of '1') and then turns left (see the second figure below). To perform *m+(n+p)*, the TM then moves left to find the first '+', changes it to '1' and turns right (see the third figure below). It then moves right to find '∗'. Once '∗' is reached, the TM converts '∗' to 'Λ', turns left, changes the '1' in the next cell to 'Λ', moves one unit to the left and stop.

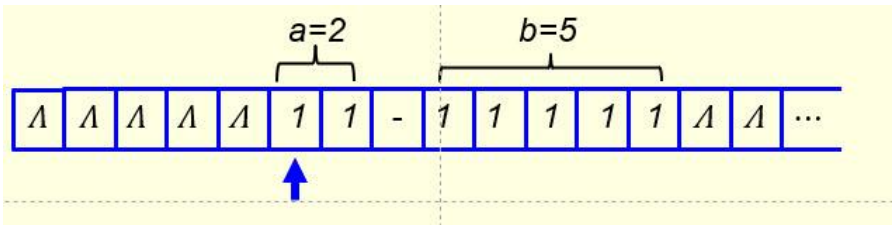


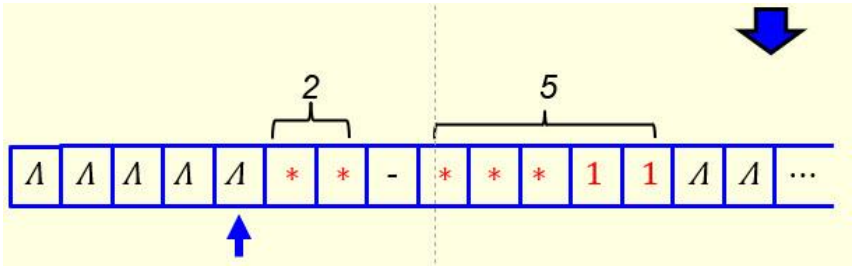
Your task here is to fill out the five blanks in the following figure to make it a TM that can perform addition on three given non-zero positive numbers in unary form directly.

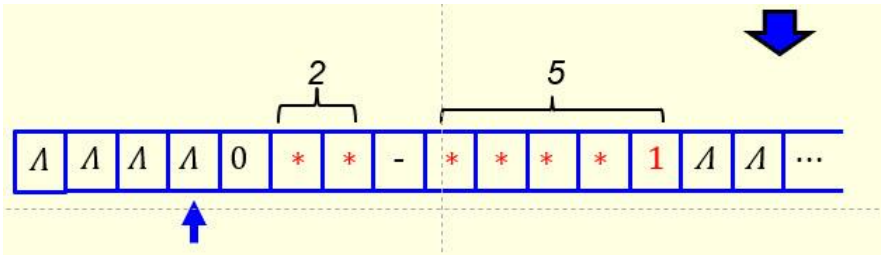


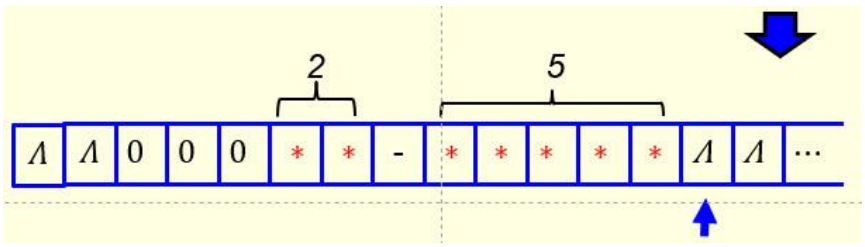
5. (12 points)

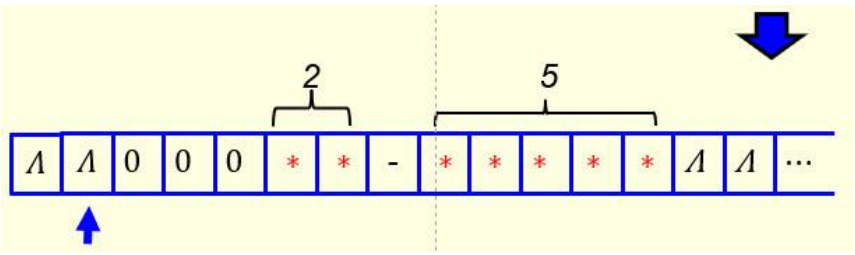
The TM that can perform the subtraction function $f(a-b)=c$ on two unary numbers *a* and *b* when *a* is bigger than or equal to *b* in notes "Turing Machines and Equivalent Models-II" can be modified to cover the case when *b* is bigger than *a* as well. Consider the following input string with a=2 and b=5. After two 1's have been converted to '$*$' in both *a* and *b*, when the third 1 in *b* is converted to '$*$', we don't have a 1 in *a* to convert, instead, we find a 'Λ' (see the second figure below). We change that 'Λ' to '0' and turn right to find another 1 in *b* to convert to '$*$'. Again, there is no 1 in *a* to match this '$*$, but a '0'. We skip this '0' to reach a 'Λ' on the left-hand side (see the third figure below). We convert this 'Λ' to 0 and turn right to find another 1 in *b* to convert to '$*$'. We repeat the same process again, get one more 0 on the *a* side (we have three 0's now) and turn right to find another 1 in *b* to convert. We don't find any 1, but a 'Λ' (see the fourth figure below). That is, no more 1's in *b* to convert. So we turn left to find the left end of the 0 string to stop. This is done by moving left to find a 'Λ' (see figure five below) and then turn right, move one cell to the right and stop (see figure six below).

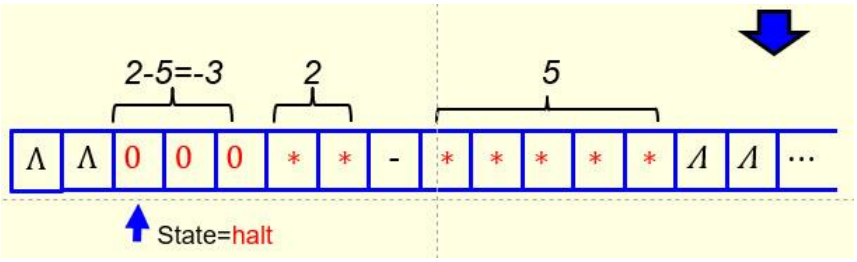












Your task here is to fill out the six blanks in the following figure to make it a TM that can subtract a bigger number from a smaller number as well.