PAGE NO.:
DATE: / /

DEX Q.B.

Q.1. a) A physical quantity, which contains some information and which is a function of one or more independent variables.
b) A digital signal is defined as the signal which has only a finite number of distinct values.
c) An analog signal is defined as the signal having continuous values.
d) The smallest "unit" of data is defined as a single bit.

Q.2) Convert

a) $(01101)_2 = (?)_{10}$

$(01101) = 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 0 + 8 + 4 + 0 + 1$

$= (13)_{10}$

b) $(123)_8 = (?)_{10}$

$(123)_8 = 1\times8^2 + 2\times8^1 + 3\times8^0$

$= 64 + 16 + 3$

$= (83)_{10}$

c) $(FC)_{16} = (?)_{10}$

$\because F = 15$, $C = 12$

$\therefore (FC)_{16} = 15\times16^1 + 12\times16^0$

$= 240 + 12$

$= (252)_{10}$

Q.3) Convert

a) $(12.59)_{10} = (?)_2$

2	12	R
2	6	0
2	3	0
2	1	1
	0	1

$(12)_{10} = (1100)_2$

$0.59 \times 2 = 1.18 \rightarrow 1$
$0.18 \times 2 = 0.36 \rightarrow 0$
$0.36 \times 2 = 0.72 \rightarrow 0$

$(0.59)_{10} = (0.100)_2$

$\therefore (12.59) = (1100.100)_2$

b) $(345.37)_{10} = (?)_8$

8	345	R
8	43	1
8	5	3
	0	5

$(345)_{10} = (531)_8$

$0.37 \times 8 = 2.96 \rightarrow 2$
$0.96 \times 8 = 7.68 \rightarrow 7$
$0.68 \times 8 = 5.44 \rightarrow 5$
$0.44 \times 8 = 3.52 \rightarrow 3$

$(0.37)_{10} = (0.2753)_8$

$\therefore (345.37)_{10} = (531.2753)_8$

) $(45.76)_{10} = (?)_{16}$

16	45	R
	2	D
	0	2

$0.76 \times 16 = 12.16 \rightarrow C$
$0.16 \times 16 = 2.56 \rightarrow 2$
$0.56 \times 16 = 8.96 \rightarrow 8$
$0.96 \times 16 = 15.36 \rightarrow F$

$(45)_{10} = (2D)_{16}$

$(0.76)_{10} = (0.C28F)_{16}$

$\therefore (45.76)_{10} = (2D.C28F)_{16}$

Q.4) Convert

a) $(12.59)_{10} = (?)_{16}$

$(12)_{10} = (C)_{16}$

$0.59 \times 16 = 9.44 \rightarrow 9$
$0.44 \times 16 = 7.04 \rightarrow 7$
$0.04 \times 16 = 0.64 \rightarrow 0$
$0.64 \times 16 = 10.24 \rightarrow A$

$\therefore (0.59)_{10} = (0.970A)_{16}$

$\therefore (12.59)_{10} = (C.970A)_{16}$

b) $(345.37)_8 = (?)_2$

$(3)_8 = (011)_2$, $(4)_8 = (100)_2$, $(5)_8 = (101)_2$

$\therefore (345)_8 = (011100101)_2$

$(0.3)_8 = (0.011)_2$, $(0.7)_8 = (0.111)_2$

$\therefore (0.37)_8 = (0.011111)_2$

$\therefore (345.37)_8 = (011100101.011111)_2$

c) $(45.76)_{16} = (?)_8$

$(4)_{16} = (0100)_2$, $(5)_{16} = (0101)_2$
$\therefore (45)_{16} = (01000101)_2$

$(0.7)_{16} = (0111)_2$, $(0.6) = (0110)_2$
$(0.76)_{16} = (0.01110110)_2$

$(45.76)_{16} = (01000101.01110110)_2$

Divide both Integer and decimal part into 3 parts,

$I = 010\ 001\ 010 \rightarrow 212$

$D = 011\ 101\ 100 \rightarrow 354$

$\therefore (45.76)_{10} = (212.354)_8$

Q.5) Addition in Binary.

a) $(1100)_2 + (1011)_2 = (?)_2$

$$\begin{array}{r} 1100 \\ +1011 \\ \hline 10111 \end{array}$$

$\therefore (1100)_2 + (1011)_2 = (10111)_2$

b) $(12)_{10} + (34)_{10} = (?)_{10}$

$(12)_{10} = (1100)_2$, $(34)_{10} = (100010)_2$

$$\begin{array}{rl} 100010 & \rightarrow 34 \\ + \quad 1100 & \rightarrow 12 \\ \hline 101110 & - 46 \end{array}$$

$(12)_{10} + (34)_{10} = (46)_{10}$

Q.6) Subtraction in binary

a) $(1100)_2 + (1011)_2 = (?)_2$

```
  1 1
  1100   - 12
  1011   - 11
 ------
  0001   - 1
```

$(1100)_2 - (1011)_2 = (0001)_2$

b) $(52)_{10} - (32)_{10} = (?)_{10}$

$(52)_{10} = (110100)_2$, $(32)_{10} = (100000)_2$

```
  110100  - 52
- 100000  - 32
 ------
  010100  - 20
```

$(52)_{10} - (32)_{10} = (20)_{10}$

Q.7) Subtraction using 1's complement

a) $(1100)_2 - (1011)_2 = (?)_2$

1's complement of 1011 → 0100

```
  1
  1100
+ 0100
 -----
 10000
+    1
 -----
  0001
```

$\therefore (1100)_2 - (1011)_2 = (0001)_2$

b) $(52)_{10} - (32)_{10} = (?)_{10}$

$(52)_{10} = (110100)_2$, $(32)_{10} = (0100000)$

1's complement of 100000 ⟶ 011111

```
  1 1 0 1 0 0   - 52
+ 0 1 1 1 1 1   - 32
  1 0 1 0 0 1 1
+ └─────────→ 1
  ─────────────
  0 1 0 1 0 0   - 20
```

$(52)_{10} - (32)_{10} = (20)_{10}$

Q8) Subtraction using 2's complement.

a) $(1100)_2 - (1011)_2 = (?)_2$

1's complement of 1011 ⟶ 0100

Add 1,

```
 0100
+0001
 ----
 0101  - 2's complement.
```

Now,

```
  1 1 0 0
+ 0 1 0 1
  -------
1 0 0 0 1
↓
Discard
```

$\therefore (1100)_2 - (1011)_2 = (0001)_2$

b) $(52)_{10} - (32)_{10} = (?)_{10}$

$(52)_{10} = (110100)_2$, $(32)_{10} = (100000)_{10}$

1's complement of $100000 \rightarrow 011111$

2's complement is,

$$\begin{array}{r} 011111 \\ +\ 000001 \\ \hline 100000 \end{array}$$ — 2's complement

Now,

$$\begin{array}{r} 110100 \\ +\ 100000 \\ \hline 1010100 \end{array}$$

↓ Discard

$\therefore (110100)_{10} - (100000)_{10} = (010100)_{10}$

$\therefore (52)_{10} - (32)_{10} = (20)_{10}$

Q.13) a) Any binary operation which satisfies the following expression is referred to as commutative operation:

1. $A.B = B.A$ 2. $A+B = B+A$

b) This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

i.e. $(A.B).C = A.(B.C)$

and $(A+B)+C = A+(B+C)$

Similarly we can verify the other statement

i.e. $A+(B+C) = (A+B)+C$

c) The distributive law states that,

$A.(B+C) = AB+AC$

d) This law uses the AND operation therefore they are called as "AND" laws.

