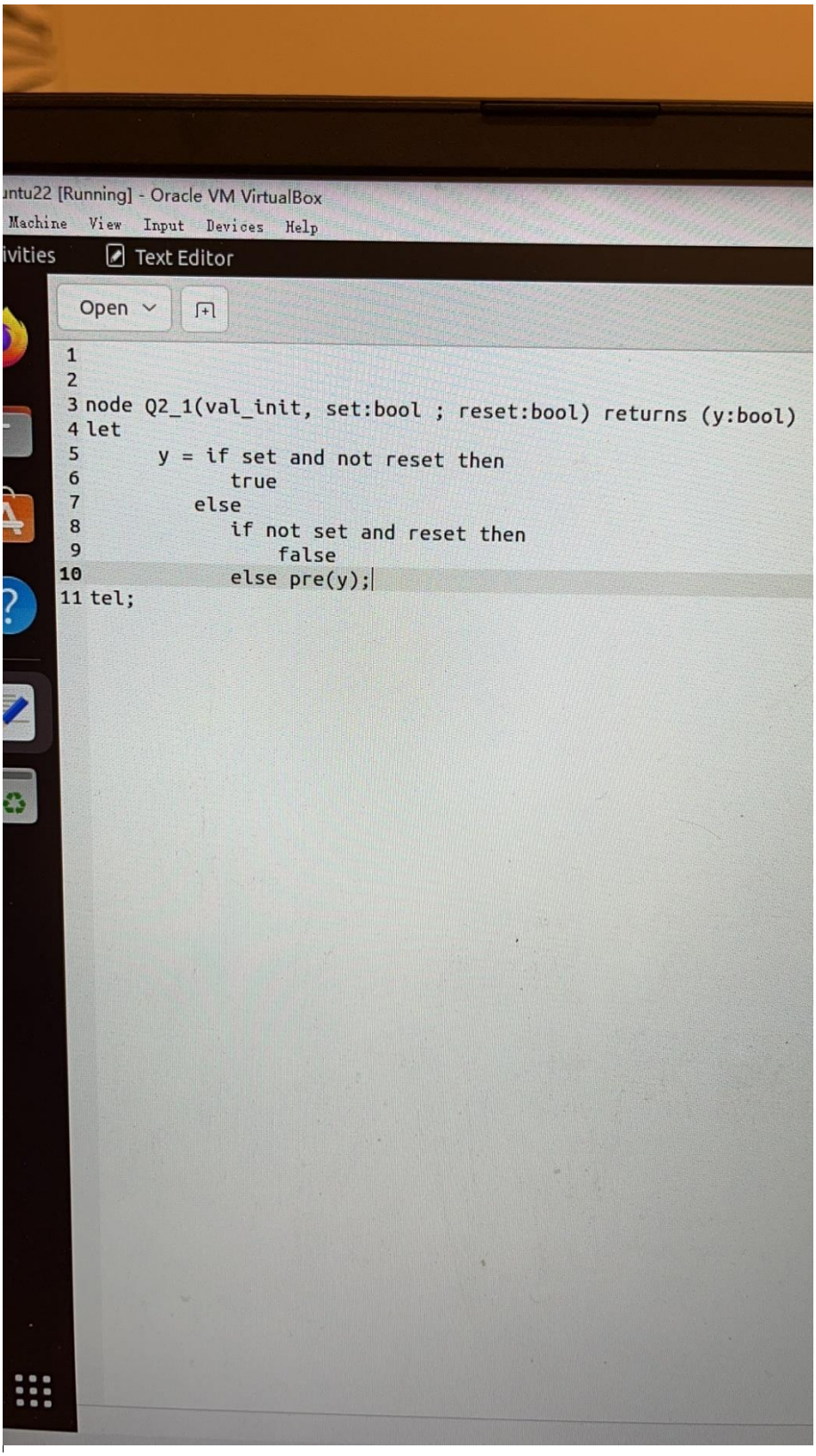
