Solver: Andre Hartanto

Email Address: C150007@e.ntu.edu.sg

1. (a)

(i) First we know that each digit in octal occupy 3 bits, so we can write it as:
6 = 110
7 = 111
2 = 010
1 = 001
5 = 101
So we can write it ass 110 111 010.001 101 (2)
Then to convert it to hexadecimal, we can group it by 4 bits each starting from the binary point, so it becomes 0001 1011 1010.0011 0100 (2)
We can add additional zeros in front of the decimal part and in the back of the fractional part.
So it becomes 1BA.34(16)

(ii) To convert it to decimal, we have to multiply each digit by 16^n where n is 0, 1, 2.. for decimal part, and -1,-2,-3 for the fractional part, starting from the radix point. So:
$C1.FD = 12 * 16 + 1 * 1 + 15 * 16^{-1} + 13 * 16^{-2}$
$= 193.98828125\ (10)$
So the answer is 193.988(10)

(iii) To convert decimal to binary, we have to divide the decimal part by 2, and multiply the fractional part by 2. So it becomes

6 / 2 = 3 R 0	0.48 * 2 = 0.96
3 / 2 = 1 R 1	0.96 * 2 = 1.92
1 / 2 = 0 R 1	0.92 * 2 = 1.84
	0.84 * 2 = 1.68
	0.68 * 2 = 1.36

So the answer is 110.01111(2).

(b) We know that the maximum number of visitors per day is 3000. So, the maximum number of visitors in one year is:

365 * 3000 = 1095000

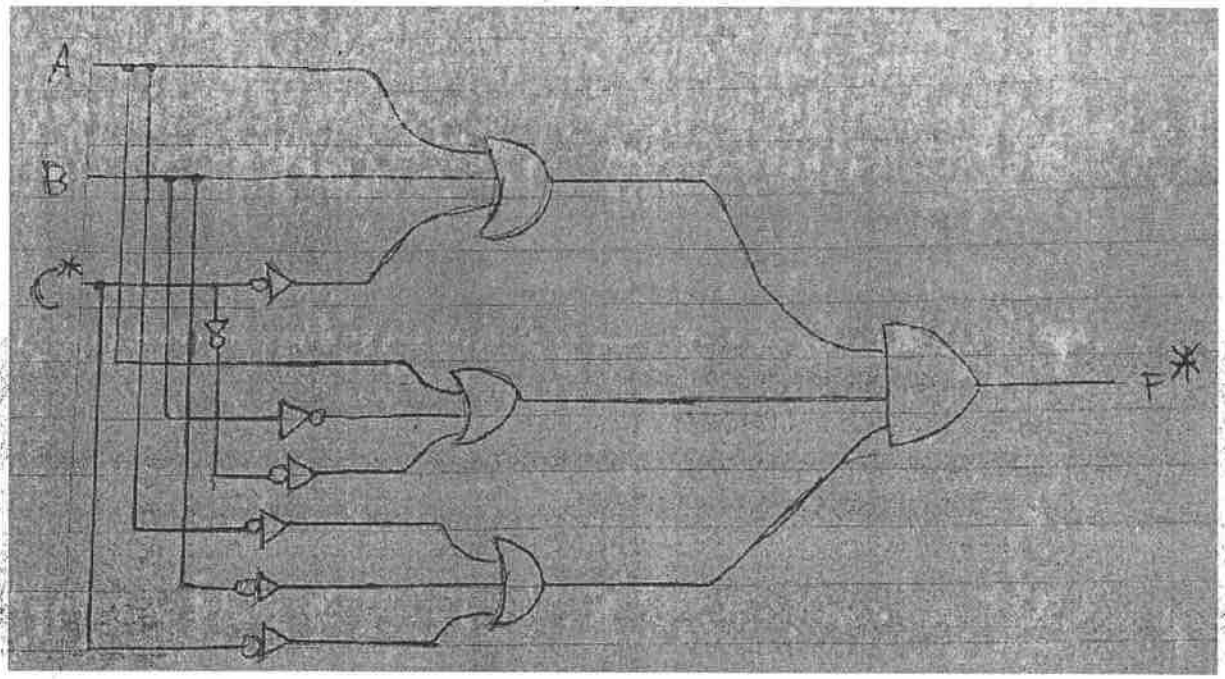
To represent 1095000 in bits:

$^2\log 1095000 = 20.0625 = 20$

But we know that we need to add 1 to the answer, because 1095000 is more than $2^{20} - 1$, so the answer is 21.

