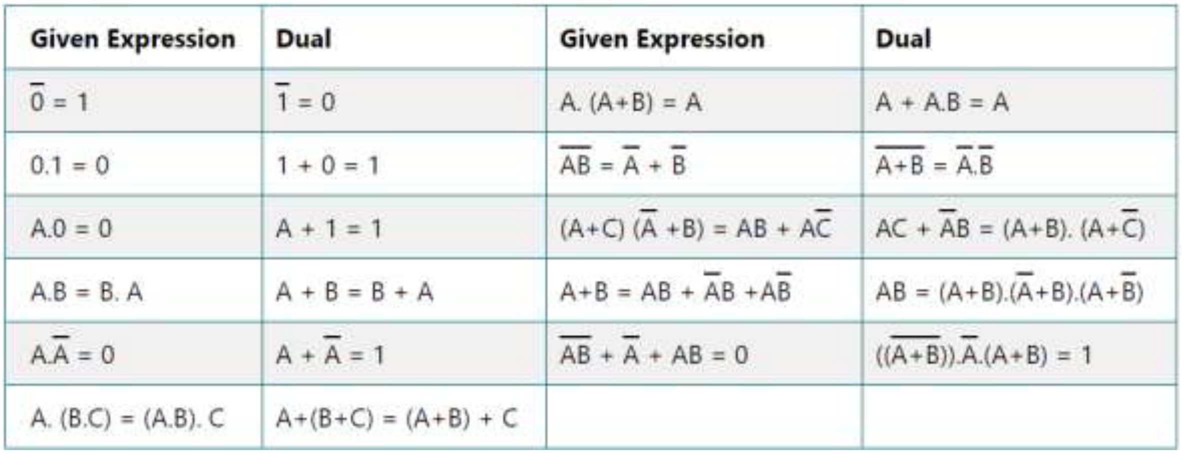
Computer System Architecture
COMP201TH
Lecture-5
Don’t Care Conditions

- **Duality Theorem of Boolean Algebra:**
  - This theorem states that the dual of the Boolean function is obtained by interchanging the logical AND operator with logical OR operator and zeros with ones. For every Boolean function, there will be a corresponding Dual function.