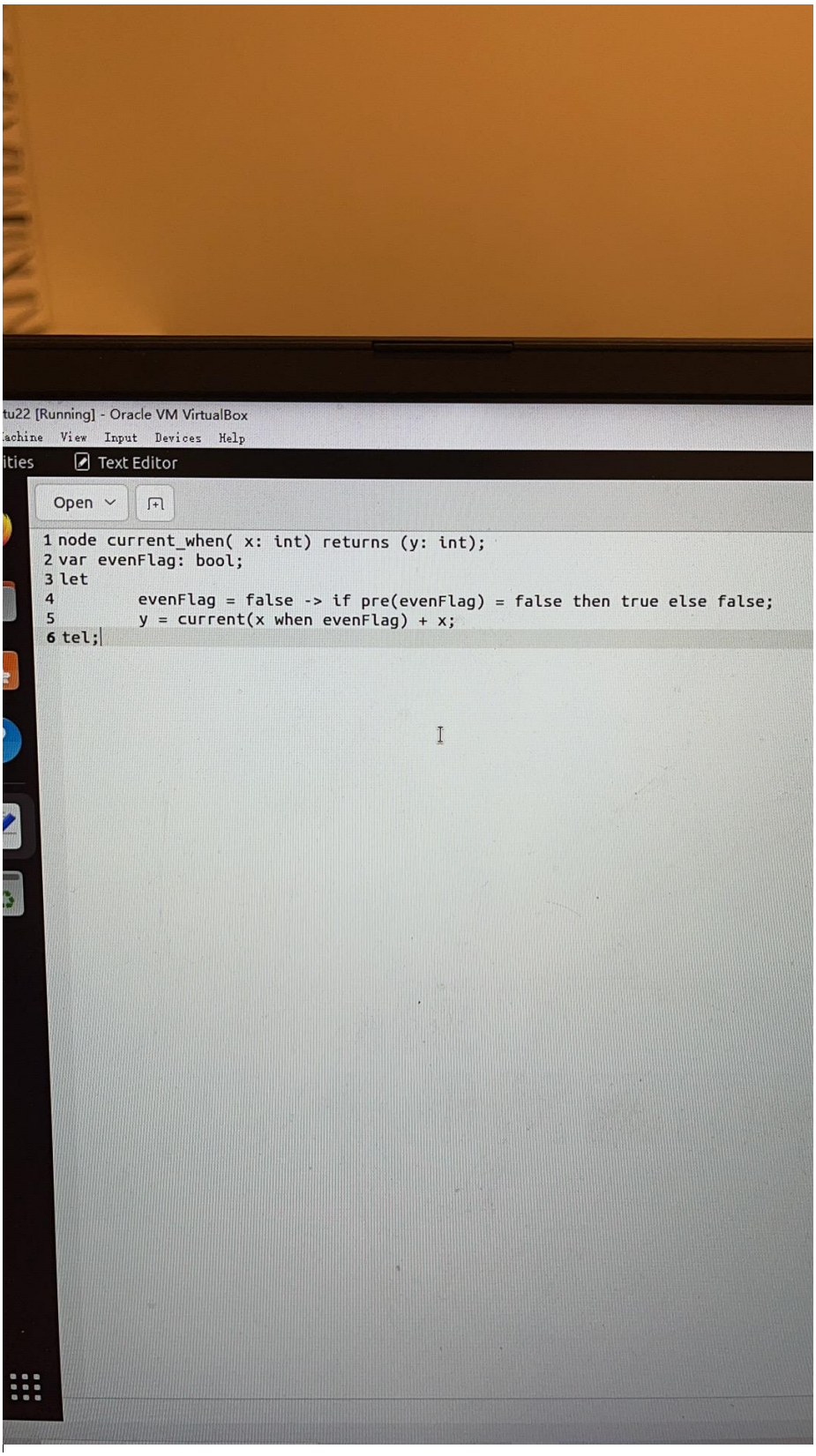
1.