The AND laws are as follows:

1. $A.0 = 0$ i.e. ANDing with a 0, always results in 0 output.
2. $A.1 = A$ i.e. ANDing of A with a 1, results in A.
3. $A.A = A$ i.e. ANDing of an input with itself will produce the same output.
4. $A.\bar{A} = 0$ i.e. ANDing of an input with its complement results in a 0 output

PAGE NO:
DATE: / /

e) These laws use the OR operation. Hence they are called as OR laws. The OR laws are as follows:

1. $A + 0 = A$ i.e. ORing an input with 0, results in the output equal to input.
2. $A + 1 = 1$ i.e. ORing an input with 1, always results in a high output.
3. $A + A = A$ i.e. ORing an input with itself, results in the same output.
4. $A + \overline{A} = 1$ i.e. ORing an input and its complement always results in a high output.

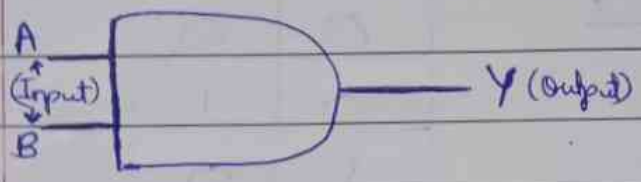
f) This law uses the "NOT" operation. The inversion law states that double inversion of a variable results in the original variable itself i.e., $\overline{\overline{A}} = A$

PAGE NO.:
DATE: / /

Q.14)

Logic Symbol

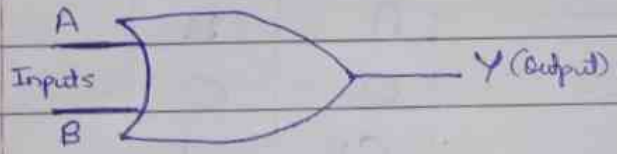
a)



Truth Table

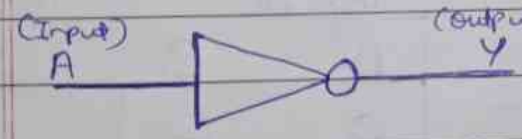
Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

b)



Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

c)



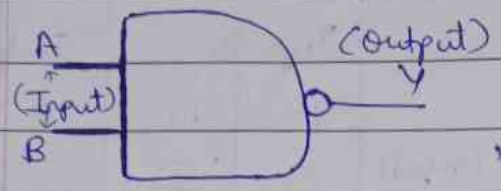
Input	Output
A	Y
0	1
1	0

PAGE NO.:
DATE: / /

Logic Symbol

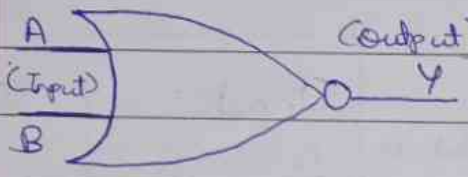
Truth Table

d)



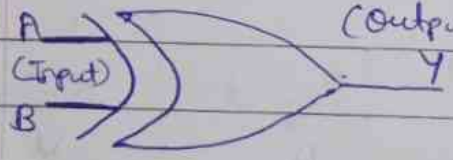
Inputs		Outputs
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

e)



Inputs		Outputs
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

f)



Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

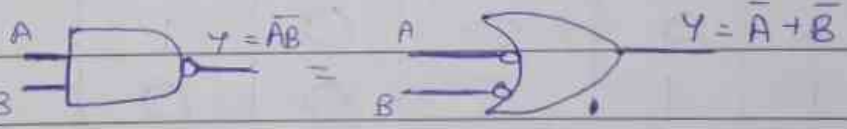
PAGE NO.:
DATE: / /

Q.15) The two theorems suggested by De-Morgan and which are extremely useful in Boolean algebra are as follows:

Theorem 1: $\overline{AB} = \bar{A} + \bar{B}$: (NAND = Bubbled OR)

- This theorem states that the complement of a product is equal to addition of the complements.
- The left hand side (LHS) of this theorem represents a NAND gate with inputs A & B. whereas the RHS of the theorem represents ~~an OR gate~~ a bubbled OR gate with inputs A & B. Thus we can state De-Morgan's first theorem as,

NAND = Bubbled OR



This theorem can be proved by writing a truth table

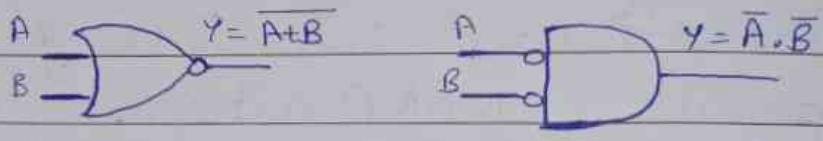
A	B	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A} + \bar{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

LHS → $\overline{AB} = \bar{A} + \bar{B}$ ← RHS

Theorem 2: $\overline{A+B} = \bar{A} \cdot \bar{B}$: NOR = Bubbled AND

- This theorem states that the complement of a sum is equal to product of complements
- The LHS of this theorem represents NOR gate with inputs A and B whereas the RHS represents a bubbled AND with inputs A and B. Thus we can state De-Morgan's 2nd theorem as

NOR = Bubbled AND



This theorem can be proved by writing a truth table

A	B	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A} \cdot \bar{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

LHS → $\overline{A+B} = \bar{A} \cdot \bar{B}$ RHS

PAGE NO.:
DATE: / /

Q.16)

a) $(A+B)(A+C) = A+BC$

$LHS = (A+B)(A+C)$

$= AA + AC + AB + BC$

$= A + AC + AB + BC \quad \ldots (\because AA = A)$

$= A(1+C) + AB + BC$

$= A + AB + BC \quad \ldots (\because 1+C = 1)$

$= A(1+B) + BC$

$LHS = A + BC \quad \ldots (\because 1+B = 1)$

$LHS = RHS$

b) $A + A'B + AB = A+B$

$LHS = A + A'B + AB$

$= A(1+B) + A'B$

$= A + A'B \quad \ldots (\because 1+B = 1)$

$= (A+A')(A+B)$

$= 1(A+B) \quad \ldots (\because A + A' = 1)$

$LHS = A+B$

$LHS = RHS$

Q.17) $F = \bar{A}B + A\bar{B}$

$= (\bar{A}B + A\bar{B})$

Q.17) $F = \bar{A}B + A\bar{B}$

$= \overline{\bar{A}B + A\bar{B}}$ ... ($\because \overline{A+B} = \bar{A}.\bar{B}$) - DeMorgan