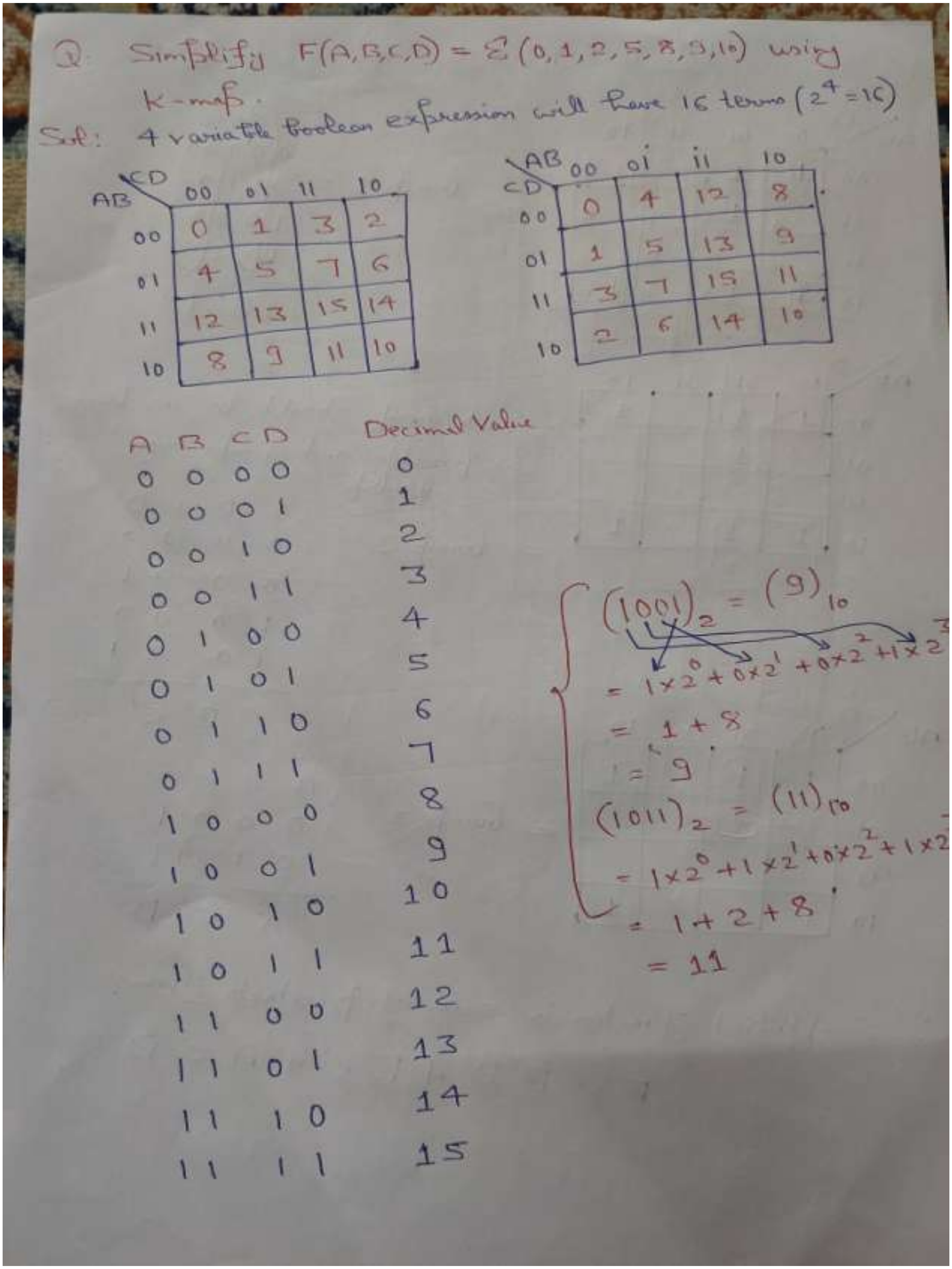


Given Expression	Dual	Given Expression	Dual
$\overline{0} = 1$	$\overline{1} = 0$	$A. (A+B) = A$	$A + A.B = A$
$0.1 = 0$	$1 + 0 = 1$	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A+B} = \overline{A}.\overline{B}$
$A.0 = 0$	$A + 1 = 1$	$(A+C)\ (\overline{A} +B) = AB + A\overline{C}$	$AC + \overline{A}B = (A+B).\ (A+\overline{C})$
$A.B = B.\ A$	$A + B = B + A$	$A+B = AB + \overline{A}B + A\overline{B}$	$AB = (A+B).(\overline{A}+B).(A+\overline{B})$
$A.\overline{A} = 0$	$A + \overline{A} = 1$	$\overline{AB} + \overline{A} + AB = 0$	$((\overline{A+B})).\overline{A}.(A+B) = 1$
$A.\ (B.C) = (A.B).\ C$	$A+(B+C) = (A+B) + C$		

- **Complement of a function (Using De-Morgan’s Law):**
  - De-morgan’s law :
    - (xy)`=x`+y`
    - (x+y)` = x`.y`
  - E.g. find the complement of function f = AB + CD +BD`

f` = (AB+CD+BD`)`
= (AB)`.(CD)`.(BD`)`
= (A`+B`).(C`+D`).(B`+(D`)`)
= (A`+B`)(C`+D`)(B`+D)

- **Product of Sums Simplification:**
  - In previous lecture, the Boolean expressions we derived from the maps were expressed in sum-of-products form.
  - The 1’s in the map represent the minterms that produce 1 for the function. The squares not marked by 1 represent the minterms that produce 0 for the function. If we mark the empty squares with 0’s and combine them into groups of adjacent squares, we obtain the complement of the function f`. Taking the complement of f` produces an expression for f in product-of-sums form.
  - If Boolean expression is given in form of maxterms (π), then K-map is simplified by following the same procedure as used for SOP form. In this case group of 0’s are formed rather than group of 1’s.



Q. Simplify $F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$ using K-map.

Sol: 4 variable Boolean expression will have 16 terms $(2^4=16)$

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

A B C D	Decimal Value
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	10
1 0 1 1	11
1 1 0 0	12
1 1 0 1	13
1 1 1 0	14
1 1 1 1	15

