Daffodil International University

Faculty of Science & Information Technology

Department of Software Engineering

Assignment

On

Midterm Question Solve

Course Code: SWE 424

Course Title: Artificial Intelligence

Submitted To:

Mr. K. M. Qaiduzzaman

Lecturer

Department of Software Engineering

Submitted By:

Md. Maksudur Rahman
Id: 163-35-1778

Level: 3 Term: 3

Section: B

Ans to the Q no.1

Percept:

variable	values
loc	UC, PC, CC
P_loc	UC, PC, CC
status	S>T, S<T

Ans to the Q no.2

Action:

variable	values
movement	move_to_UC move_to_PC move_to_CC stop

Ans to the Q no.3

Environment fore this problem:

Observevable; Fully

Determinintic; Determinintic

Episodic; Episodic

Static; Static

Dincrete; Dincrete

Single Agent; Single

Ans to the Q no.3

I will build simple reflex agent with state to solve the problem.

Because in this case the agent have to remember his past location, otherwise the agent will move to the same ~~loop~~ location every time. For example:

City ⟶ Uttarea ⟶ City ⟶ uttarea →----

If the agent doesnot ~~follow~~ remember his previous location he will be in a loop like above situation.

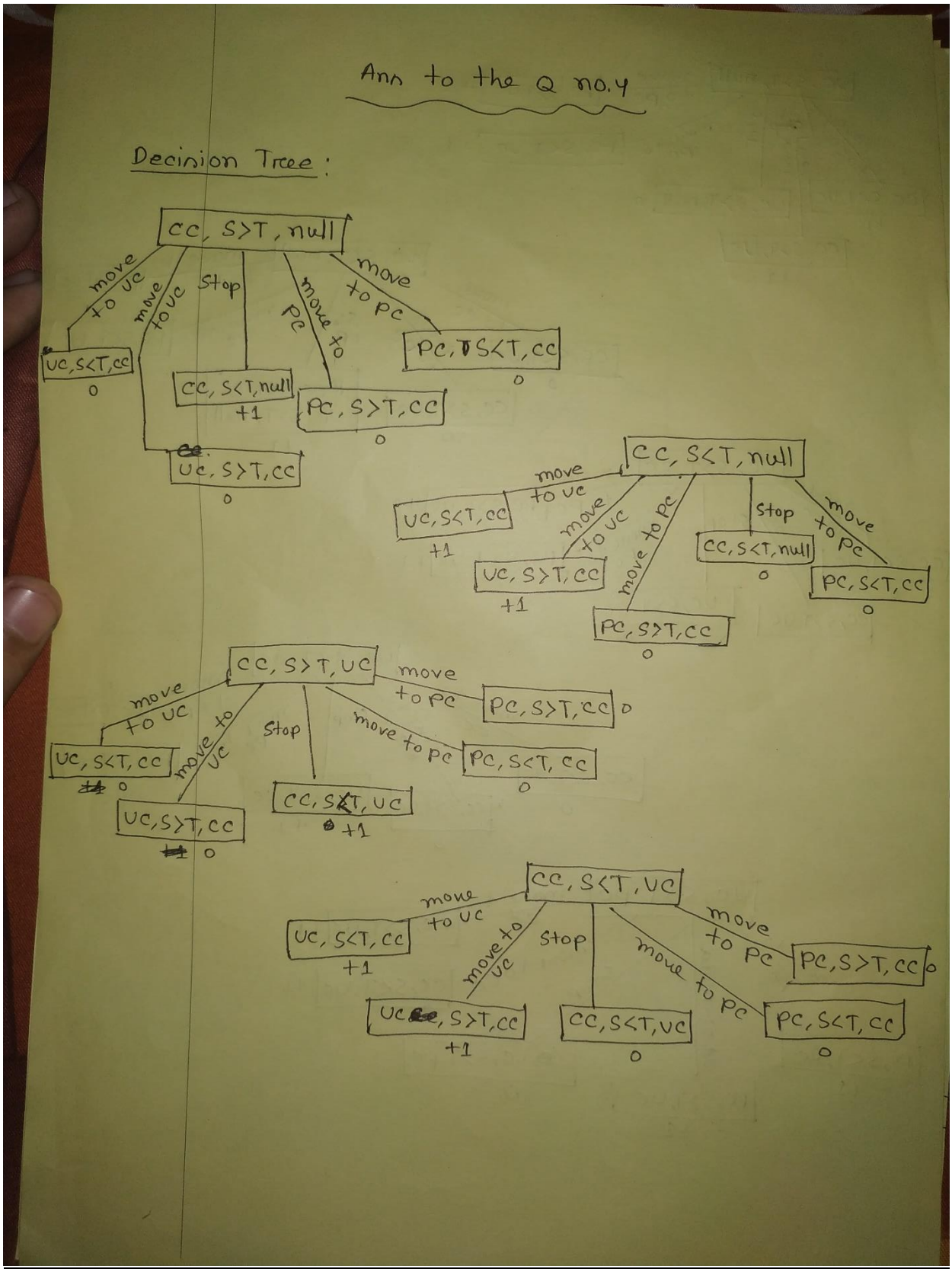
And what if the agent follow simple reflex with state and remember previous location then he can move like this:

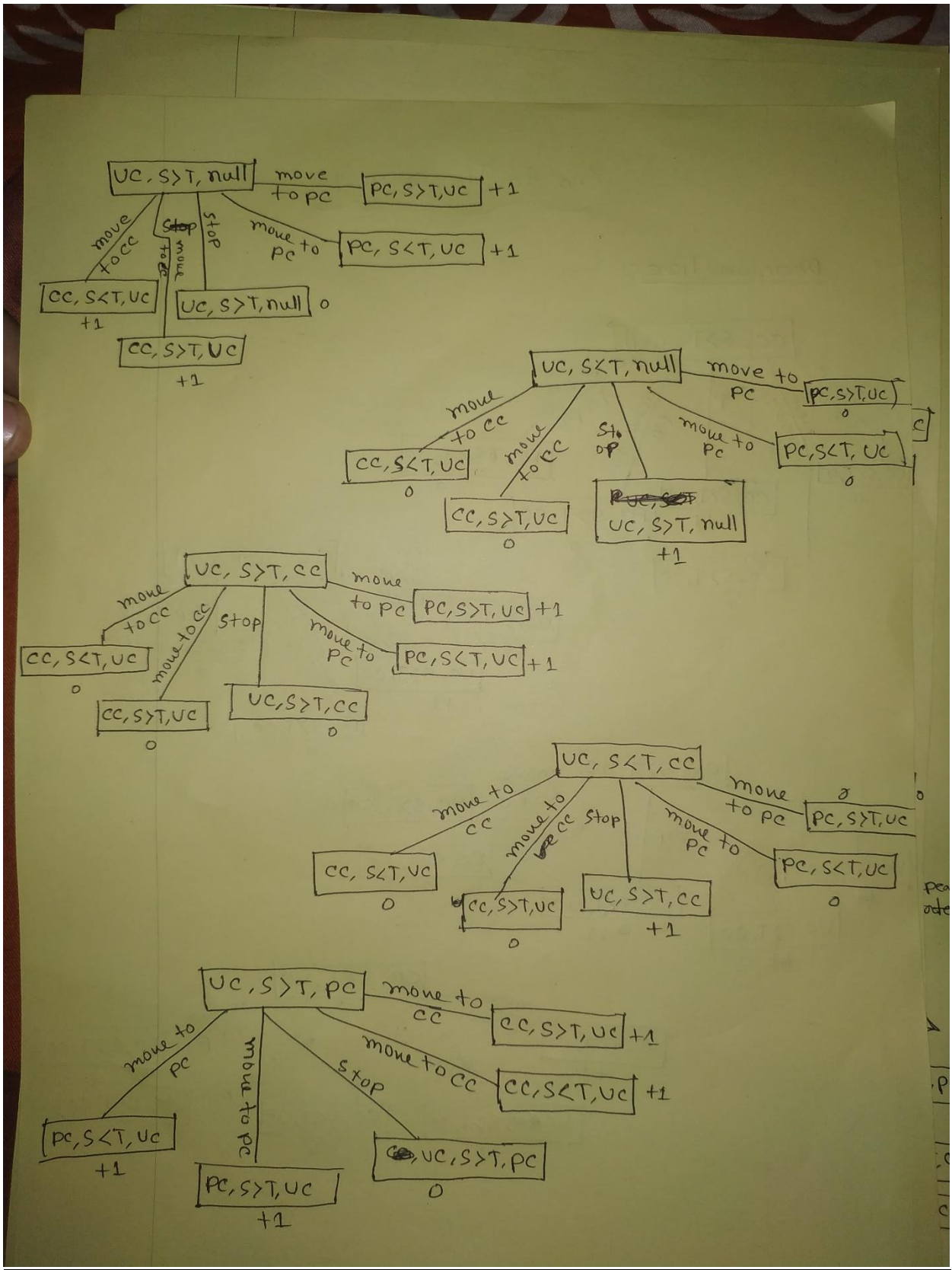
City → uttarea ⟶ permanent ⟶ uttarea ⟶ city

