Theory of Computation Assignment

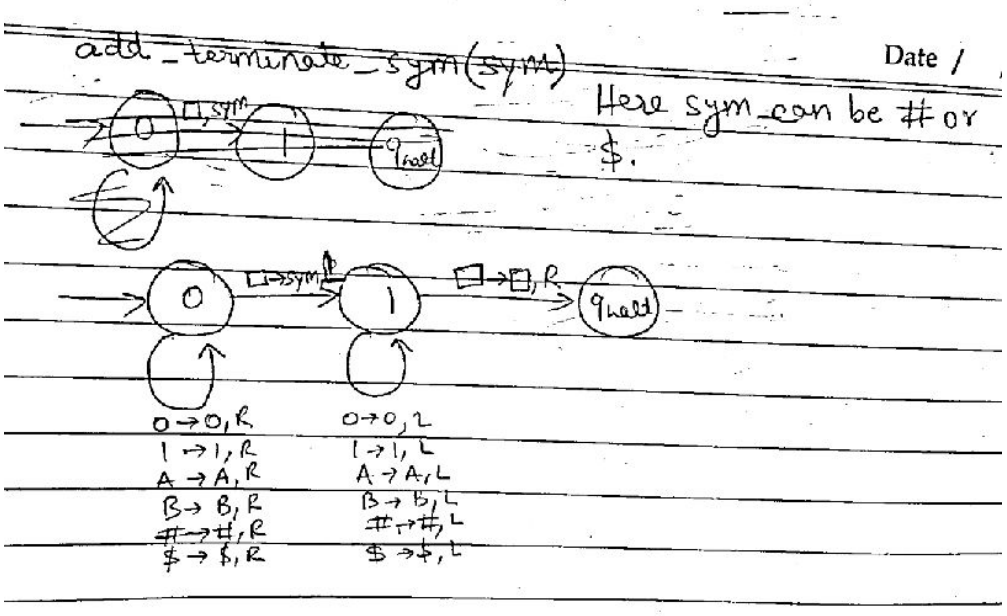
Name: Deepak Pant
Entry No: 2017csb1072

Initial Code: I have used structures to implement Turing Machine.
Structure TM contains tape and tape_head. For states, state structure is present which has its state_id, number of transitions and list of transitions. The transition is performed as it is usually defined. In each subroutine I have used NULL as the halting state for that subroutine. Initially the input is copied to the tape after some spaces are padded before it. Then the turing machine proceeds. At the end the tape is printed beginning from tape_head till tape_head encounters a blank symbol.