$= (\overline{\bar{A}B}).(\overline{A\bar{B}})$

$= (\overline{\bar{A}} + \overline{B}).(\bar{A} + \overline{\bar{B}})$ ... ($\because \overline{A.B} = \bar{A} + \bar{B}$) - DeMorgan

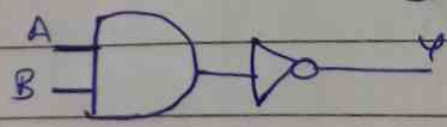
$= (A + \bar{B}).(\bar{A} + B)$ ... ($\because \overline{\bar{A}} = A$; $\overline{\bar{B}} = B$)

$= A.\bar{A} + A.B + \bar{A}.\bar{B} + \bar{B}.B$

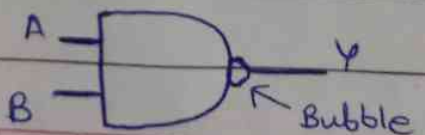
$= 0 + A.B + \bar{A}.\bar{B} + 0$ ... ($\because A.\bar{A} = 0$; $B.\bar{B} = 0$)

$= AB + \bar{A}.\bar{B}$

Q.18) The term NAND can be split as NOT-AND which means that the NAND operation is a combination of an AND gate and a NOT gate i.e. inverter. Thus a NAND gate is equivalent to an AND gate followed by an inverter.



The logical symbol of a NAND gate is shown below, where the bubble (○) represents inversion on the output side



The truth table of a two input NAND gate shows that the output will be low (0) if both the inputs are high (1). For all other inputs the output will be high (1).

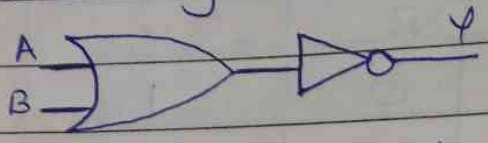
Inputs		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

The boolean expression for the NAND gate is $Y = \overline{A.B}$

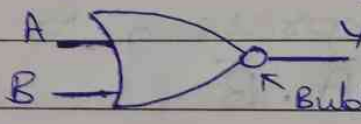
Here, A.B represents an AND gate while the bar represents an inversion (NOT gate).

Therefore, a NAND gate is called as "Universal Gate" because we can construct AND, OR and NOT gates using only NAND gates.

Q.19) The word NOR can be split as NOT-OR which means that the NOR operation is a combination of an OR gate and a NOT gate, i.e. inverter. Thus a NOR gate is equivalent to an OR gate followed by an inverter



The logical symbol of two input NOR gate is shown below where the bubble (o) on the output side represents inversion.



The truth table of a two input NOR gate shows that the output will be high (1) if both the inputs are low (0). For all other inputs the output will be low (0).

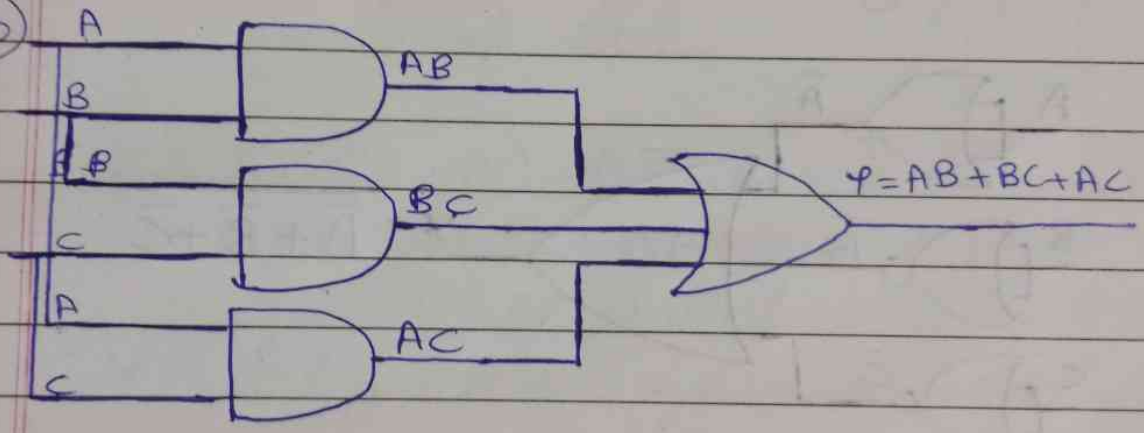
Inputs		Outputs
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

The boolean expression for the NOR gate is $Y = \overline{A+B}$

Here, A+B represents an OR gate while the bar represents an inversion (NOT gate).

Therefore, a NOR gate is called as "Universal Gate" because we can construct AND, OR and NOT gates using only NAND gates.

(Q.20)



PAGE NO.:
DATE: / /

Q.2) The given equation is,

$Y = (A\bar{B}C + \bar{B}\bar{C})C$

$= A\bar{B}CC + \bar{B}\bar{C}C$

$= A\bar{B}C + 0 \quad \ldots (\because \bar{C}C = 0 \ \& \ C.C = C)$

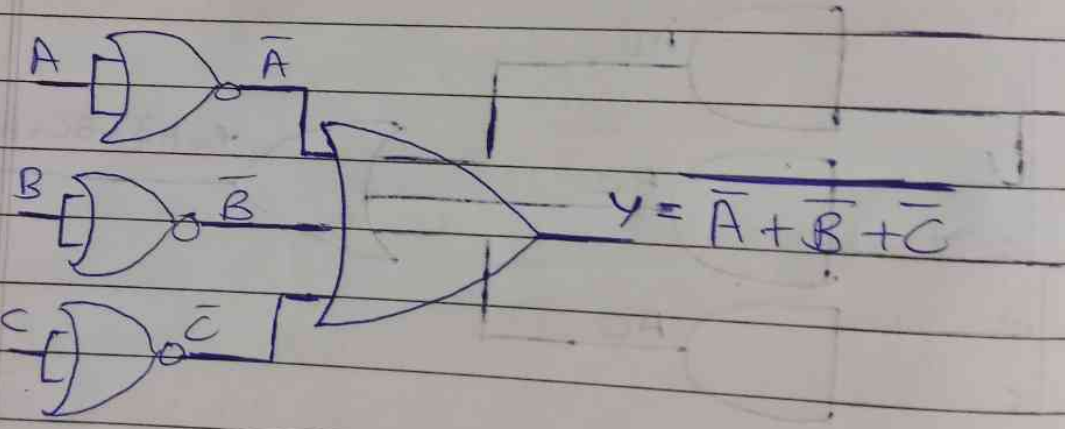
$= A\bar{B}C$

$Y = \overline{A\bar{B}C}$

Using De-Morgan's first theorem we get

$Y = \bar{A} + \bar{\bar{B}} + \bar{C}$

This expression can be realized using NOR gates as shown below:



