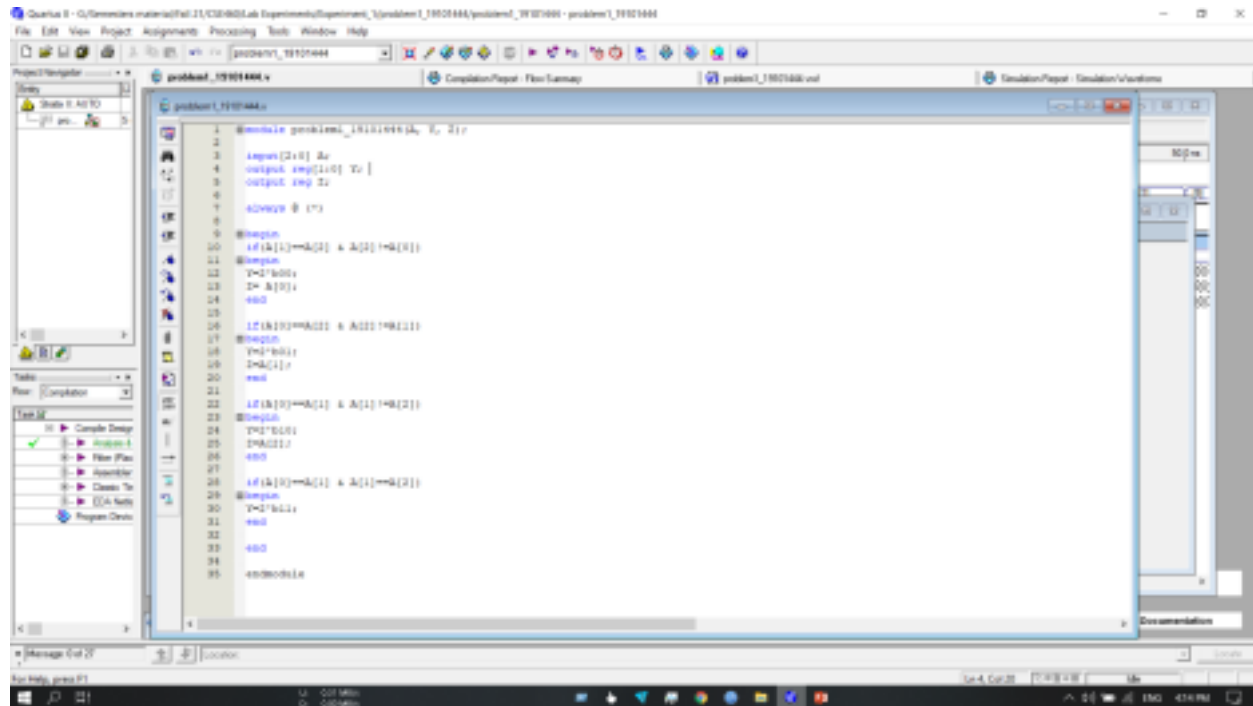


Ans of 1:

Code:



```
1 module problem1_19101444(A, B, Z);
2
3     input [2:0] A;
4     output [3:0] Y;
5     output [3:0] Z;
6
7     A[0:0] <= 1;
8
9     @begin
10        if(A[1]==A[0] & A[2]==A[3])
11            @begin
12                Y<=A[0];
13                Z<=A[0];
14            end
15        else if(A[0]==A[2] & A[2]==A[3])
16            @begin
17                Y<=A[2];
18                Z<=A[2];
19            end
20        else if(A[0]==A[1] & A[1]==A[2])
21            @begin
22                Y<=A[1];
23                Z<=A[2];
24            end
25        else if(A[0]==A[1] & A[1]==A[3])
26            @begin
27                Y<=A[1];
28                Z<=A[3];
29            end
30        end
31    end
32
33 endmodule
```

Output:



000, that matches with the condition when all the bits are equal output should be 3.