(c) F(a,b,c,d) = (a'c' + d') (bd + c')' (a'd + (b'c)')' + (c + d')'
= (a'c' + d') ((bd)'c) (a'd + b + c')' + c'd
= (a'c' + d') ((b' + d')c) ((a'd)'b'c) + c'd
= (a'c' + d') (b'c + cd') ((a + d')b'c) + c'd
= (a'c' + d') (b'c + cd') (ab'c + b'cd') + c'd
= (b'cd + cd') (ab'c + b'cd') + c'd
= (ab'cd +ab'cd') + c'd
= ab'c(d + d') + c'd
= ab'c + c'd X
b'cd' + c'd

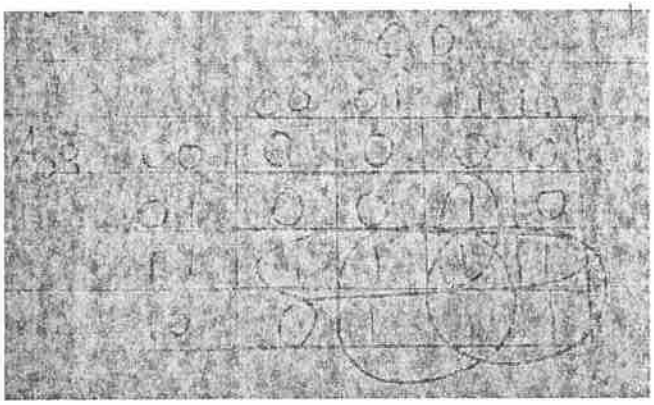
(d)



2. (a)
(i)

A	B	C	D	SU
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
v1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(ii)



SU = A.B + A.C + A.D + B.C.D

(b)

(i) First we find the bits representation for the magnitude first, so :

72 / 2 = 36 R 0
36 / 2 = 18 R 0
18 / 2 = 9 R 0
9 / 2 = 4 R 1
4 / 2 = 2 R 0
2 / 2 = 1 R 0
1 / 2 = 0 R 1

So 72 is equal to 1001000, but we know that in 2's complement representation, we must flip all the digits and add 1 to get the negative value, so
1001000 = 0110111 + 1 = 0111000

We add digit 1 to the front to indicate that this is negative value, so the answer is 10111000

67 / 2 = 33 R 1
33 / 2 = 16 R 1
16 / 2 = 8 R 0
8 / 2 = 4 R 0
4 / 2 = 2 R 0
2 / 2 = 1 R 0
1 / 2 = 0 R 1

So 67 is equal to 1000011, and because it's positive, the 2's complement representation doesn't change. We add digit 0 to get the positive value, so the answer is 01000011.

(ii) First we represent -72 and +67 in 8-bit 2's complement format and we can use the answers in part a. So, 72 is 1001000 and 67 is 1000011. Let's assume that the input is X and have 8 digits. We can divide this into 4 cases:

-72 to -65 = X[7].X[6]'.X[5].X[4].X[3]
-64 to -1 = X[7].X[6]
0 to 63 = X[7]'.X[6]'
64 to 67 = X[7]'.X[6].X[5]'.X[4]'.X[3]'.X[2]'

We add all the cases together and we get:

X[7].X[6]'.X[5].X[4].X[3] + X[7].X[6] + X[7]'.X[6]' +
X[7]'.X[6].X[5]'.X[4]'.X[3]'.X[2]'

So the answer is X[7].X[6]'.X[5].X[4].X[3] + X[7].X[6] + X[7]'.X[6]' +
X[7]'.X[6].X[5]'.X[4]'.X[3]'.X[2]'

(c)