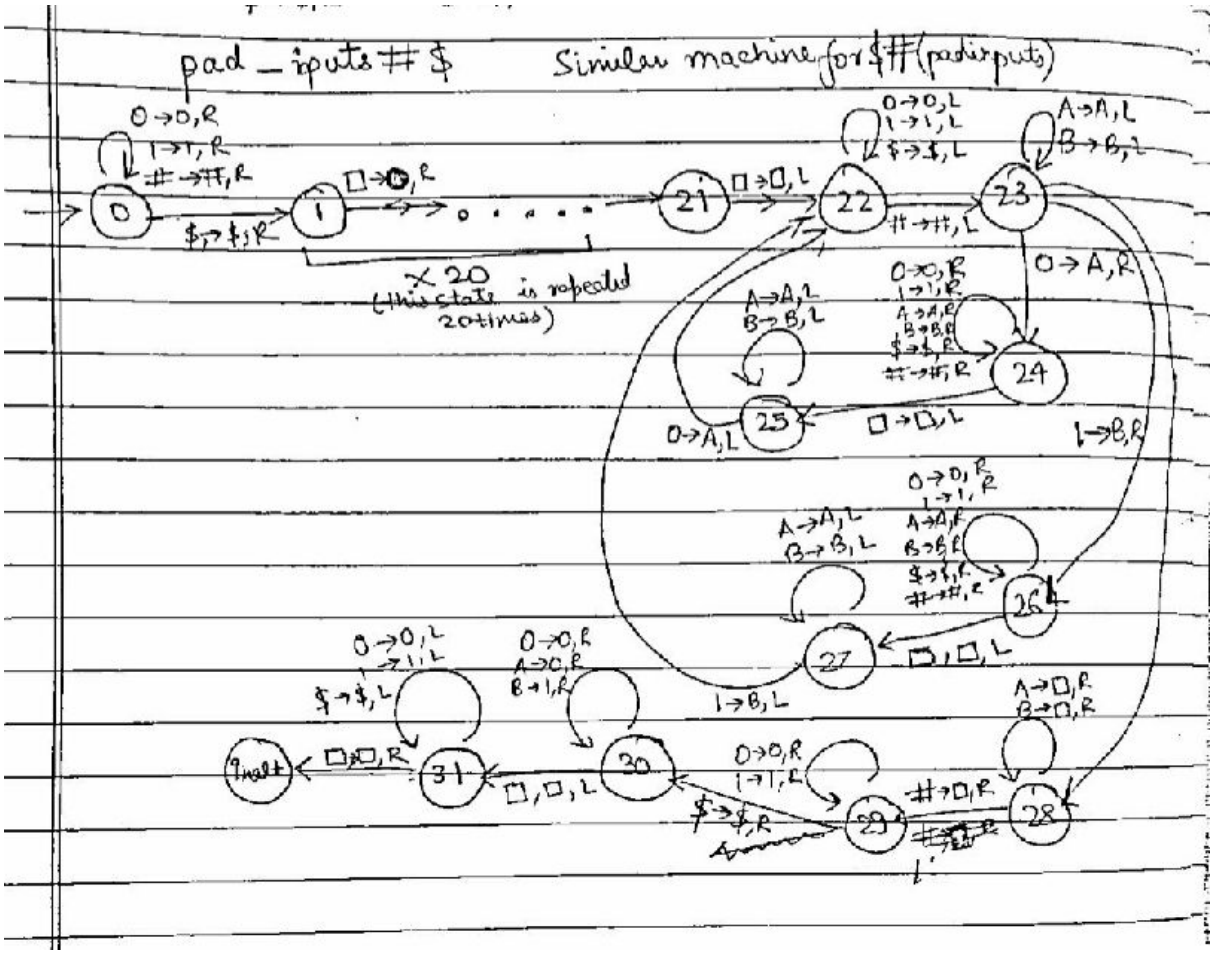
Note: I have used symbol q_{halt} in subroutines to show their end. This doesn't mean the whole turing machine has ended. It is just pointing to the start state of the next subroutine.

Q1: Find remainder of a when divided by b.
Explanation: I have used 5 subroutines.
They are performed in following order:
add_terminate_sym$->pad_inputs#$->add_terminate_sym#->pad_inputs$#->add_terminate_sym$->find_remainder. The first four are only for padding the input with zeros.

1. add_terminate_sym$: It appends $ to the end of the string.
2. add_terminate_sym#: It append # to the end of the string.



3. pad_inputs#$: It copies the first number to the end of the non-blank part of the tape. Its main function is to pad the inputs with zeros so both are of 20 length.
4. pad_inputs$#:It copies the first number to the end of the non-blank part of the tape. Its main function is to pad the inputs with zeros so both are of 20 length.
   Here the following diagram is for subroutine#3. Similar diagram is for #4 by swapping '# and '$'.