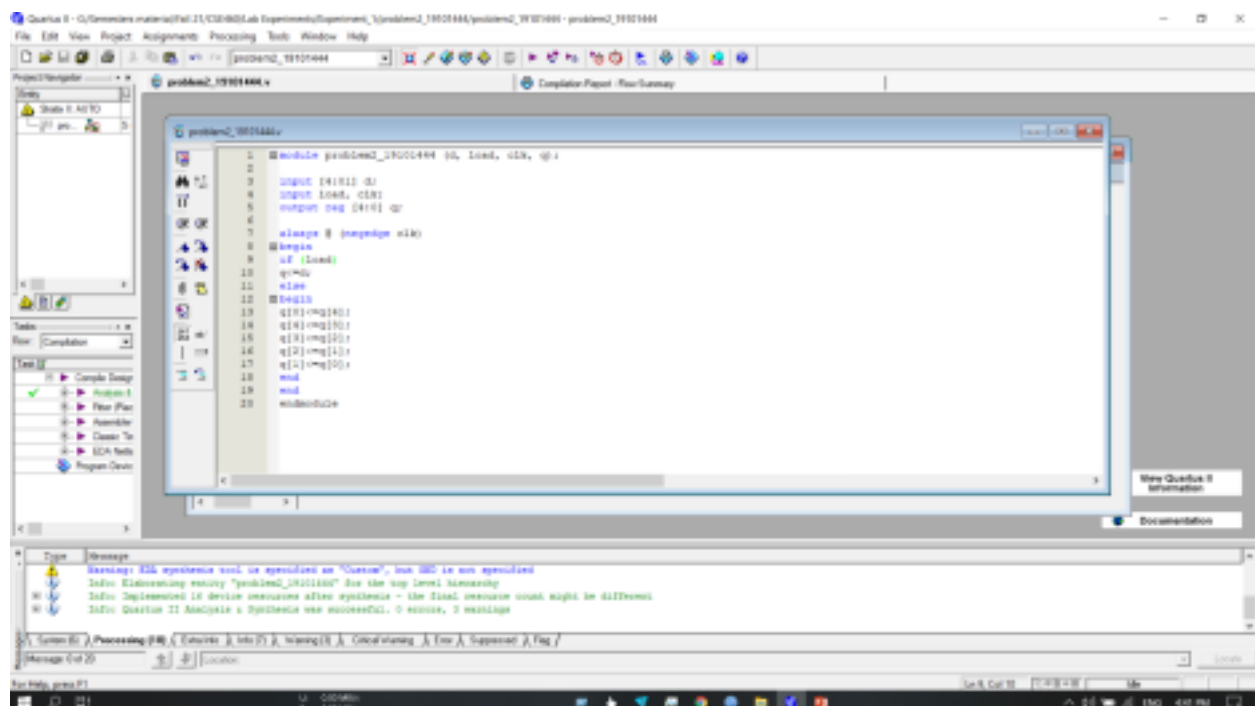
ii) when output Y is 2, which is the value of A is 4 or 3 which means in binary 100 or 011 where at 2nd position either is 0 and rest is 1 Or 2nd position is 1 and rest is 0 that matches with the condition when unique bit at 2 output should be 2 and also Z is 1.

ii) when output Y is 1, which is the value of A is 2 or 5 which means in binary 010 or 101 where at 1st position either is 0 and rest is 1 Or 2nd position is 1 and rest is 0 that matches with the condition when unique bit at 1, the output should be 1 and also Z is 1.

iv) when output Y is 0, which is the value of A is 6 or 1 which means in binary 110 or 001 where at 0 no position either is 0 and rest is 1 or 1 and rest is 0 that matches with the condition when unique bit at 0 output should be 0 and also Z is 0.