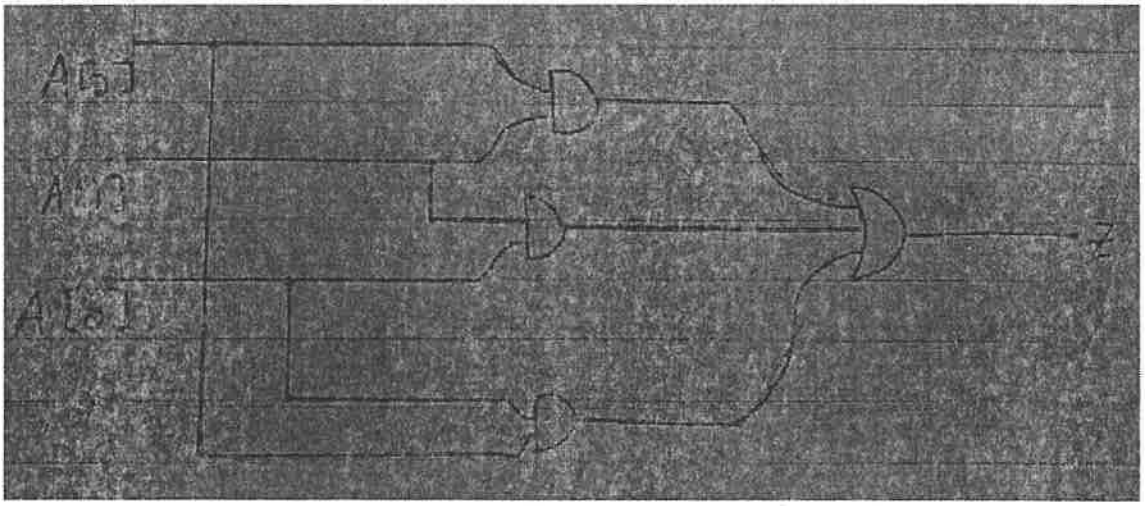
EN	A	X
0	0	Hi
0	1	Hi
1	0	0
1	1	1

This circuit is a high – impedance buffer that only works if the EN is on / 1. If not, then the output is undetermined. If activated, then it transfer A to X.

3. (a)
(i)

		A[1],A[0]			
		00	**01**	**11**	**10**
A[2]	**0**	0	0	1	0
	1	0	1	1	1

Z = A[2].A[0] + A[1].A[0] + A[2].A[1]
This is the SOP form.
(ii)



(iii)

```
module majority(input [2:0] A, output Z);
      wire int1, int2, int3;

      or (Z,int1,int2,int3);
      and (int1,A[2],A[0]);
      and (int2,A[1],A[0]);
      and (int3,A[2],A[1]);

endmodule
```

(b)

(i) We know that the output Z will be 1 if the majority of the inputs are 1's, and we want to implement it using multiplexer. A multiplexer is a combinational circuit that selects one output from several inputs. The select input will determine which input should be connected to the output. Because the only input is A[2:0] we know that it must be the select input. We can choose A[2] and A[1], A[1] and A[0]. It makes no difference. Then there will be 4 cases:

00: Because two of the digits are 0, so the output must be 0 because the maximum number of digit 1 is 1.
01: The output depends on the last digit because the number of digit 1 from the two select input is 1.
10: Same as 01.
11: The output must be 1, because two of the digits are 1.

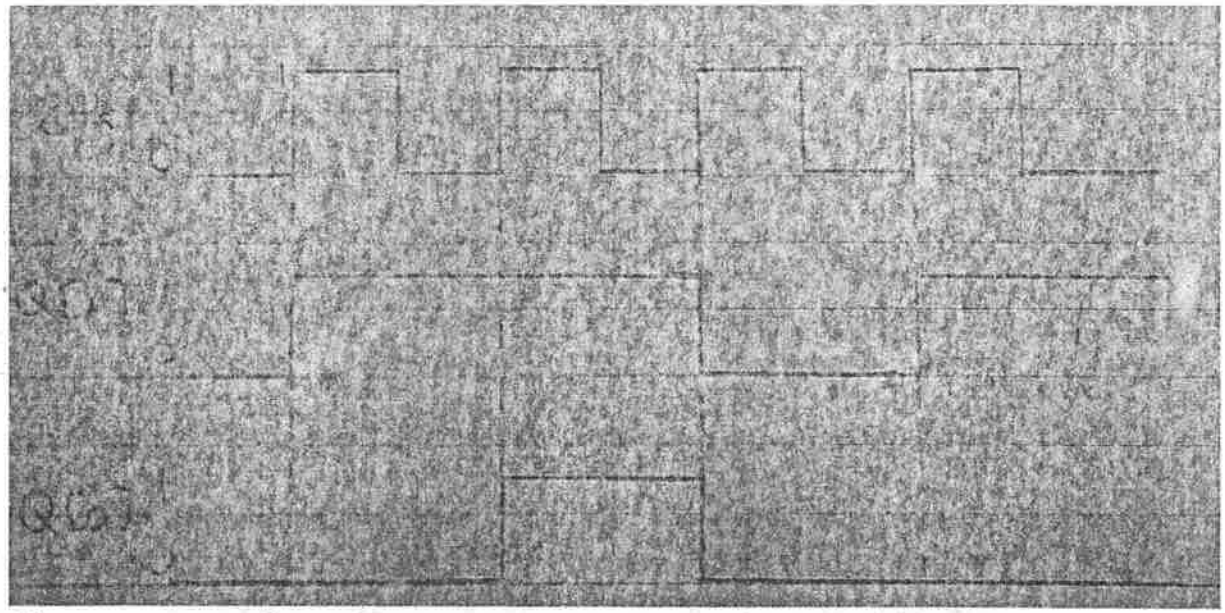
So the "?" for the select input can be A[2:1] or A[1:0] and the "?" for the input is A[0] or A[2].

