CS 181 HW 4

Problem 1

Design a TM that recognizes $L = \{x : \text{number of 1's in } x \text{ is at least twice the number of 0's}\}$

Procedure :

This means that TM will return 1 as long as meet the minimum requirement which is there are twice as many 1's as 0's.

High level Description :

① First, scan the entire tape & look for the first 0 which has not been marked

(1.1) If we find the first unmarked 0, then we marked it as read. After that, we move the arrow (head) back to the start of the tape.

(1.2) If we cannot find the first unmarked 0, then we move on to clean up process & accept the tape

② Second, we scan the tape again & look for an unmarked 1

(2.1) If we find the first unmarked 1, marked the 1 as read. After that, move the arrow back to the start of tape

(2.2) If we cannot find the first unmarked 1, move on to the clean up process & reject the tape.

③ Scan the tape once again & looked for an unmarked 1 again

③.① If we find another unmarked 1, marked the 1 as read.

③.② If we cannot find another unmarked 1, move on to the clean up process & reject the tape

④ Move the arrow (head) back to the start of the tape & go to stage ①

More Details about the implementation

① $s(o, o) = (\text{GoStart}, a, L)$; we at the initial state, then we found & read 0, mark it as $a \equiv$ read, then move left back to the head of the tape.

② $s(o, 1) = (\text{Search}0, 1, R)$; we at the initial state, then we read 1, keep moving right & search for unmarked 0.

③ $s(o, \phi) = (\text{Write}1, \phi, S)$; we at the initial state, then we reach the end of string; clean up & accept the tape

④ $s(\text{GoStart}, 0) = (\text{Search}1, 0, R)$; after we find the first 0, then we want to find the first 1, keep searching & move to the right.

⑤ $s(\text{GoStart}, 1) = (\text{Go Start}, a, L)$; The found sequence 0, 1. Then, move back to head to search for seand 1.

⑥ $s(\text{GoStart}, a) = (\text{Search}1, a, R)$

⑦ $s(\text{GoStart}, \phi) = (\text{Write}0, \phi, S)$

⑧ $s(\text{Search}0, 0) = (\text{GoStart}, a, L)$

⑨ $s(\text{Search}0, 1) = (\text{Search}0, 1, R)$

⑩ $s(\text{Search}0, a) = (\text{Search}0, a, R)$

⑪ $s(\text{Search}0, \phi) = (\text{Write}1, \phi, S)$

⑫ Write 1 ≡ Keep going left until you see start.
Write 1 & Halt.

(13) Write 0 ≡ Keep going left until you reach start. Write 0, halt.

(14) $s(\text{Search } 1, 0) = (\text{Search } 1, 0, R)$

(15) $s(\text{Search } 1, 1) = (\text{Go Start}, a, L)$

(16) $s(\text{Search } 1, a) = (\text{Search } 1, a, R)$

(17) $s(\text{Search } 1, \phi) = (\text{Write } 0, \phi, S)$

(18) $s(o, a) = (\text{Search } o, a, R)$

Problem 2 Prove that for every function $F: \{0,1\}^* \to \{0,1\}^*$, F is computable by a standard Turing machine iff F is computable by a 2 tape Turing machine.

Proof (=>: We want to show given F is computable by 2 tape turing machine, then F is computable by a standard turing machine.

We can prove this by translating the 2 input tapes to one input tape

① First, we concatenate the 2 input tapes as one input with a delimiter # between the tapes

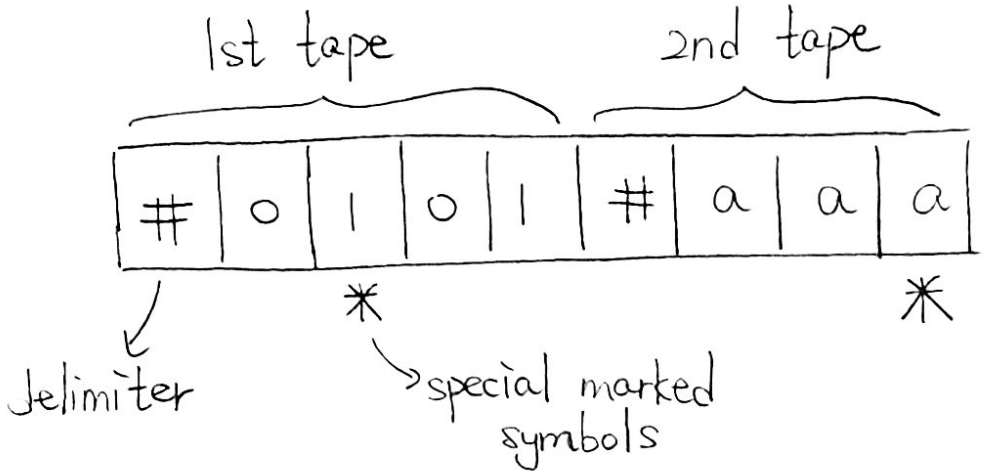
② Since we have 2 heads, we set up 2 special markers to remember the positions of the 2 heads.