ntu22 [Running] - Oracle VM VirtualBox
Machine View Input Devices Help
ivities
Text Editor
Open
node Q2_1(val_init, set:bool ; reset:bool) returns (y:bool)
let
y = if set and not reset then
true
else
if not set and reset then
false
else pre(y);
tel;

2. 
tu22 [Running] - Oracle VM VirtualBox
achine View Input Devices Help
ities
Text Editor
Open
1 node current_when(x: int) returns (y: int);
2 var evenFlag: bool;
3 let
4 evenFlag = false -> if pre(evenFlag) = false then true else false;
5 y = current(x when evenFlag) + x;
6 tel;

3.

Ubuntu22 [Running] - Oracle VM VirtualBox

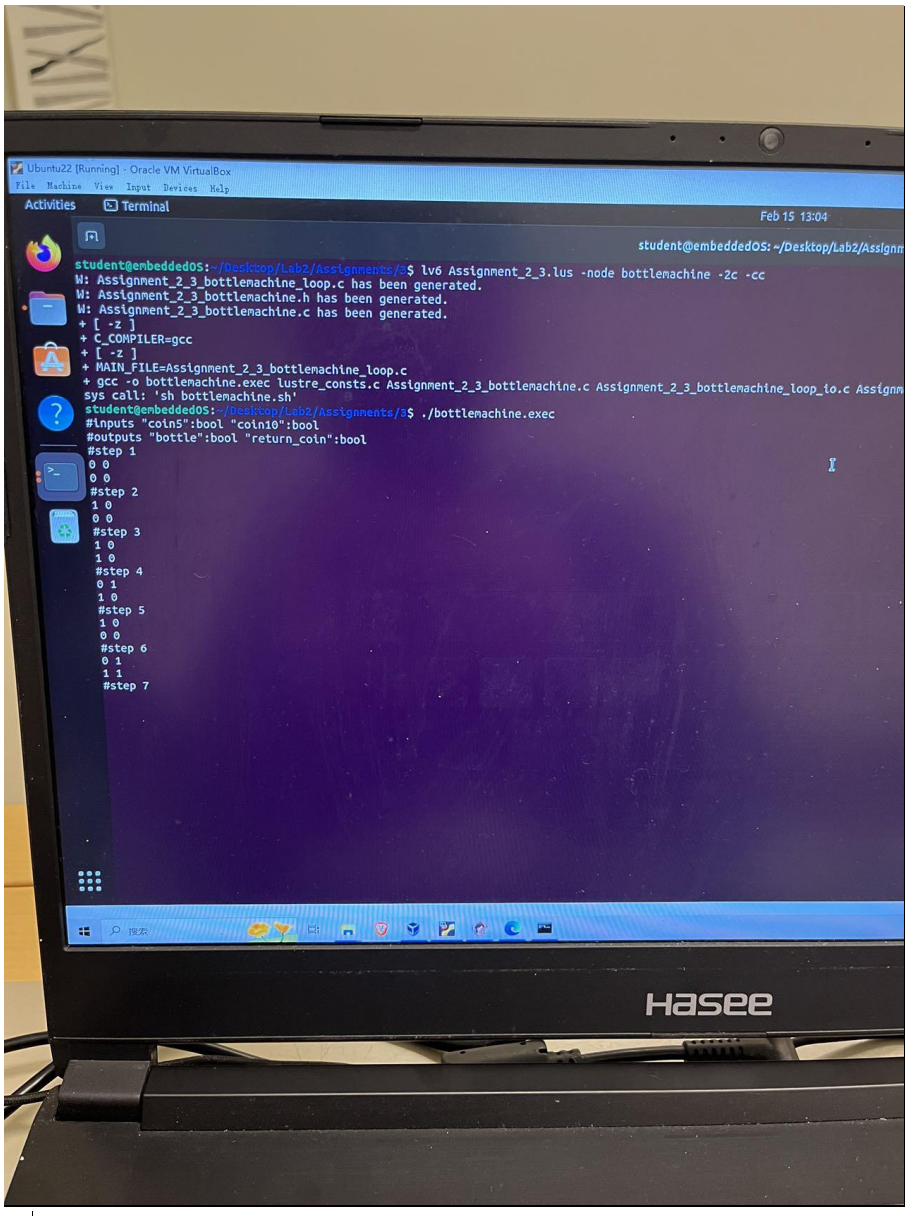
File Machine View Input Devices Help

Activities Text Editor

Open

```
node bottlemachine (coin5, coin10 : bool) returns (bottle, return_coin : bool);
var sum : int;
let
assert(not (coin5 and coin10));

sum = if pre(sum) = 5 and coin10 then 15
          else if pre(sum) = 5 and coin5 then 10
          else if coin5 then 5
          else 10;
bottle = false -> if sum > 9 then true
                  else false;
return_coin = false -> if sum > 10 then true
                  else false;
tel.
```