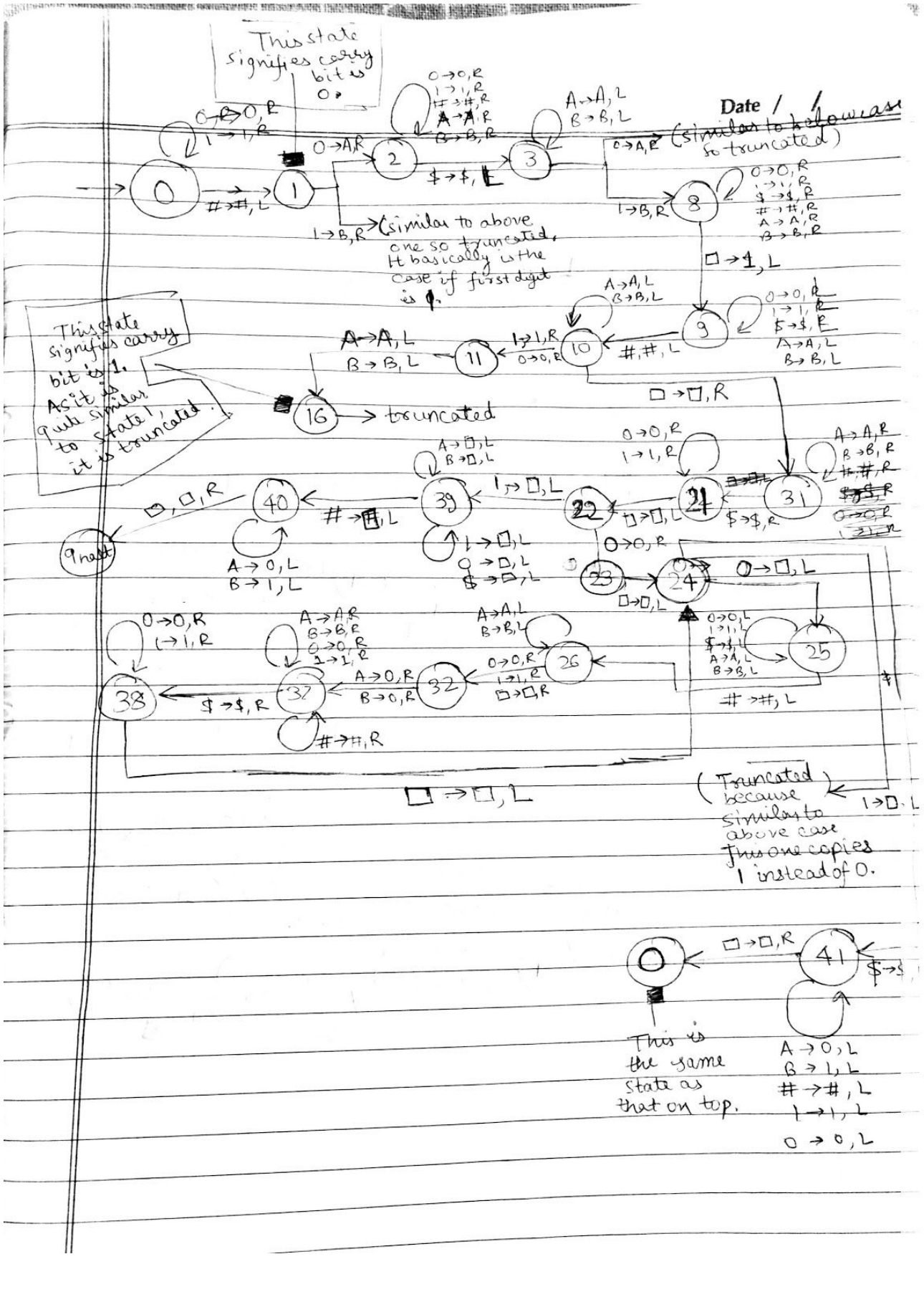
5. Find_remainder: This subroutine uses repeated subtraction to find remainder. For subtraction it uses similar logic as in a full adder circuit. To implement the truth table for modified full adder it uses branches for each case. Hence it contains many states as a few states are repeated 8 times. For a simple diagram, I will only show one cases in the state machine. The subtraction output is appended at the end of input separated by a $ sign. Also the output is reversed. After subtraction it checks whether the last symbol is 1 or not.

   If it is 0, then it tries to copy this number to the beginning of a. After doing this it returns to the initial state to repeat subtraction till necessary.