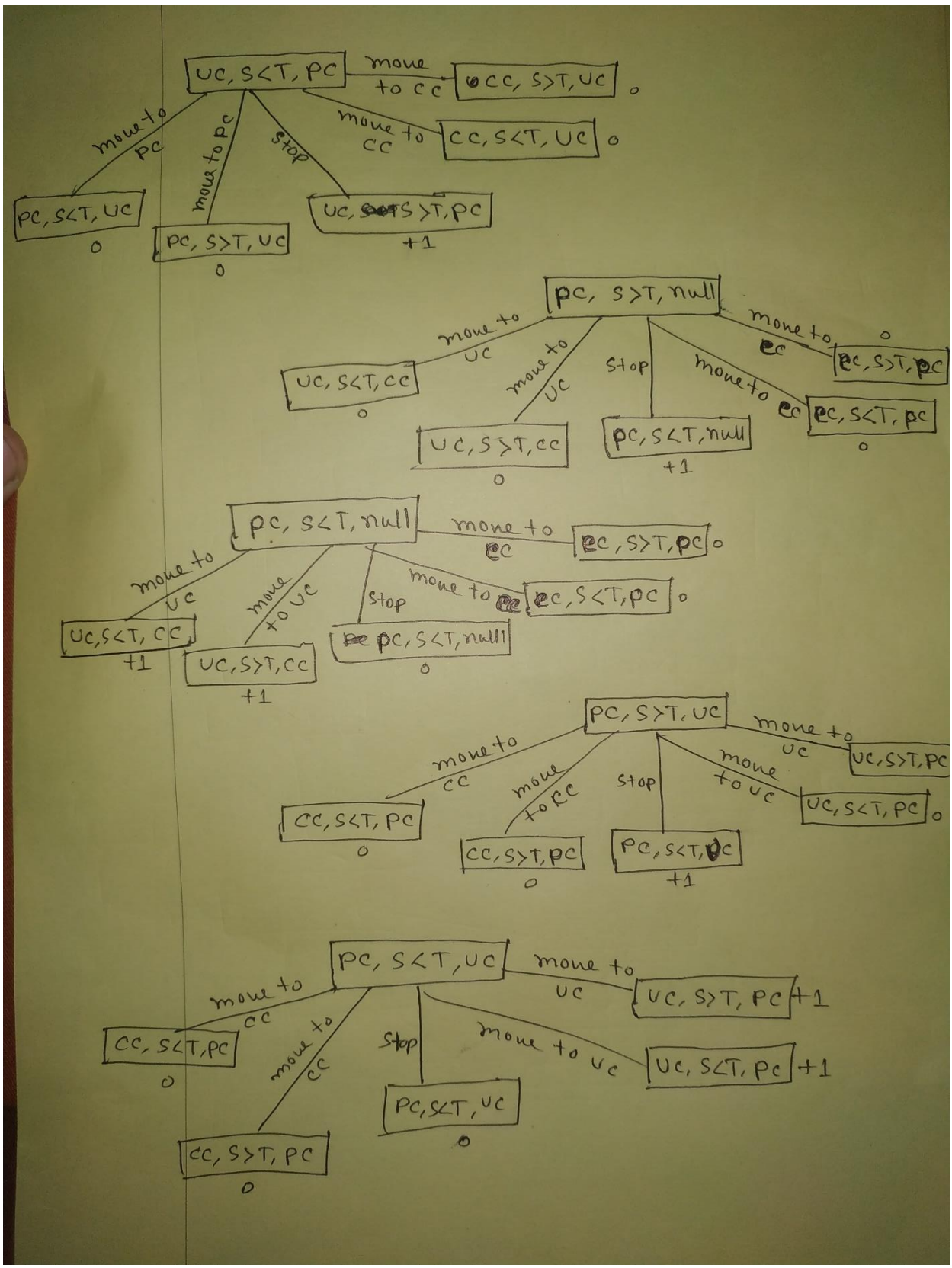
So, in this case we have to follow simple reflex agent with state to solution.