PAGE NO.:
DATE: / /

Q.22) Step 1: Find the missing literal for each term

$$Y = \underbrace{AB}_{C} + \underbrace{A\bar{C}}_{B} + \underbrace{BC}_{A}$$

Missing literal → C B A

Step 2: AND each term with (Missing literal + Its Complement)

$$Y = AB \cdot (C+\bar{C}) + A\bar{C}(B+\bar{B}) + BC(A+\bar{A})$$

ANDING ↑ (Missing literal + Its Complement)

Step 3: Simplify the expression to get the canonical SOP.

$$Y = ABC(C+\bar{C}) + A\bar{C}(B+\bar{B}) + BC(A+\bar{A})$$
$$= ABC + AB\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC + \bar{A}BC$$
$$= (ABC + ABC) + (AB\bar{C} + AB\bar{C}) + \bar{A}BC + A\bar{B}\bar{C}$$
$$\therefore Y = ABC + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC \leftarrow \text{canonical SOP form}$$

Each term contains all the literals

Q23) Step 1: Find the missing literal for each term.

$Y = (A+B)(A+C)(B+\bar{C})$

Missing Literals → C, B, A

Step 2: OR each term with (Missing literal & Its complement)

$Y = (A+B+C\bar{C}).(A+C+B\bar{B}).(B+\bar{C}+A\bar{A})$

This term is ORed with missing literal and its complement.

missing literal ANDed with its complement.

Step 3: Simplify the expression to get canonical POS.

$Y = (A+B+C\bar{C})(A+C+B\bar{B})(B+\bar{C}+A\bar{A})$

$= (A+B+C)(A+B+\bar{C})(A+B+C)(A+\bar{B}+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+\bar{C})$

Each term contains all the literals

Canonical POS form

Q.25)

Minterm: Each individual term in the canonical SOP form is called as minterm and is denoted by "m".

Ex

Canonical SOP $Y = A\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C}$

Each individual term is called minterm.

Maxterm: Each individual term in the canonical POS form is called as maxterm & is denoted by "M".

Ex:

Canonical POS $Y = (A+B) \cdot (A+\bar{B})$

Each individual term is called Maxterm

Minterm and Maxterm for two variables:

Variables		Minterm	Maxterm
A	B	m_i	M_i
0	0	$\bar{A}\bar{B} = m_0$	$A+B = M_0$
0	1	$\bar{A}B = m_1$	$A+\bar{B} = M_1$
1	0	$A\bar{B} = m_2$	$\bar{A}+B = M_2$
1	1	$AB = m_3$	$\bar{A}+\bar{B} = M_3$

$(\bar{A}+B+\bar{C})$

PAGE NO.:
DATE: / /

3.26) K-map is a graphical method of simplifying a Boolean equation. It is a graphical chart which contains boxes

Structure of K-map :

- A K-map is a grid of cells, where each cell represents a minterm (for SOP) or a maxterm (for POS) of the Boolean function.
- A 2-variable K-map has 4 cells (2^2),
  A 3-variable K-map has 8 cells (2^3)
  A 4-variable K-map has 16 cells (2^4)
- The values (0 or 1) filled inside the boxes are the outputs (Y) corresponding to each combination of input variables.
- The arrangement of cells follows gray code so that only one variable changes between adjacent cells, which helps in identifying common terms during grouping.
- In SOP simplification, 1's are placed in cells corresponding to minterms.
  In POS simplification, 0's are placed in cells corresponding to maxterms.

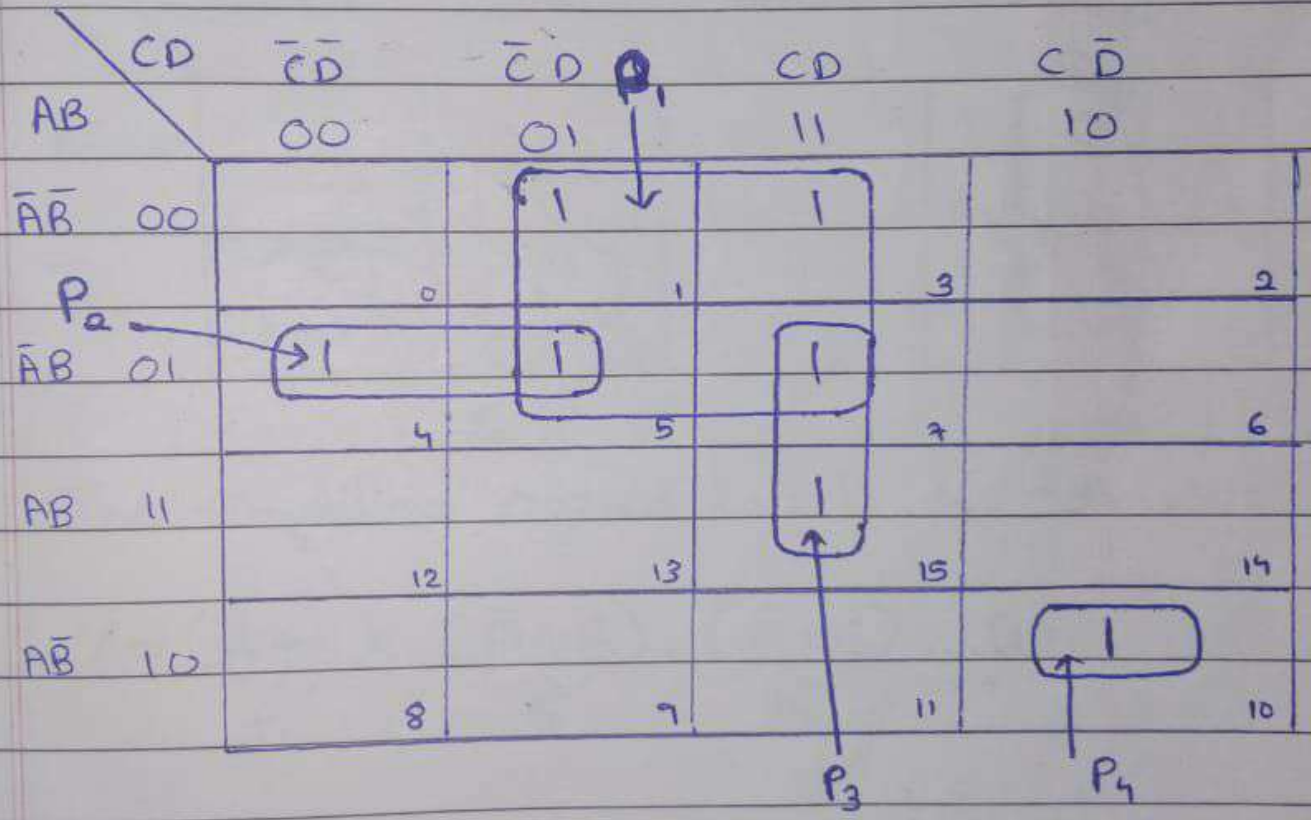
PAGE NO.:
DATE: / /

Q27)

a) The given expression is,

$Y = m_1 + m_3 + m_4 + m_5 + m_7 + m_{10} + m_{15}$

It can be expressed as K-map as shown below.