$$(1001)_2 = (9)_{10}$$

$$= 1\times 2^0 + 0\times 2^1 + 0\times 2^2 + 1\times 2^3$$

$$= 1 + 8$$

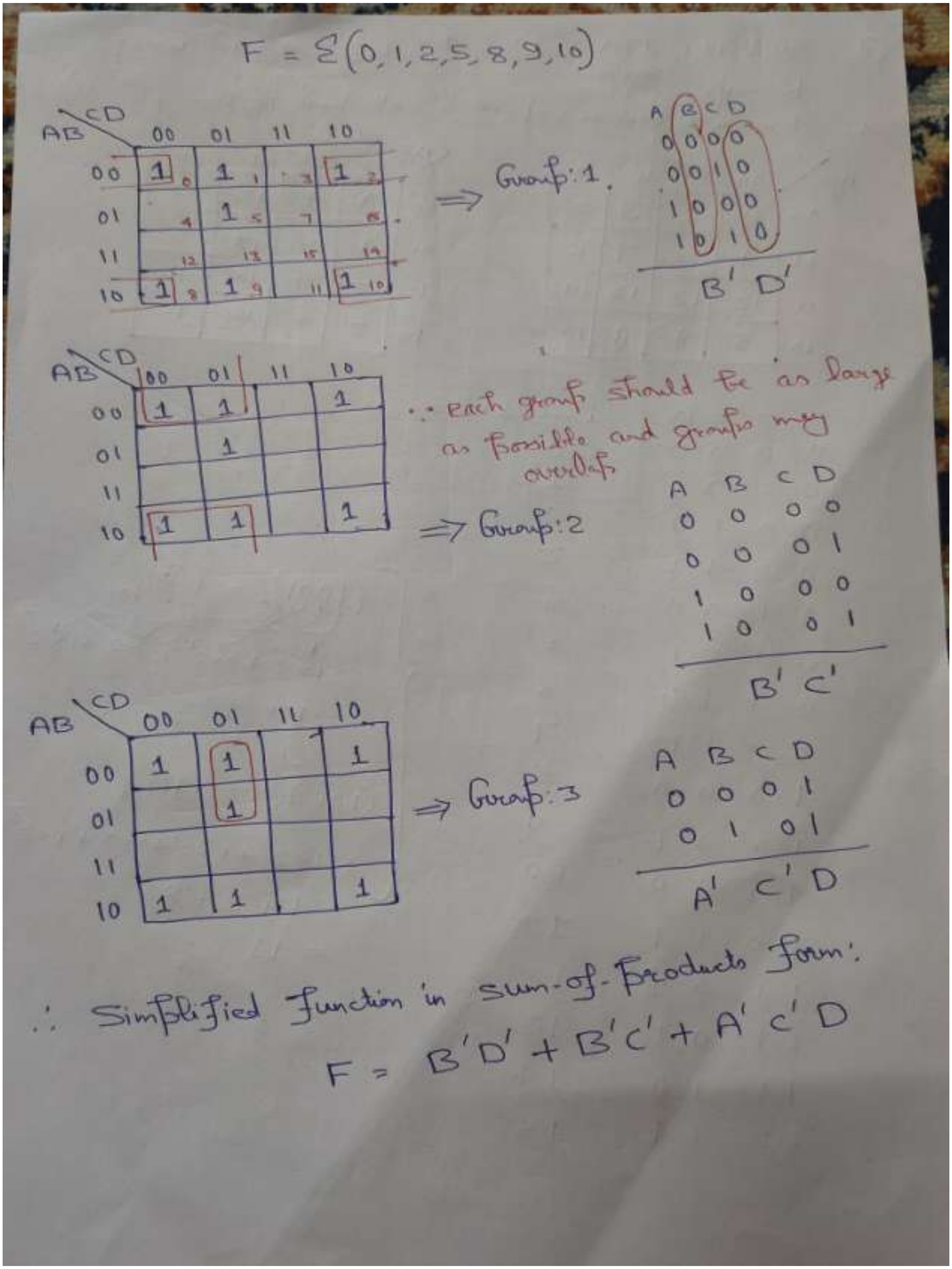
$$= 9$$

$$(1011)_2 = (11)_{10}$$

$$= 1\times 2^0 + 1\times 2^1 + 0\times 2^2 + 1\times 2$$

$$= 1 + 2 + 8$$

$$= 11$$



$F = \Sigma(0,1,2,5,8,9,10)$

AB \ CD	00	01	11	10
00	1 0	1 1	3	1 2
01	4	1 5	7	6
11	12	13	15	14
10	1 8	1 9	11	1 10

$\Rightarrow$ Group: 1.