③ In order to simulate one iteration of the 2 tape turing machine, we scan the tape to the left & remembering each of the 2 values where the 2 heads are.
Then, we make all of the changes the 2 tape turing machine would, for example moving back to the right & perform the changes to the cells of the tape.

④ If any of 2 heads has to move onto symbol #, before the next character of input, then we need to shifting everything on the tape to the right.
This will generate one extra empty cell for the movement of the head. The purpose of the act is to

simulate the fact that each of the 2 tape has an infinite amount of working space when they are being combined into a standard tape turing machine.

Sample picture of combine 2 tapes to 1 tape



Proof =): We want to show given F is computable by a standard turing machine, then F is computable by a 2 tape turing machine.

We can prove this by translating the 1 input tape to 2 input tapes.

① First, we can separate the 1 input tape to 2 track tape which are called upper track tape & lower track tape

② Upper track tape represents the cells to the right of the initial start position.

③ Lower track tape represents the cells to the left of the initial start position in reverse order.

④ The machine starts from the initial state called q_0.
For the modified 1 input tape, it has an left end but does not have an right end. The left end is marked with a marker.

⑤ The head scans from the left end marker "End".

⑥ For each move on the tape, it reads the symbol on the tape pointed by the head.

⑦ The machine writes a new symbol on the current cell of the tape.

⑧ The machine then moves the head either left or right one cell on tape. The movement of the head

determined by the transition functions.

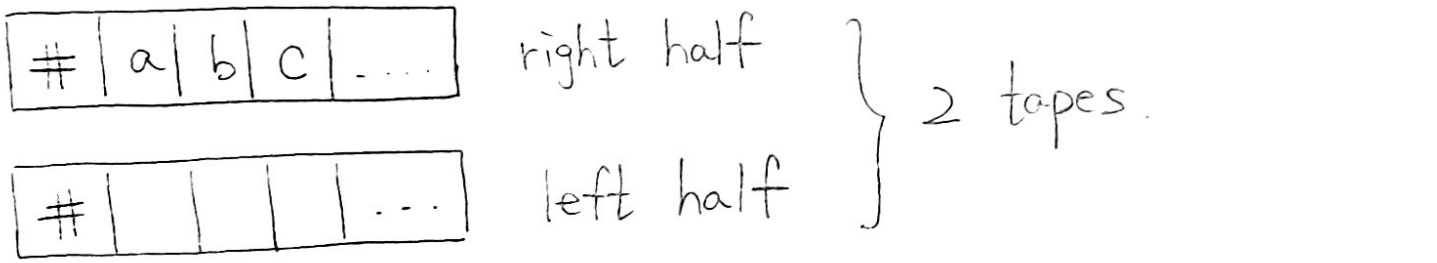
⑨ There are 2 specified states which are called accept states and reject states.

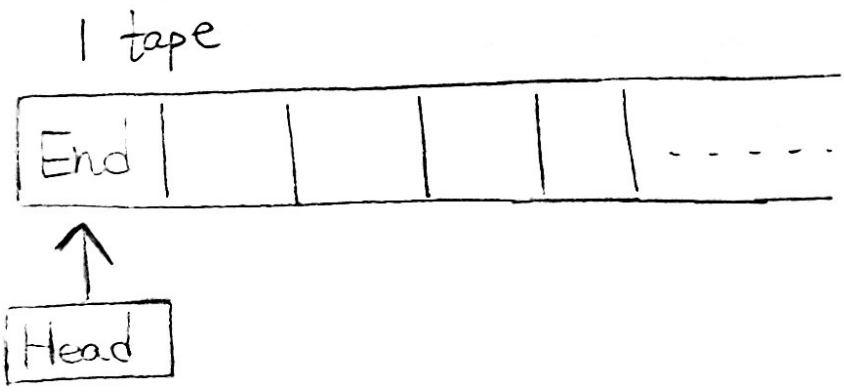
⑩ If the head end up entering into the accept states, then the turing machine accept the input.