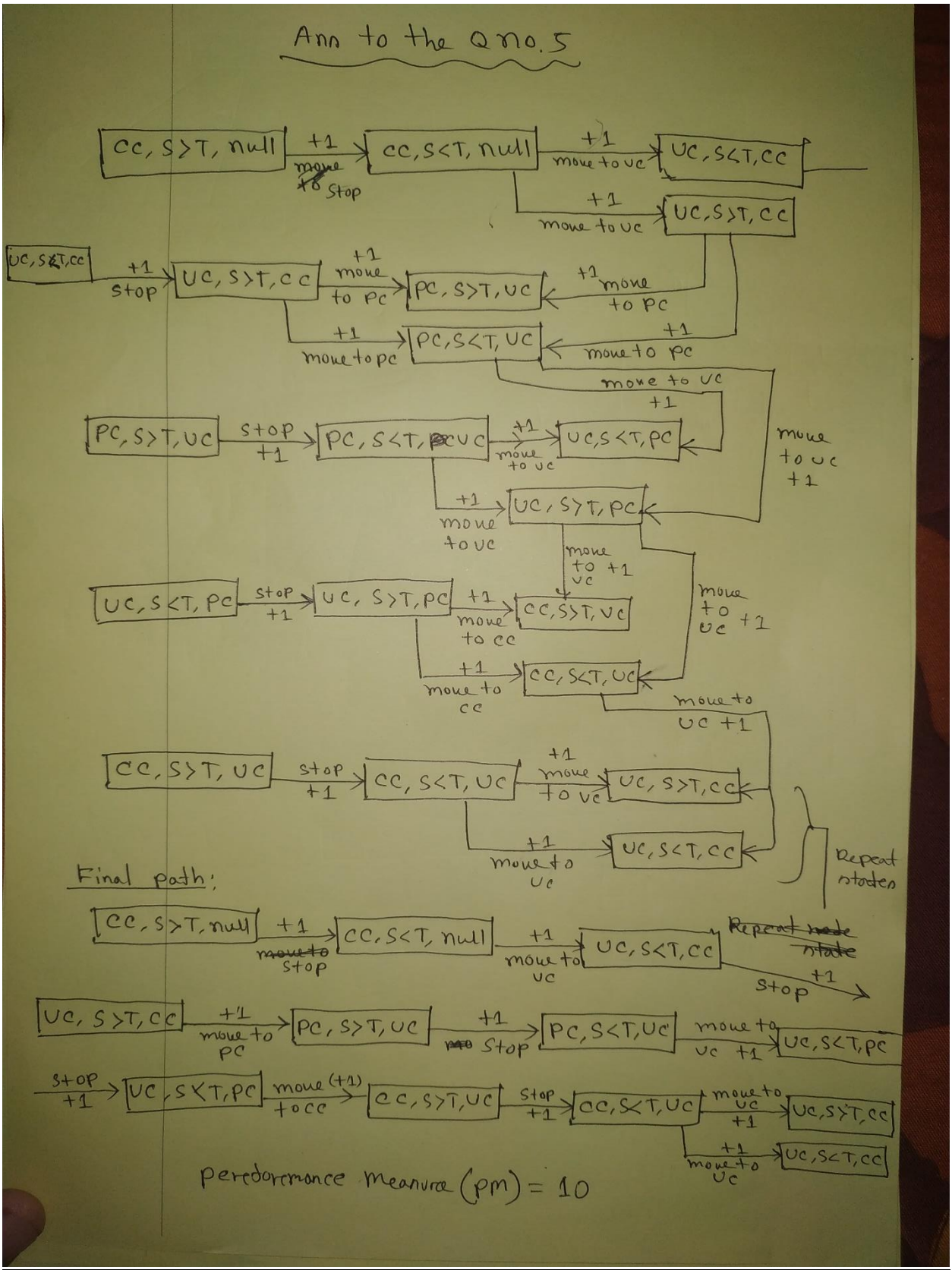
Ans to the Q no.4

Decision Tree:




UC, S>T, null
move to pc
PC, S>T, UC
+1
move to cc
stop
move to pc
PC, S<T, UC
+1
CC, S<T, UC
+1
UC, S>T, null
0
CC, S>T, UC
+1
UC, S<T, null
move to pc
PC, S>T, UC
0
move to cc
stop
move to pc
PC, S<T, UC
0
CC, S<T, UC
0
CC, S>T, UC
0
UC, S>T, null
+1
UC, S>T, CC
move to pc
PC, S>T, UC
+1
move to cc
stop
move to pc
PC, S<T, UC
+1
CC, S<T, UC
0
CC, S>T, UC
0
UC, S>T, CC
0
UC, S<T, CC
move to cc
move to cc
stop
move to pc
PC, S>T, UC
move to pc
PC, S<T, UC
0
CC, S<T, UC
0
CC, S>T, UC
0
UC, S>T, CC
+1
UC, S>T, PC
move to cc
CC, S>T, UC
+1
move to cc
CC, S<T, UC
+1
move to pc
stop
move to pc
PC, S<T, UC
+1
PC, S>T, UC
+1
UC, S>T, PC
0