   If it is 1 our task of finding remainder is done as after subtracting first 1 implies negative number which implies that in a#b a<b and so it is the remainder itself. However as I have replaced the 1s and 0s in a and b by Bs and As, we have to restore the number for output. At the end it erases all the symbols except the answer itself.

   The following state diagram contains 41 states but most of them has been truncated Because of their repeated nature and simplicity of the diagram.

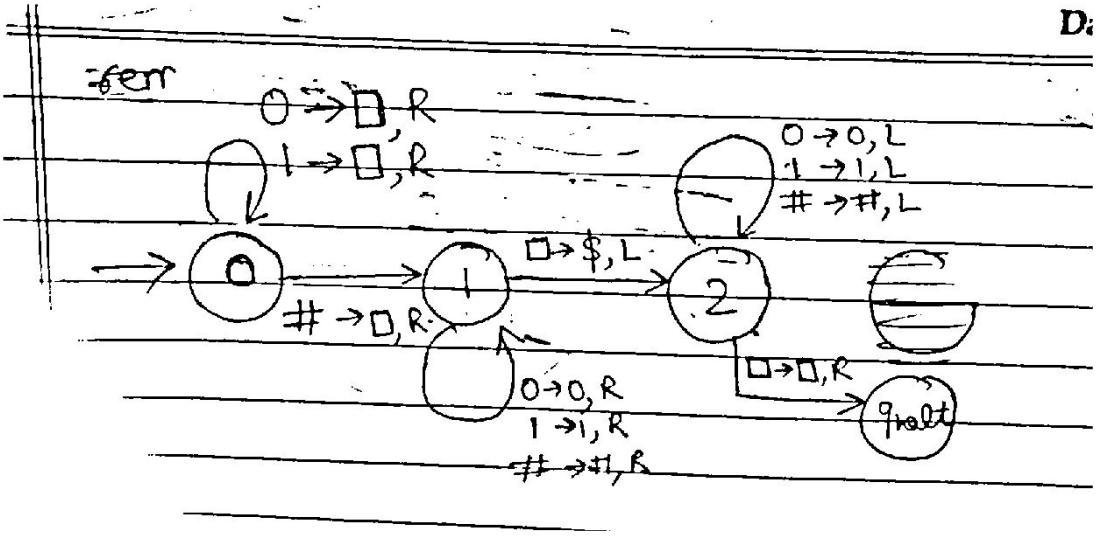
PS: Due to repeated subtraction, this method is slow when a is large and b is small. You can comment line 45 for quicker result because printing large amount of data slows down the program.


This state signifies carry bit is 0.
Date / /
(similar to lowercase so truncated)
(similar to above one so truncated. It basically is the case if first digit is 1.
This state signifies carry bit is 1. As it is quite similar to state 1, it is truncated.
truncated
(Truncated because similar to above case This one copies 1 instead of 0.
This is the same state as that on top.

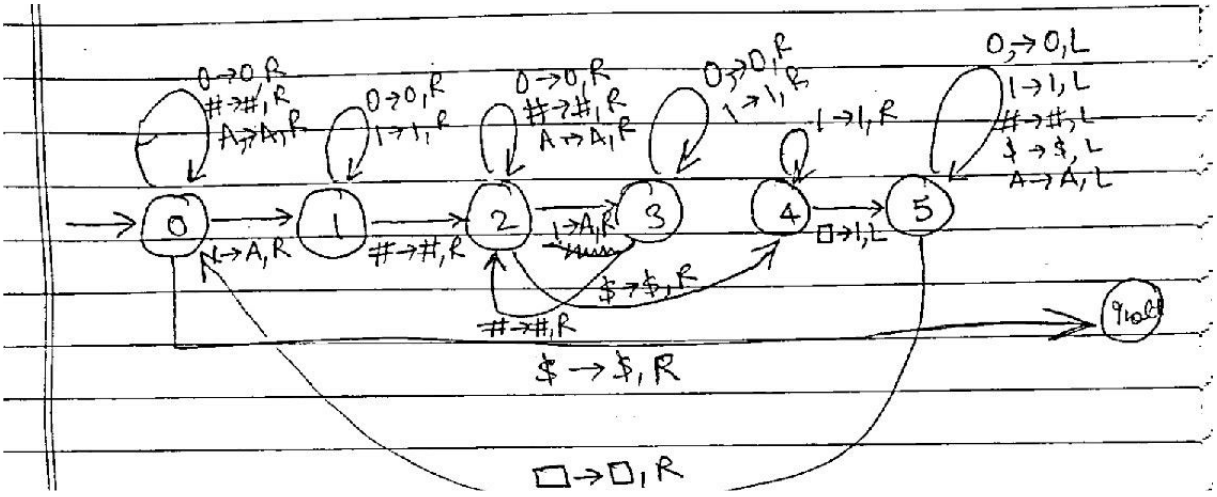
Q2: Find Max Degree in a Directed Graph.