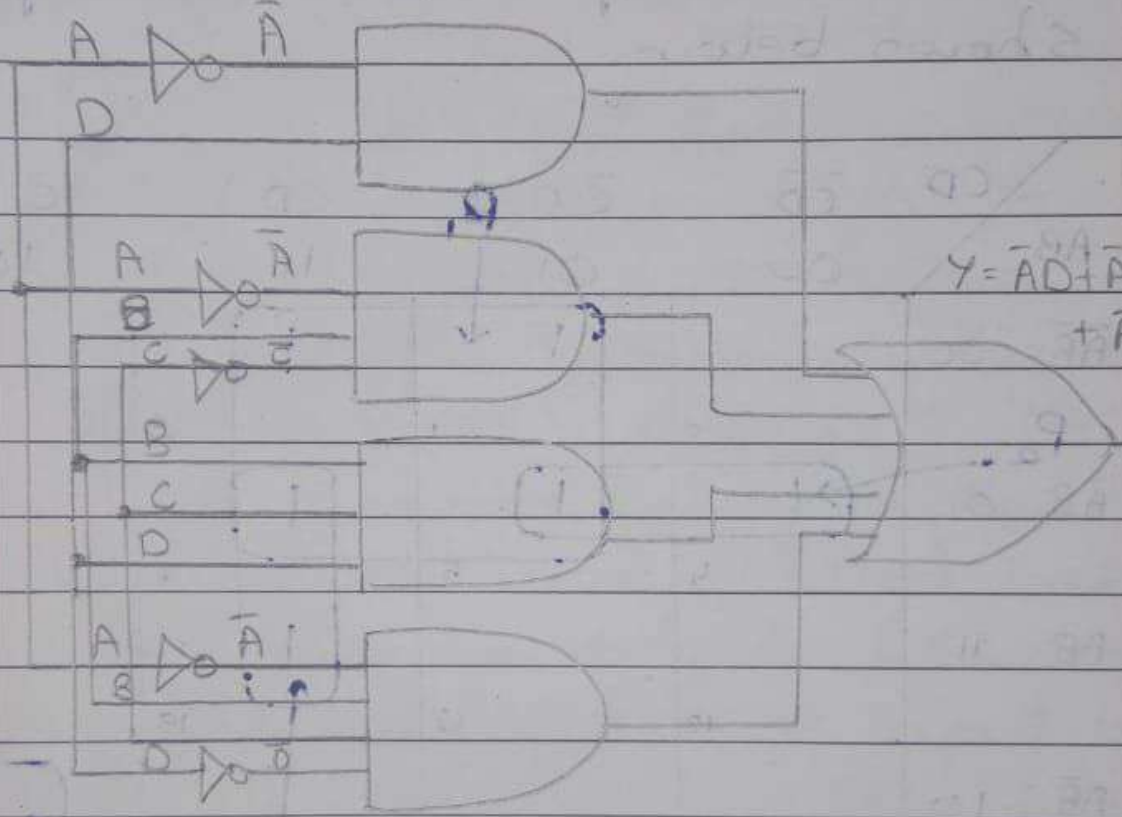
The simplified expression is as follows:

$Y = \underbrace{\bar{A}D}_{P_1} + \underbrace{\bar{A}B\bar{C}}_{P_2} + \underbrace{BCD}_{P_3} + \underbrace{A\bar{B}C\bar{D}}_{P_4} \ldots (1)$

PAGE NO.:
DATE: / /

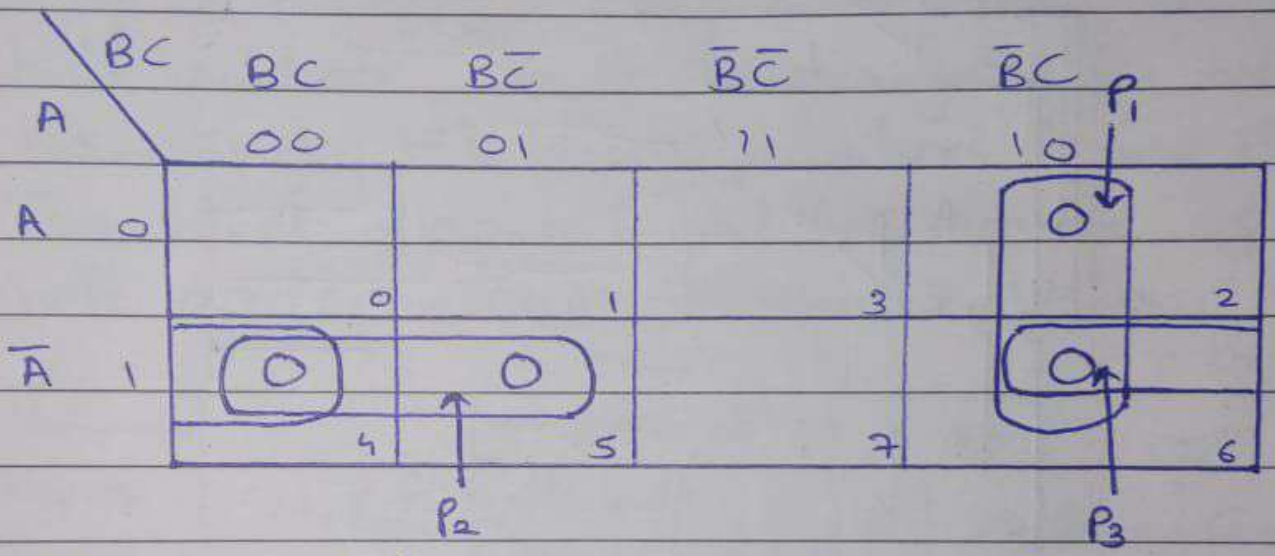
Realization:

Equation (1) can be realized as shown below.