uc, s<T, pc
move to cc
cc, s>T, uc
0
move to pc
cc, s<T, uc
0
move to cc
stop
pc, s<T, uc
0
pc, s>T, uc
0
uc, s>T, pc
+1
pc, s>T, null
move to uc
move to cc
stop
move to cc
uc, s<T, cc
0
uc, s>T, cc
0
pc, s<T, null
+1
pc, s>T, pc
0
pc, s<T, null
move to cc
pc, s>T, pc
0
move to uc
move to uc
stop
move to cc
uc, s<T, cc
+1
uc, s>T, cc
+1
pc, s<T, null
0
pc, s<T, pc
0
pc, s>T, uc
move to cc
move to uc
stop
move to uc
cc, s<T, pc
0
cc, s>T, pc
0
pc, s<T, pc
+1
uc, s>T, pc
uc, s<T, pc
0
pc, s<T, uc
move to cc
move to cc
stop
move to uc
cc, s<T, pc
0
cc, s>T, pc
0
pc, s<T, uc
0
uc, s>T, pc
+1
uc, s<T, pc
+1


Ans to the Qno.5
cc, S>T, null
+1
move to stop
cc, S<T, null
+1
move to uc
uc, S<T, cc
+1
move to uc
uc, S>T, cc
uc, S<T, cc
+1
stop
uc, S>T, cc
+1
move to pc
pc, S>T, uc
+1
move to pc
+1
move to pc
pc, S<T, uc
+1
move to pc
+1
move to uc
pc, S>T, uc
stop
+1
pc, S<T, uc
+1
move to uc
uc, S<T, pc
move to uc +1
+1
move to uc
uc, S>T, pc
move to +1 uc
uc, S<T, pc
stop
+1
uc, S>T, pc
+1
move to cc
cc, S>T, uc
move to uc +1
+1
move to cc
cc, S<T, uc
cc, S>T, uc
stop
+1
cc, S<T, uc
+1
move to uc
uc, S>T, cc
+1
move to uc
uc, S<T, cc
Repeat states
Final path:
cc, S>T, null
+1
move to stop
cc, S<T, null
+1
move to uc
uc, S<T, cc
Repeat state
+1
stop
uc, S>T, cc
+1
move to pc
pc, S>T, uc
+1
stop
pc, S<T, uc
move to uc +1
uc, S<T, pc
stop
+1
uc, S<T, pc
move (+1) to cc
cc, S>T, uc
stop
+1
cc, S<T, uc
move to uc
+1
uc, S>T, cc
+1
move to uc
uc, S<T, cc
Performance Measure (PM) = 10

Ans to the Q no. 6

Agent function

```
Global variable: int P_loc == "null";
Agent_Function (int loc, ~~ntatun, P~~ int ntatun, ~~int P_loc~~) {

if (loc == cc && ntatun == S>T && P_loc == "null") {
    P_loc = loc;
    return ~~move_to~~ ntop; }

else if (loc == cc && ntatun == S<T && P_loc == "null") {
    P_loc = loc;
    return move_to_uc; }

else if (loc == cc && ntatun == S>T && P_loc == uc) {
    P_loc = loc;
    return ntop; }

else if (loc == cc && ntatun == S<T && P_loc == uc) {
    P_loc = loc;
    return move_to_~~pc~~ uc; }

else if (loc == uc && ntatun == S>T && P_loc == "null") {
    P_loc = loc;
    return move_to_pc; }

else if (loc == uc && ntatun == S<T && P_loc == "null") {
    P_loc = loc;
    return ntop;
}
```

```
else if (loc == uc && status == S>T && P_loc == cc){
        P_loc == loc;
        return move_to_pc; }
else if (loc == uc && status == S<T && P_loc == cc){
        P_loc == loc;
        return stop; }
else if (loc == uc && status == S>T && P_loc == pc){
        P_loc = loc;
        return move_to_pc; }
else if (loc == uc && status == S<T && P_loc == pc){
        P_loc = loc;
        return stop; }
else if (loc == pc && status == S>T && P_loc == "null"){
        P_loc = loc;
        return stop; }
else if (loc == pc && status == S<T && P_loc == "null"){
        P_loc = loc;
        return move_to_uc; }
else if (loc == pc && status == S>T && P_loc == uc){
        P_loc = loc;
        return stop; }
else if (loc == pc && status == S<T && P_loc == uc){
        P_loc = loc;
        return move_to_uc; }
    else
        return Invalid;
}
```