REPORT

NAME: VANSH MARDA

ROLL NO: 2021101089

BRANCH: IIIT H CSE

Approach explained:

Algorithm:

My program is divided into 2 functions

1)solve () - for handling cases where there is more than one transition from a given state.

2)loop_solver() - for handling the cases where there is only one transition from the given state.

Algorithm for **solve** function:

Step1: Find the next parent state (R/P/S) that the user should play.

=> For this I have counted the most occurring next transition of the system.

If it is R, then our next parent state would be P

R=>P

P=>S

S=>R

Step2: Program stores the transitions of R, P, S played by the system on inputting the parent state to it. It is stored in 3 different arrays r,p,s .

Step3: Program then first checks for the rock transition … it first checks if the next transition has previous occurred with the same parent and same transition. If yes, then we will loop the transition back to its previous state.

If not, then we will first store the transition with its parent and index number in a storage vector. And then again call solve () with array input as r.

We will repeat the same steps for p, s vectors also.

Algorithm for **loop_solver** function:

We will enter the loop_solver function if the size of input array to function solve is 1.

Step1: In the first step we will check if the next transition is already in the ans array or not. For checking this we must store each transition in an array which I have called as arr .If the transition is already in the ans array then we will loop back to the same transition and then return.

Step2: If we do not return, then we will first store the transition in arr and then we will find the parent (next thing the user will play) with the same logic as in function solve.

Step3: we will find the next transition that the system will have when we play the parent on the current transition. And then we will recursively pass this array in the loop solver ().

Thank you.

Vansh Marda.