PAGE NO.:
DATE: / /

b) The given expression is,
$Y = M_2 M_4 M_5 M_6$
It can be expressed as K-map as shown below:



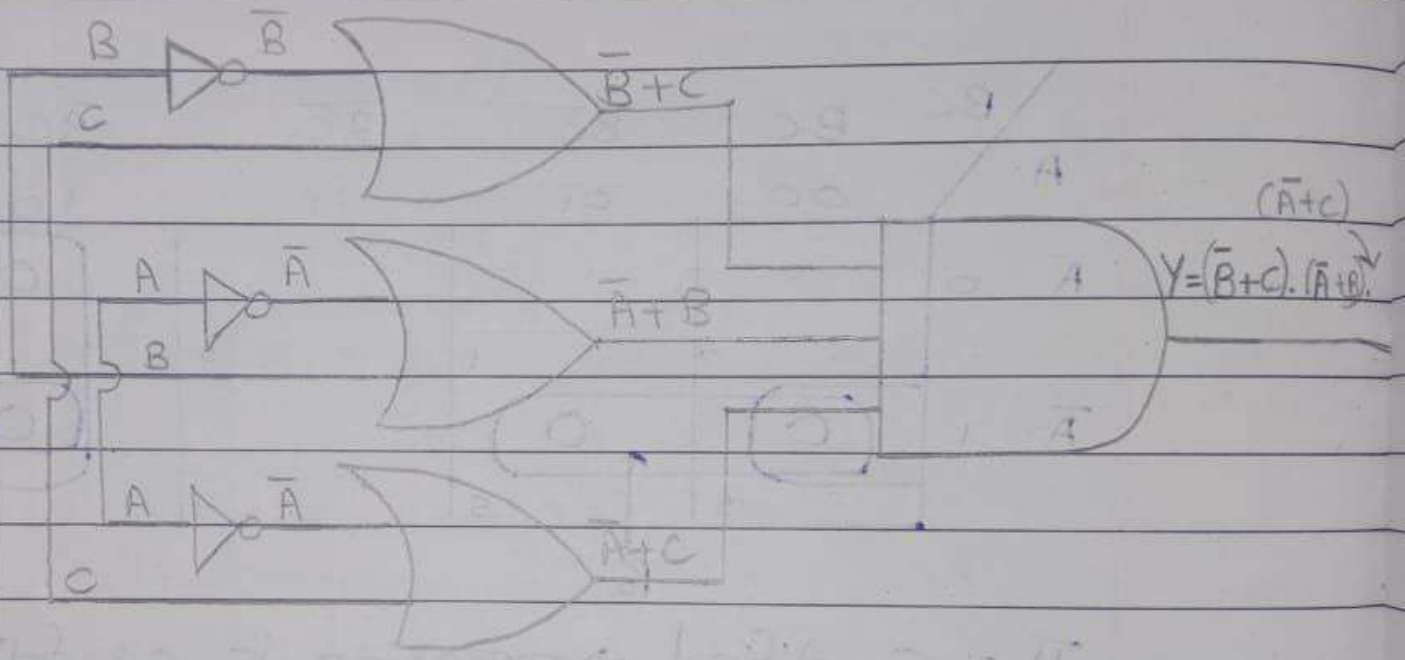
The simplified expression is as follows:

$$Y = \underbrace{(\bar{B}+C)}_{P_1} \cdot \underbrace{(\bar{A}+B)}_{P_2} \cdot \underbrace{(\bar{A}+C)}_{P_3} \quad \ldots (1)$$

PAGE NO.:
DATE: / /

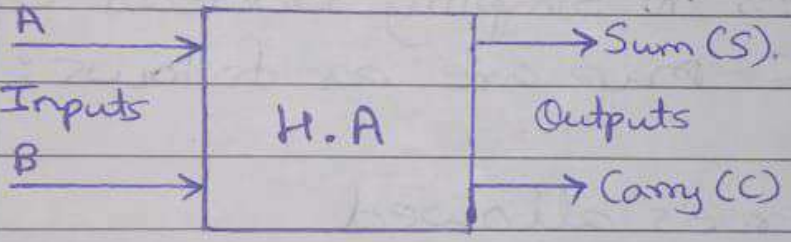
Realization:

Equation (1) can be realised as shown below:



PAGE NO.:
DATE: / /

Q.28) Half adder is a combinational logic circuit with two inputs and two outputs. It is the basic building block for addition of two "single" bit numbers. This circuit has two output namely "Carry" and "Sum". The half adder circuit is designed to add two single bit binary numbers A and B. The block diagram and truth table of half adder circuit is shown below.



BLOCK DIAGRAM

I/P		O/P	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TRUTH TABLE

A \ B	$\bar{B}$ 0	B 1
$\bar{A}$ 0	0	(1)
A 1	(1)	0

K-map for sum

A \ B	$\bar{B}$ 0	B 1
$\bar{A}$ 0	0	0
A 1	0	(1)

K-map for carry

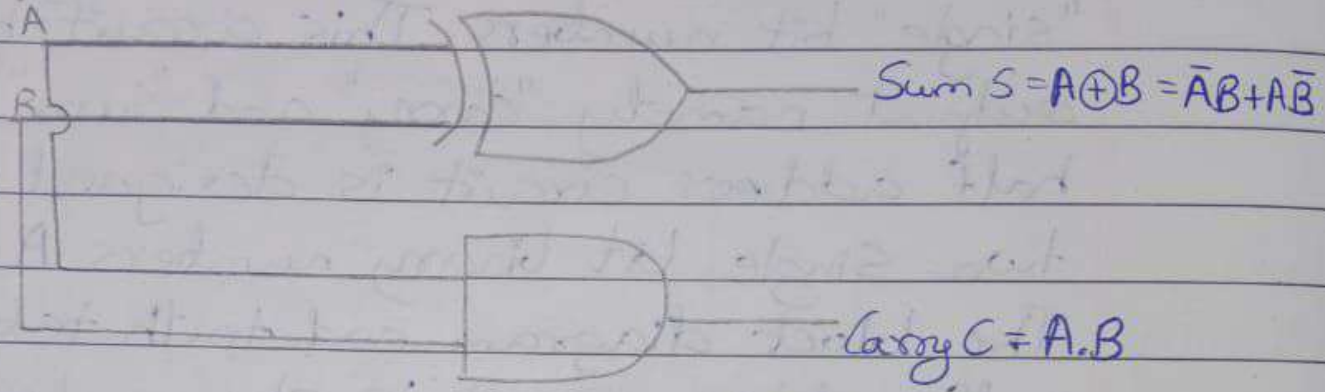
The boolean expression obtained from the K-map for sum and carry are as follows:

$S = \bar{A}B + A\bar{B} = A \oplus B$ ... (For sum)

$C = AB$ ... (For carry)