⑪ If the head end up entering into the reject states, then the turing machine reject the input.

⑫ However, there could be cases that the machine continue to run without accepting or rejecting for some special or specified input symbols.

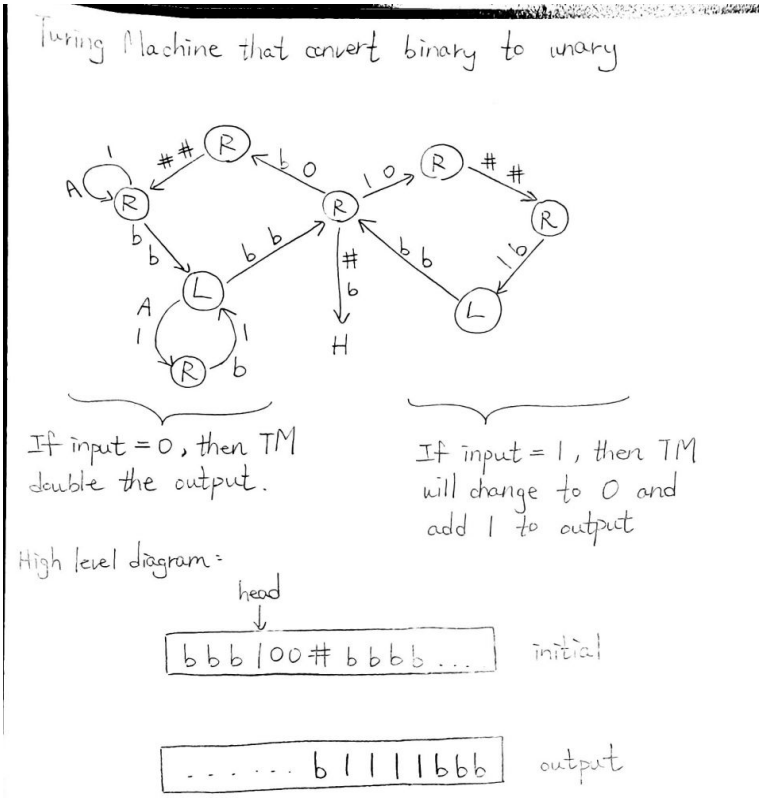
Picture





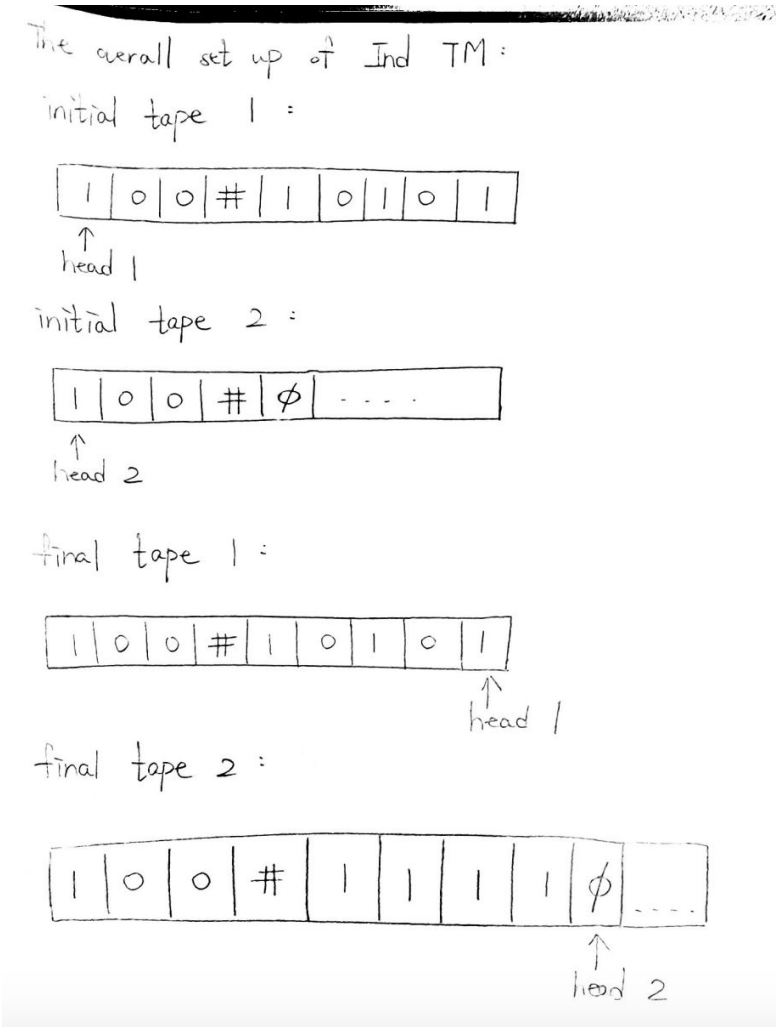
Steps:

1. First, we need two tapes. Tape 1 contains the input i#x. Tape 2 contains a copy of i#.
2. On tape 1, we will copy the symbol in each cell to tape 2 until we reach #. It is because # indicates the end of i which is binary representation of the integer.
3. After we copy i# to the tape 2. We need to convert the binary representation into unary on tape 2 before we start indexing on the tape 1. The conversion is being conducted on the tape 2. The idea for this conversion is it will make a string of 1's to the right of the binary number after delimiter #. Every time it subtracts 1 from the number, it should add an 1 to the string.
    a. If the number has not reached 0 yet, keep subtracting 1 from the number and add 1 to the string on tape 2
    b. If the number has reached 0, the string of 1's complete on tape 2.
4. The following diagram illustrate the binary to unary conversion:



    a.
5. The final output format on the tape 2 is i#unary. Tape 1 still remains the same now.
6. Then, we need to start indexing on tape 1 according to the unary number from tape 2.
7. Both tape 1 and tape 2, scan to the right until reach #.

8. Then, tape 2 scan to the right by one cell to check if it is empty:
    a. If the first character in the unary number in tape 2 is empty, then tape 1 scan to the right by one cell (right after #), then return the corresponding symbol in x[i] on tape 1.
    b. If the first character in the unary number in tape 2 is 1, then tape 1 scans to the right by two cells to start.
        i. **On tape 2, scan to the right by one cell to check if it is empty:**
            1. If it is empty, then return the corresponding symbol in x[i] on tape 1
            2. If it is not empty, move the head of tape 1 to the right by one cell. **Repeat step 8.b.i (the bolded step)**



Problem 4

Show that computable functions are closed under the "concatenation" operation: If $F, G = \{0,1\}^* \to \{0,1\}$ are 2 functions, ..., show that if F, G are computable, then so is H.

Proof:

High level Idea:

① If we want to know whether the input is in the concatenation of F & G. Then, we need to divide the input x into 2 parts as $x = x_1 \circ x_2$.

② Therefore, we need 3 tape for this machine. 1st tape stores the original loaded input. 2nd tape stores x_1; 3rd tape stores x_2.

③ Let M_1 be the machine that computes F;
Let M_2 be the machine that computes G

④ After that we run M_1 on x_1 & M_2 on x_2.
If both M_1 & M_2 accept, then return accept.
If either M_1 or M_2 reject, then return reject.

⑤ If input x is accepted by the machine M' that computes H, then there is a way that we can split x such that M_1 accepts x_1 and M_2 accepts x_2. Then M' accept input x.

⑥ If input x is not accepted by the machine M' that computes H, there is no way that we can split x such that M_1 accepts x_1 and M_2 accepts x_2 Then, M' rejects input x.

More Detail on 3 tape TM machine M' :

① First, split the input string into 2 parts $x = x_1 \circ x_2$.

② Copy x_1 on 2nd tape & Copy x_2 on 3rd tape.

③ Run M_1 on x_1 for 2nd tape.

④ If M_1 accept x_1, then proceed to ⑤.
If M_1 reject x_1, then M' reject.

⑤ Run M_2 on x_2 for 3rd tape

⑥ If M_2 accept x_2, then M' accept.
If M_2 reject x_2, then M' reject

Remarks : We can try every possible splits to divide x into x_1 & x_2 when we are searching for a way.