Explanation: I have used 4 subroutines to perform this task. This includes

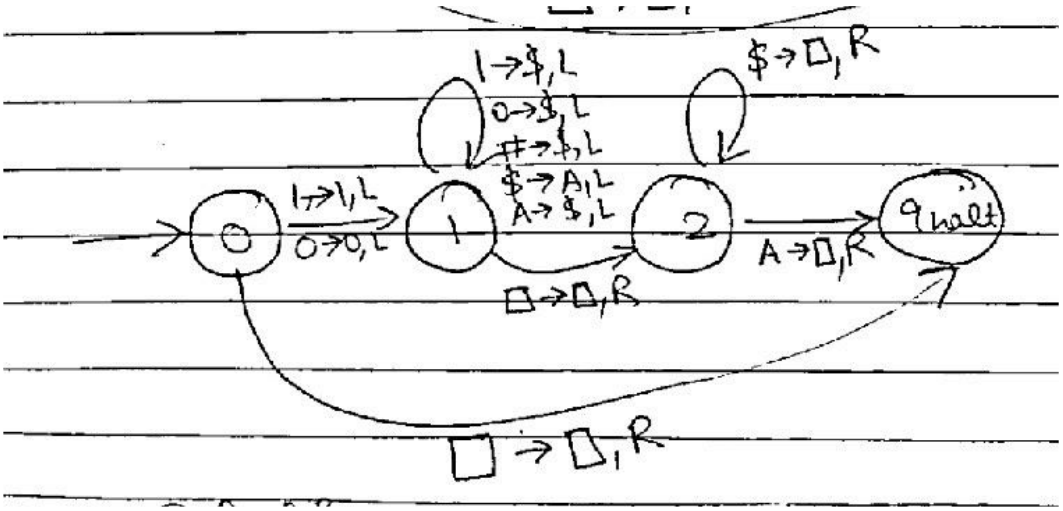
1. Remove_n: This subroutine erases the number of vertices along with one '#'. It basically traverses the input till first # is found erasing everything till it. Then it proceeds to the end adding a $ symbol. Then it heads it initial of the input where the adjacency matrix's first row starts.



2. Max_degree: This subroutine is the main subroutine which essentially finds the maximum number of ones and appends it to the tape after the $ symbol. The degree is stored in unary format. It tries to find the first one and replaces it with A. After that it moves to each row and changes exactly one 1 to A. When it encounters a $ symbol it then moves to the end of the tape and adds a 1 to it. Then it goes back to the start of the tape and the process begins again.



3. Clear_prev: This is just a helping subroutine which erases all the symbols before and including the '$ symbol. This results in tape containing only ones. This prepares the tape for the final subroutine which changes the degree in unary to binary format.



4. Convert_to_binary: It takes the 1 given in unary format and tries to remove 1 from them and add another 1 in binary format left of the ones present in unary format. At the end it erases all the 1s in unary format(which has been converted to #(similar to 1')). The tape is left with only the maximum degree in binary format where tape_head is pointing to the beginning of the answer.