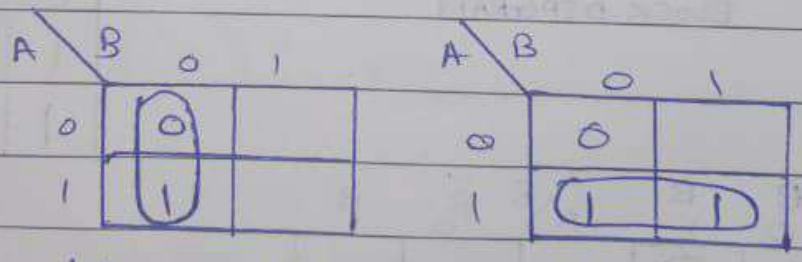
PAGE NO.:
DATE: / /

Hence the half adder circuit is shown below:

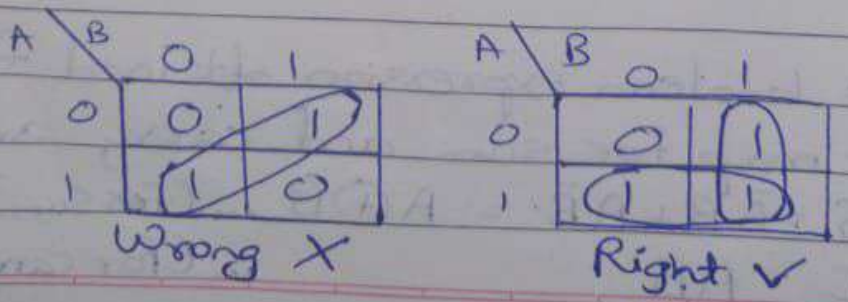


Q.31) The rules to simplify Boolean equation using K-map are as follows:

1. No zeroe's allowed

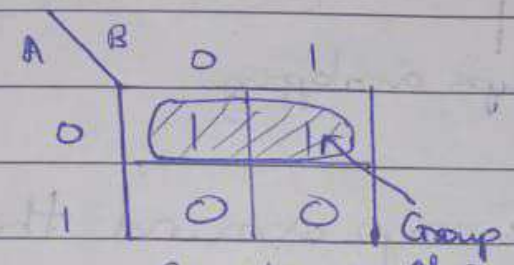


2. Groups may be horizontal or vertical but no diagonal.

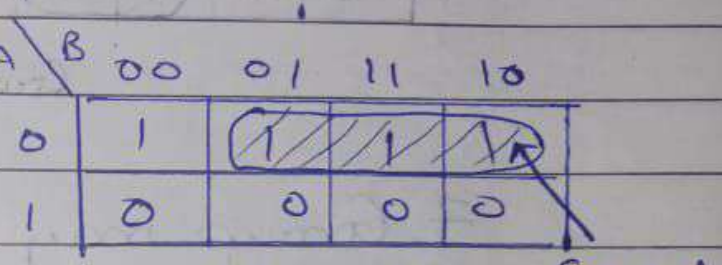


PAGE NO.:
DATE: / /

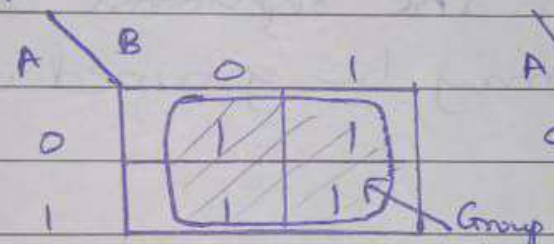
3. Only power of 2 number of cells in each group. (2^n) If $n=1$, a group will contain two 1's since $2^1 = 1$.