Ubuntu22 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Feb 15 13:04
student@embeddedOS: ~/Desktop/Lab2/Assignm
student@embeddedOS:~/Desktop/Lab2/Assignments/3$ lv6 Assignment_2_3.lus -node bottlemachine -2c -cc
W: Assignment_2_3_bottlemachine_loop.c has been generated.
W: Assignment_2_3_bottlemachine.h has been generated.
W: Assignment_2_3_bottlemachine.c has been generated.
+ [-z]
+ C_COMPILER=gcc
+ [-z]
+ MAIN_FILE=Assignment_2_3_bottlemachine_loop.c
+ gcc -o bottlemachine.exec lustre_consts.c Assignment_2_3_bottlemachine.c Assignment_2_3_bottlemachine_loop_io.c Assignm
sys call: 'sh bottlemachine.sh'
student@embeddedOS:~/Desktop/Lab2/Assignments/3$ ./bottlemachine.exec
#inputs "coin5":bool "coin10":bool
#outputs "bottle":bool "return_coin":bool
#step 1
0 0
0 0
#step 2
1 0
0 0
#step 3
1 0
1 0
#step 4
0 1
1 0
#step 5
1 0
0 0
#step 6
0 1
1 1
#step 7
Hasee

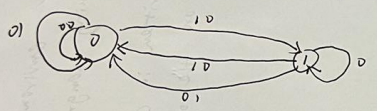
4.

a) The State Variable is s, the initial value is False.

b)

initial State	0	1 0 / 1	1 0 / 0	0 1 / 0	1 0 / 1	0 1 / 0	0 0	1 0 / 1	0 0 / 1	0 0 / 0	1 0 / 1	0 0 / 1

c)



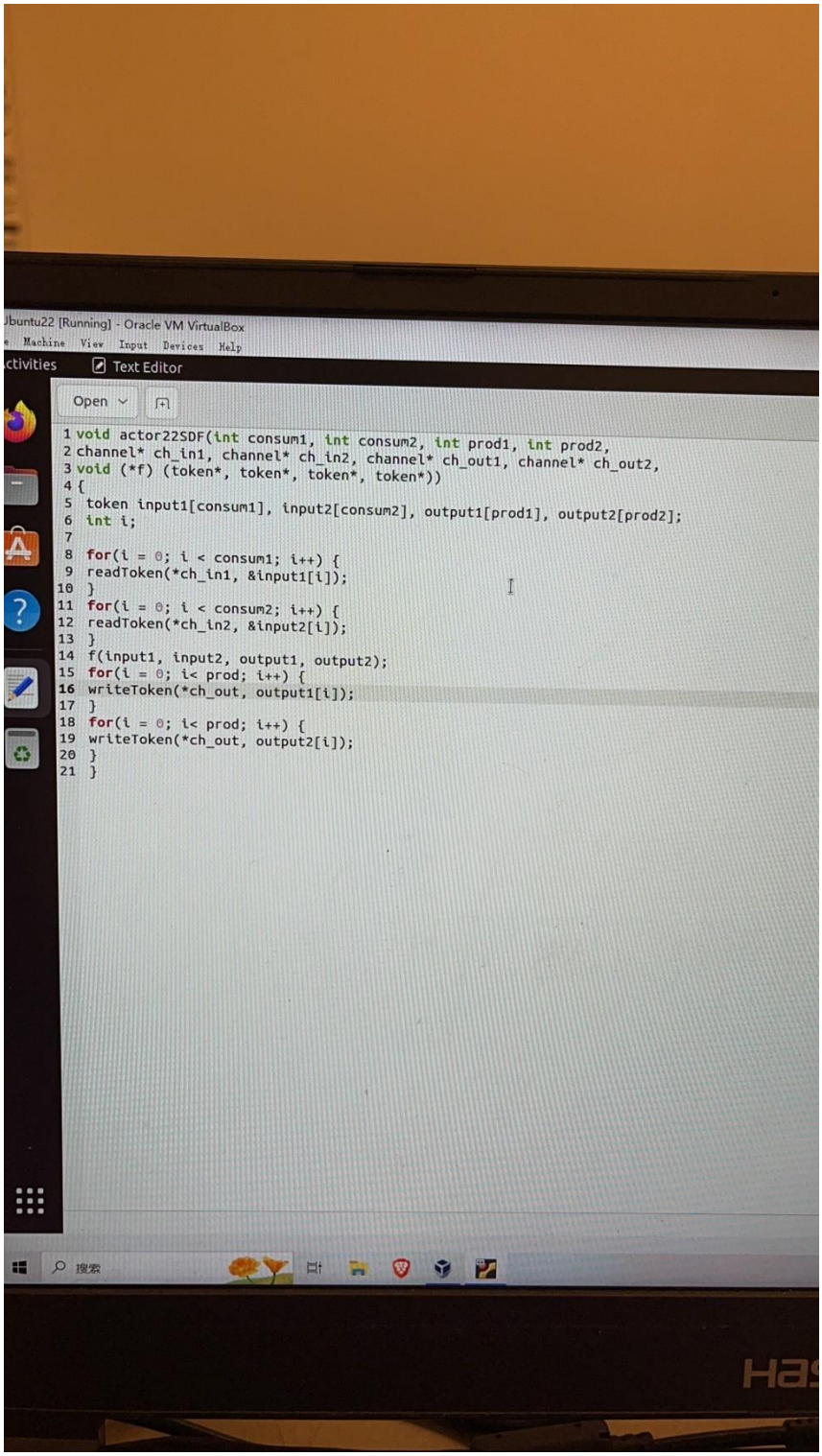
d) The function of program 1 is a state machine. It starts with state 0 and state will change to State 1 when the value of input a=1, b=0. In state 1, the input a=1, b=0 and a=0, b=1. In state 0, the input a=0, b=1 and a=0 b=0, it remains the state 0. The input a=0, b=0 will make the state unchanged. The output is related to the previous state. pre(s). and input x= b∧(a∨pre(s)), y=pre(s)∨b. It is like the machine with coin 5 and coin 10. A bottle needs 10 stok in that suitation.

```c
switch(current_state){
    case STATE0:{
        if(a==1&&b==0){
            x=0;
            y=0;
            current_state=STATE1;
            break;
        }
        if(a==0&&b==1){
            x=1;
            y=0;
            current_state=STATE0;
            break;
        }
        if(a==0&&b==0){
            x=0;
            y=0;
            current_state=STATE0;
            break;
        }
    }
    case STATE1:{
        if(a==1&&b==0){
            current_state=STATE0;
            x=1;
            y=0;
            break;
        }
        if(a==0&&b==1){
            current_state=STATE0;
            x=1;
            y=1;
            break;
        }
        if(a==0&&b==0){
            current_state=STATE1;
            x=0;
            y=0;
            break;
        }
    }
}
```

