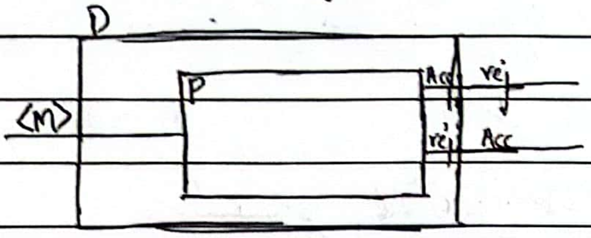
MAAZ KARIM
24503 Date ______

1) Let us assume P() exists and halts for every input.
We use that to create a new machine D.

D takes as input ⟨M⟩ and uses ~~as~~ P() as a subroutine, such that if P() accepts, D rejects and if P() rejects, D accepts.

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } P(\langle M \rangle) = \text{rej} \\ \text{reject} & \text{if } P(\langle M \rangle) = \text{rej} \end{cases}$$



Now let us take the case where ⟨M⟩ = ⟨D⟩ since ⟨M⟩ could be the encoding of ~~this~~ any machine.

⇒ Now if P(⟨D⟩) returns ~~true~~ accept, that means D(⟨D⟩) returns reject, implying that ⟨D⟩ ∉ L(D), which means P(⟨D⟩) should've returned reject.

⇒ Similarly if P(⟨D⟩) returned reject, that means D(⟨D⟩) returns accept, implying ~~⟨D⟩ ∉~~ ⟨D⟩ ∈ L(D), which means P(⟨D⟩) should've returned accept.

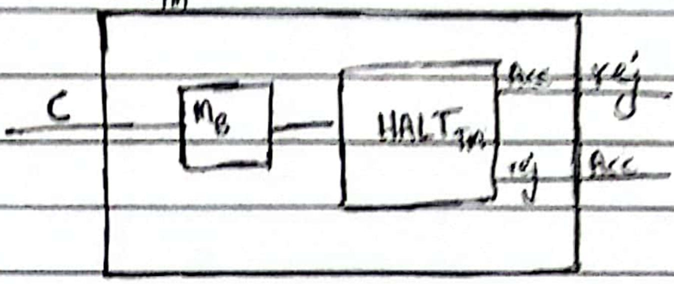
⇒ So P(⟨D⟩) can't accept, reject or go into a loop without violating its conditions.

Since we have reached a contradiction, our previous assumption of L(P) being decidable is wrong and the machine P cannot exist.

Date

2) We create a new machine $TILE_{TM}$ which uses the $HALT_{TM}$ as ~~for~~ a subroutine in the following way.

$TILE_{TM}$



M_B ~~does the following~~ writes the encoding for the following program on the tape:

```
for i from 1 to ∞{
    check ← false
    for j from 1 to |C|^(i²) {          ← Number of i by i combinations possible from set C
        combination ← generate_new_pattern(i)     ← creates new pattern of ixi tiles
        if (is_legal(combination)){     ← Checks if no coinciding borders have the same color
            check ← true
            break from inner lopp.}
    }
    if (not check) return true     ← Halting condition
    i++
}
```

This code halting ⇒ There exists an ~~[illegible]~~ i for which now combination of ixi tilings from C is legal which would mean ~~[illegible]~~ $C \notin L(TILE_{TM})$.

This program never halting ⇒ a legal combination exists $\forall n \in \mathbb{Z}^+ \therefore C \in L(TILE)$

$\therefore$ We have shown that $TILE \leq_T HALT$

3)a) Since the COUNT problem takes as input a fixed integer n, we have to consider a finite number of n-state turing machines.

Our program works in the following way. (i starts at 0)

```
for all n-state TM's {
    ~~Create_new_TM~~
    Temp ← Create_new_TM(n)
    if (HALT_TM(Temp, BLANK))
        i++;
}
return i;
```

$\therefore$ COUNT $\leq_T$ HALT

Date

3) b) We create $HALT_{TM}$ using $COUNT_{TM}$ ~~such that~~ in the following way.

~~We count the number of states given as in~~

⇒ We modify input $\langle M, w \rangle$ such that we create a new machine M_w which has w hard coded so it runs ~~on a~~ starting on a blank tape.

⇒ We ~~inp~~ count the number of states in M_w and create all valid TM's with these many states and start simulating all including M_w on a blank tape.

⇒ We input the No. of states into $COUNT_{TM}$ to get $H(n)$.

⇒ We keep count of number of machines that have halted.

⇒ When $H(n)$ = No. machines halted (which is guaranteed to happen by def of $COUNT_{TM}$), ~~&~~ we check if M_w has halted or not.

⇒ If M_w has halted, Return true, else return false.

$\therefore$ $HALT \leq_T COUNT$

Date

3)c) Let M be a machine ~~that~~ constructed in the following way

⇒ M takes input ⟨n, a⟩.
⇒ it ~~f~~ creates all possible TM's with n-states.
⇒ As soon as a or more turing machines halt, it ~~return~~ halts and returns accept.

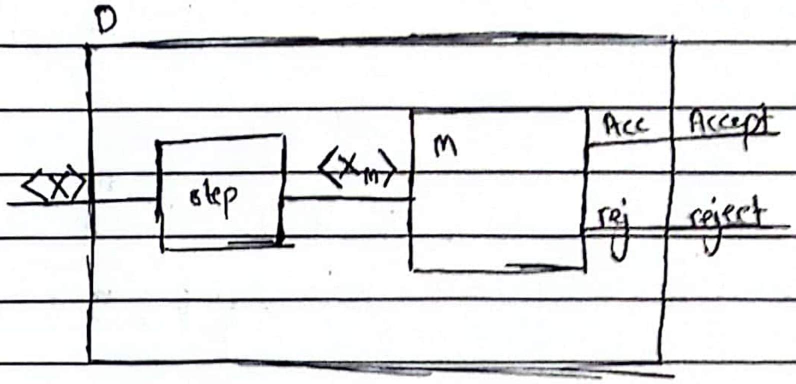
This means that if $\langle n, a\rangle \in L$ there must be atleast a out of ~~these~~ all n-state machines that will halt, implying $M(\langle n, a\rangle) = \text{Accept}$.

∴ L is Turing-Recognizable.

Date

4) Let's assume FTSOC, there does exist a TM M.

We create a new TM D as follows.



Step:

Write the encoding for a program that does as follows:

⇒ Takes input string s.
⇒ Runs the encoding $\langle x \rangle$ on itself for $|s|$ steps.
⇒ If the simulation does not halt in $|s|$ steps, return accept.
⇒ If simulation halts ~~return reject~~ and accepts, we accept, ~~if it rejects then~~ we reject;

If X halts on $\langle x \rangle$, there must exist a finite number of steps n that it halts in. Which means any string s, such that $|s| > n$ will be rejected by X_m making its language finite.

So $\langle x \rangle \in L(x) \longrightarrow L(X_m)$ is finite

and $\langle x \rangle \notin L(x) \longrightarrow L(X_m) = \Sigma^* \leftarrow$ infinite

This means that $\langle x \rangle \notin L(x) \rightarrow M(\langle x_M \rangle) = \text{Accept}$.

So $DIAG \leq_M L(M)$ and $L(M)$ is TR $\rightarrow$ DIAG is TR

contradiction.

$\therefore$ M cannot exist