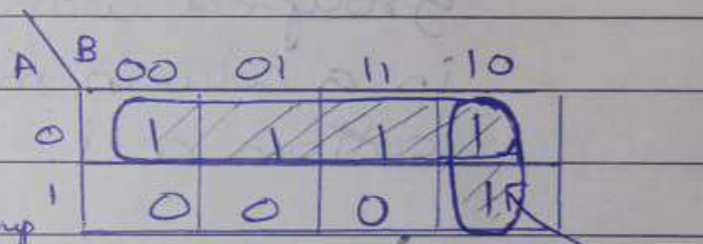
Right ✓



Wrong X

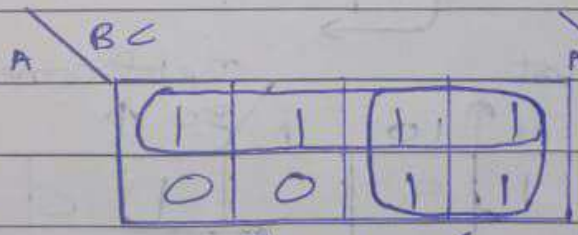


Right ✓

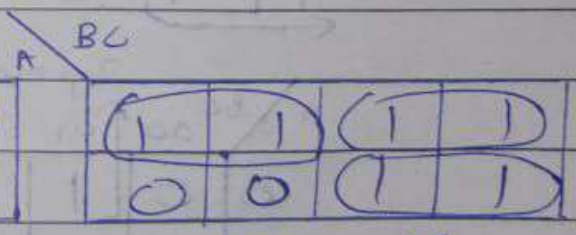


Wrong X

4. Group should be as large as possible

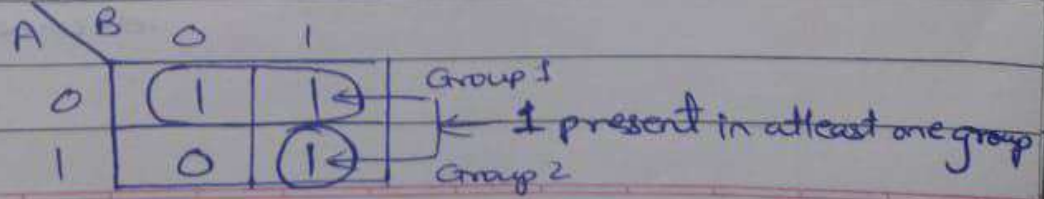


Right ✓



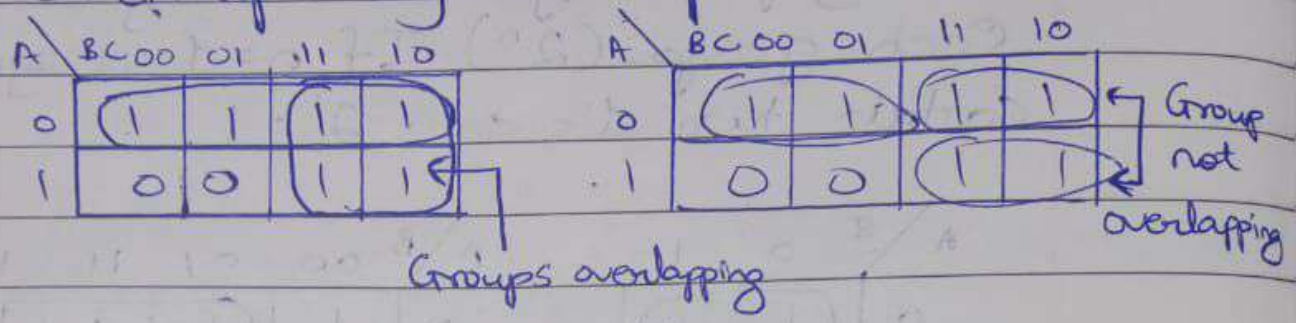
Wrong X

5. Each cell containing a "one" must be in atleast one group.

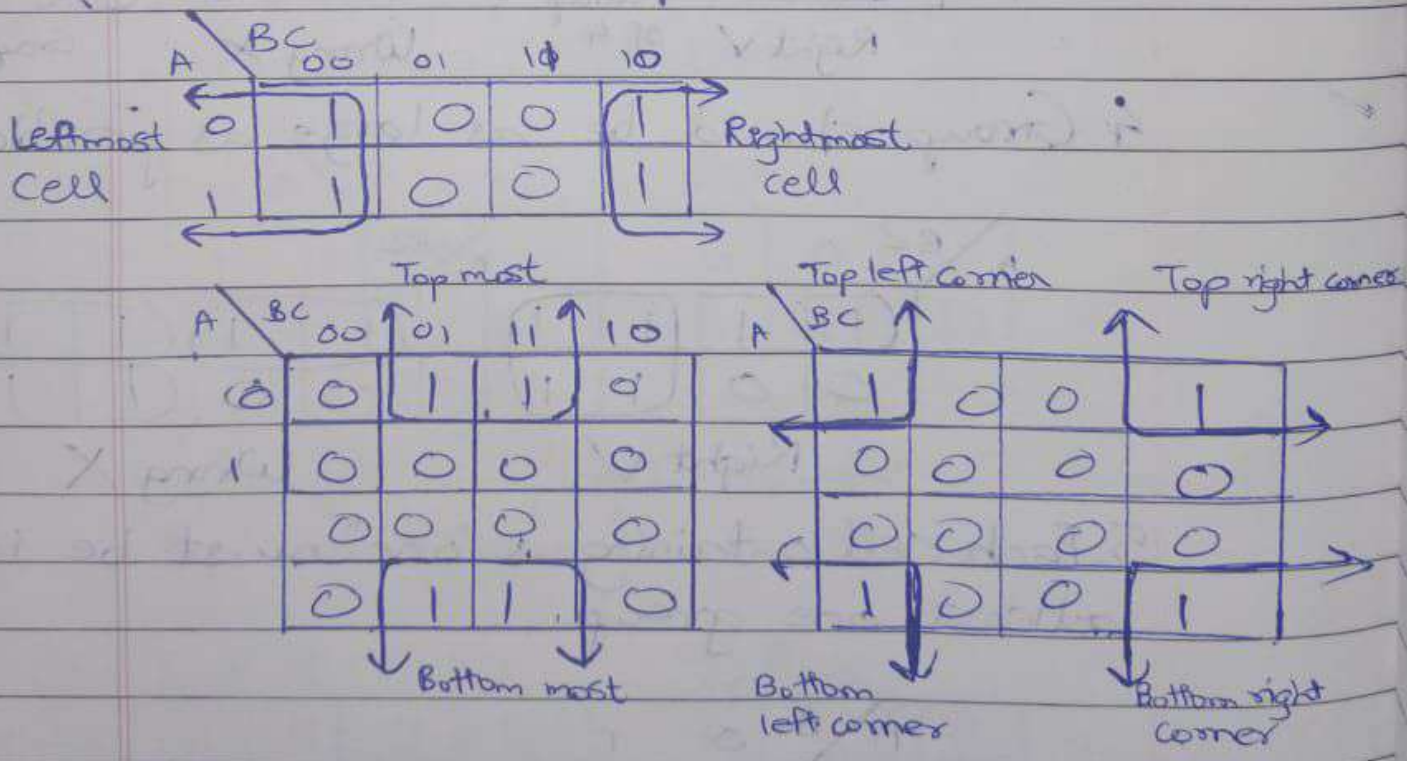


PAGE NO.:
DATE: / /

6. Group May Overlap.

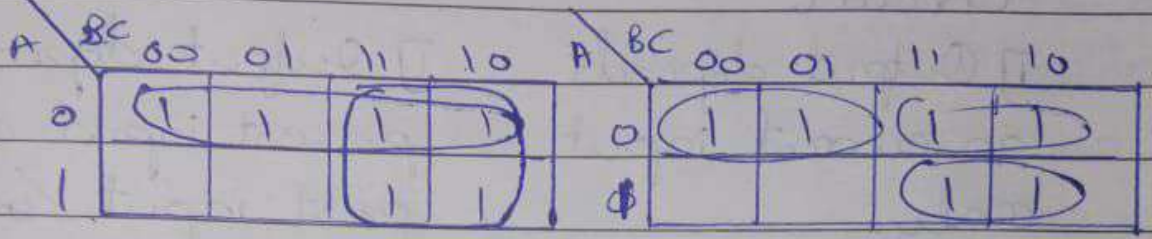


7. Group may wrap around the table. The leftmost cell in a row may be grouped with the rightmost and top cell in a column may be grouped with the bottom cell.



PAGE NO.:
DATE: / /

8. There should be as few groups as possible.



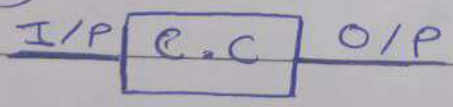
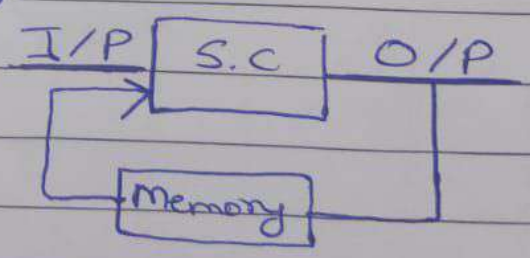
Right ✓

Wrong ✗

PAGE NO.:
DATE: / /

Flip Flops

Q.1]

Combinational Circuit	Sequential Circuit
1] Output depends on current input only.	1] Output depends on present input as well as past input / output.
2] Memory is not necessary	2] Memory is necessary
3] Clock input not necessary	3] Clock input necessary
4] Ex:- Adder, Substractors, Multiplexer, De-multiplexer, Decoder, etc.	4] Ex:- Flip flops, registers, counters, etc.
5) 	5] 

PAGE NO.:
DATE: / /

Q.2]

a) Flip flop :- Flip-flop is a bistable logic circuit i.e. its output have two stable states. The stable states of a flip-flop can be changed only by changing the set of its input.

b) Clock Signal :- The clock signal is a timing signal. Every sequential signal will have this timing signal applied. Clock is a rectangular signal with a duty cycle equal to 50%. The clock signal repeats itself after every T seconds. Hence the clock frequency is 1/T