e)

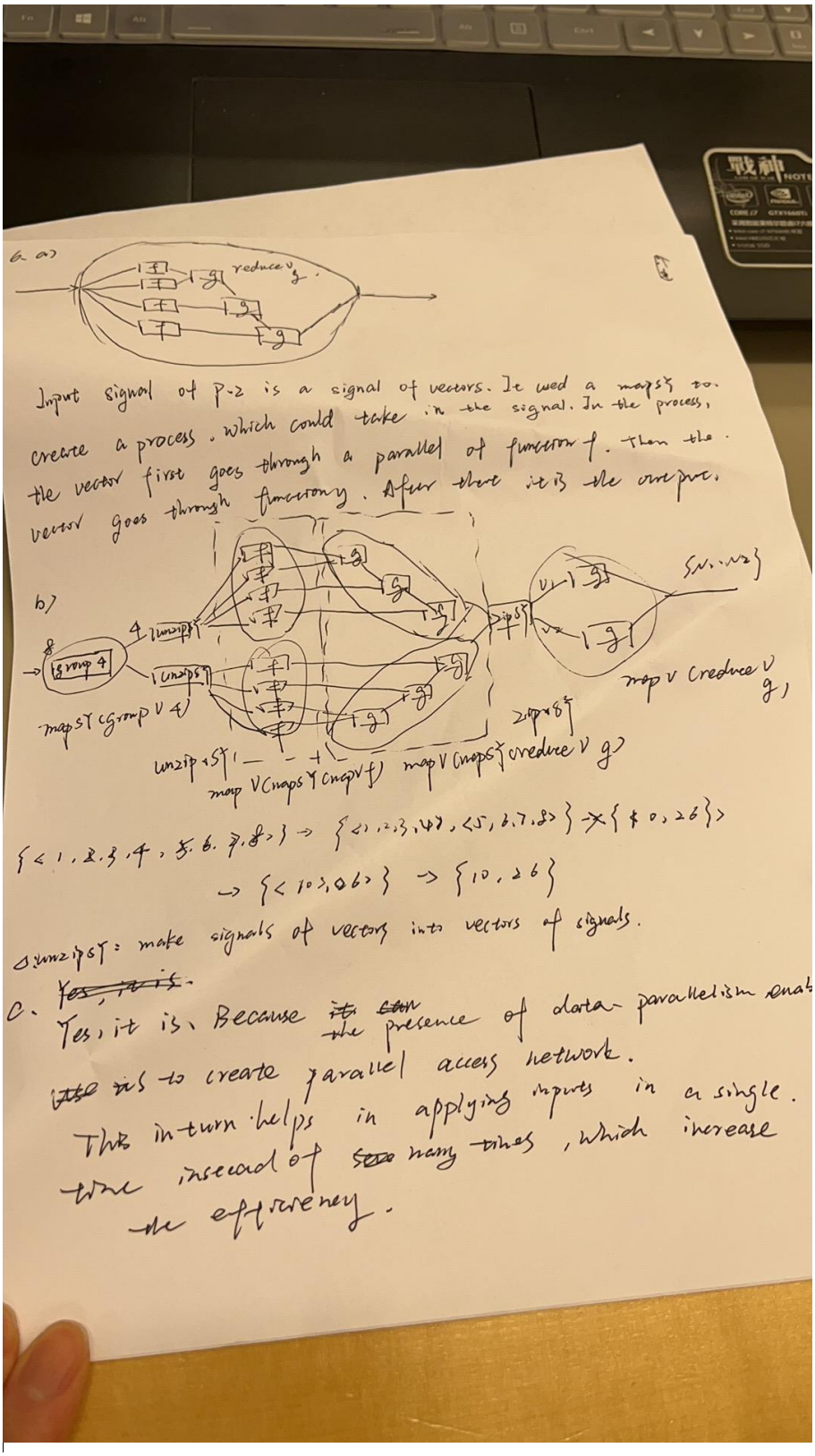
```c
while(1){
    scanf("%d",&a);
    scanf("%d",&b);
    if(a==1&&b==1){
        printf("error\n");
        break;
    }
    switch(current_state){
        case STATE0:{
            if (a ==1){
                current_state=1;
            }
            else
                current_state=0;
            x=b;
            y=0;
            break;
        }
        case STATE1:{
            if(a==0&&b==0){
                current_state=STATE1;
            }
            else
                current_state=STATE0;
            x=a||b;
            y=b;
            break;
        }
    }
    printf("x:%d y:%d\n",x,y);
```

5. a)



b) in file 5b.c

6.



7. a)

i)

Functionality: mapv . mapv it takes 2d vectors. It uses the 2nd mapv to all groups

at the same time and then uses the 1st mapv. Process the vectors and apply to the next stage of MapV in a parallel fashion to support parallelism.

parallel: fromvector and convert all the small groups at the same time.

ii)

Functionality: input: vectors(vectors) We use mapv to let reducev work on each vectors. After that, we transfer 2d into 1d. And then we use reducev for that 1d vectors. Finally, we get a specific value.

parallel: we use mapv to let all vectors use the reducev which acts sequentially.

b)

i)

functionality: we use mapv with groupv to split the vectors into small groups in which it has three elements representing rgb component. We use mapMatrix for two functions, convert and fromVector. In this part, convert helps in calculating the exact grayscale equivalent pixel from the respective rgb pixel.

Parallel: We use mapv to let groupv split each vectors into small groups which supports parallel processing.

ii)

functionality: pairwise column addition for vectors followed by pairwise row additional of vectors.

Parallel:

```
resize = mapMatrix (/ 4) . sumRows . sumCols
  where
    sumCols = mapV (mapV (reduceV (+)) . groupV 2)
    sumRows = mapV (reduceV (zipWithV (+))) . groupV 2
```

One mapv in mapmatrix is for groupv 2. So like we mentioned in the former part. Mapmatrix can do something parallel. Also in sumCols and sumRows, we use mapV to do something parallel.

8.