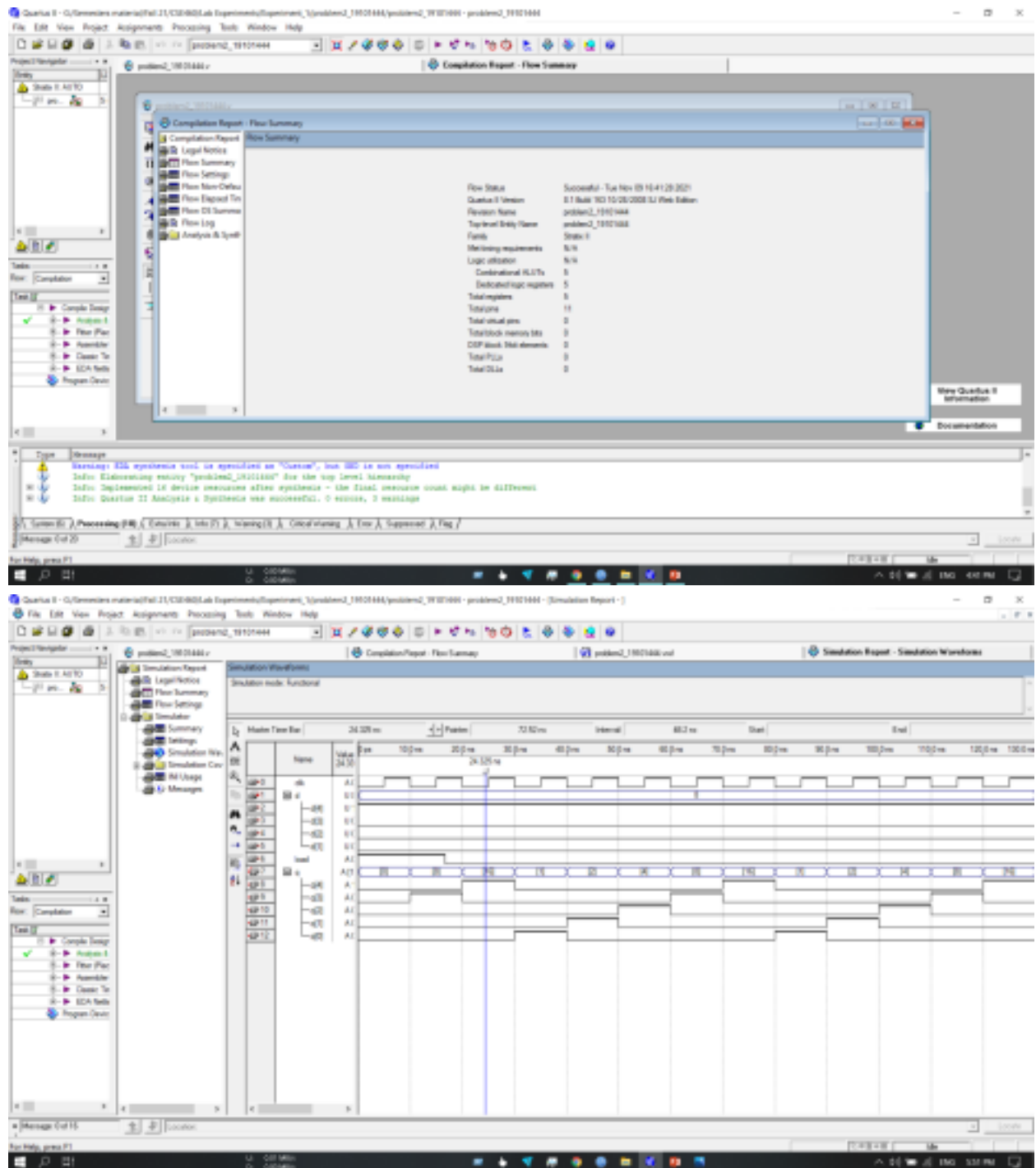
Ans of 2:

Code:



```
1 module problem2_19101444 (a, b, c, y, z);
2
3     input [1:0] a;
4     input [1:0] b;
5     output reg [1:0] y;
6
7     always @ (a,b,c)
8     begin
9         if (a==b)
10            y<=a;
11        else
12            y<=b;
13        if (a==b)
14            y<=a;
15        if (b==c)
16            y<=b;
17        if (a==c)
18            y<=a;
19    end
20 endmodule
```

Output:



Here, from the simulation waveform, we can see for taking d as 8(decimal) the 5-bit left shifter is shifting in q when there is a negative edge in load it finds.

I have taken clock as 10 ns

At first, it converts 8 to 16 decimal (which is 01000 to 10000 in binary) in 20 ns, as it finds load

negative first time, then the load is negative all the time, so in 30 ns it changes to 1 (b 00001) and 2 (b 00010), 4 (b 00100), and again 8 (b 01000) in 40, 50 and 60 ns. So it starts from 10 ns and again comes back to same number at 60 ns. So  $60 - 10 \text{ ns} = 50 \text{ ns}$

So, the total number of clock cycles required to get one repetition =  $50 / 10 = 5$

So for a 5-bit register, it takes 5 clock cycles,

so for a n bit-register, it will take n clock cycles.