(ii)

```
module major(input [2:0] A, output reg Z);

      always @ *
      begin
          case(A[1:0])
            2'b00: Z = 1'b0;
            2'b01: Z = A[2];
            2'b10: Z = A[2];
            2'b11: Z = 1'b1;
          endcase
       end
endmodule
```

4. (a)
(i)



(ii)

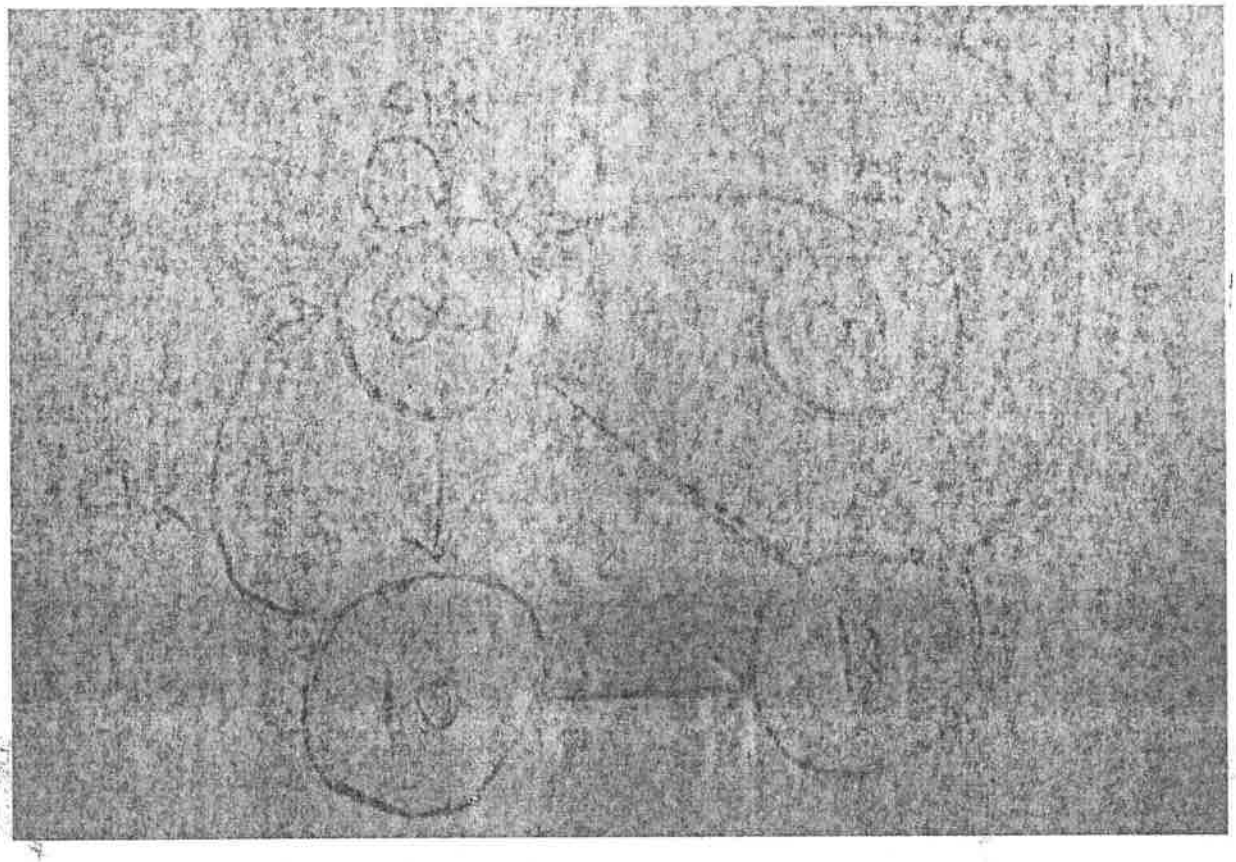
```
module state (input Clk, Rst, ouput reg [1:0] Q);
wire w1;

always@(posedge Clk)
begin
      if(rst)
          Q <= 2'b00;
      else
          Q[0] <= w1;
          Q[1] <= w2;
end

assign w1 = ~Q[0] & Q[1];
assign w2 = ~Q[0];

endmodule
```

(iii)



(b)