A	B	C	D
0	0	0	0
0	0	1	0
1	0	0	0
1	0	1	0

$B'D'$

AB \ CD	00	01	11	10
00	1	1		1
01		1		
11				
10	1	1		1

∴ Each group should be as large as possible and groups may overlap

$\Rightarrow$ Group: 2

A	B	C	D
0	0	0	0
0	0	0	1
1	0	0	0
1	0	0	1

$B'C'$

AB \ CD	00	01	11	10
00	1	1		1
01		1		
11				
10	1	1		1

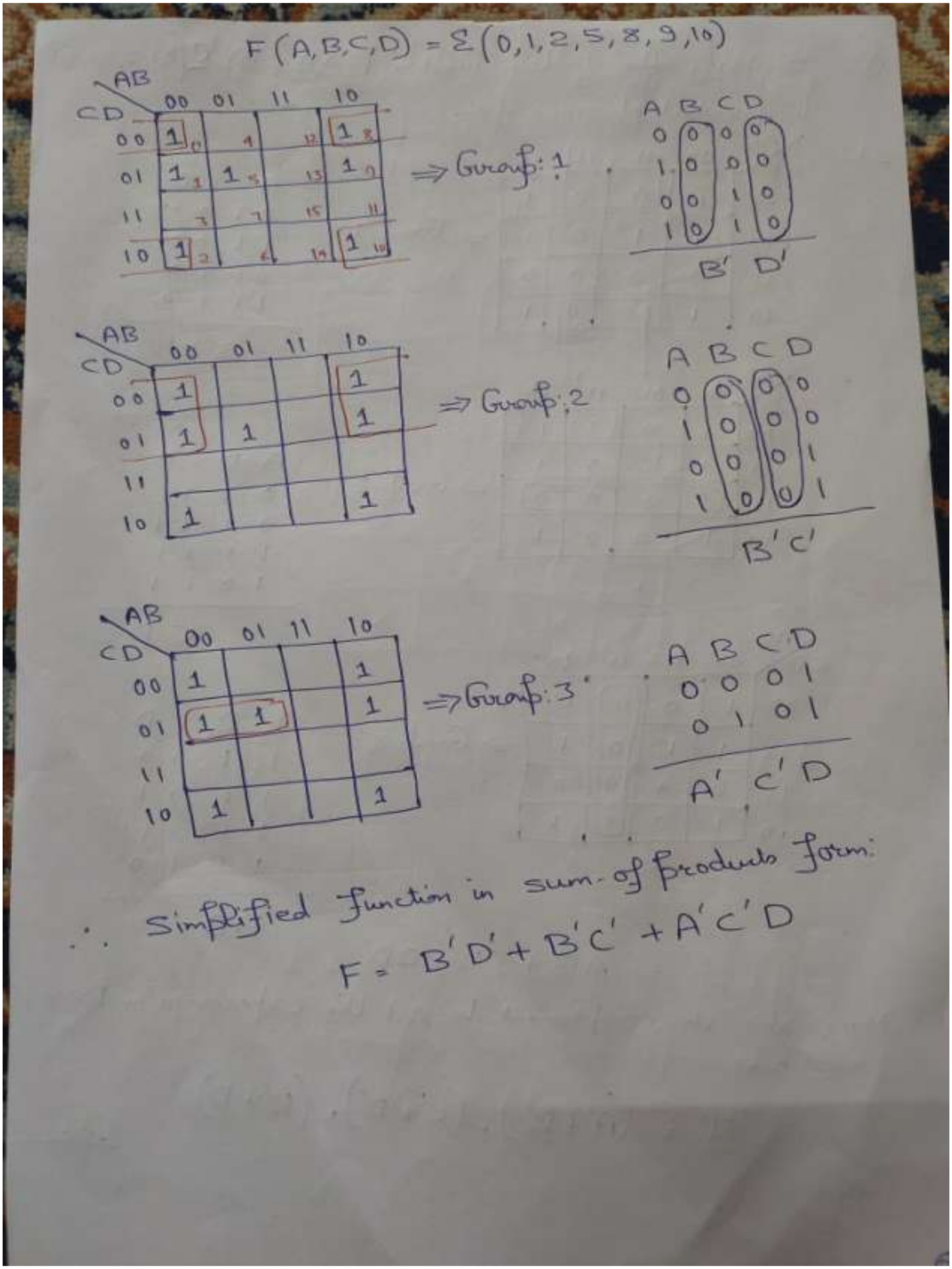
$\Rightarrow$ Group: 3

A	B	C	D
0	0	0	1
0	1	0	1

$A'C'D$

∴ Simplified function in sum-of-products form:

$F = B'D' + B'C' + A'C'D$



$F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$

CD \ AB	00	01	11	10
00	1_{0}	$_{4}$	$_{12}$	1_{8}
01	1_{1}	1_{5}	$_{13}$	1_{9}
11	$_{3}$	$_{7}$	$_{15}$	$_{11}$
10	1_{2}	$_{6}$	$_{14}$	1_{10}

$\Rightarrow$ Group: 1

A	B	C	D
0	0	0	0
1	0	0	0
0	0	1	0
1	0	1	0
	B'		D'

CD \ AB	00	01	11	10
00	1			1
01	1	1		1
11				
10	1			1

$\Rightarrow$ Group: 2

A	B	C	D
0	0	0	0
1	0	0	0
0	0	0	1
1	0	0	1
	B'	C'	

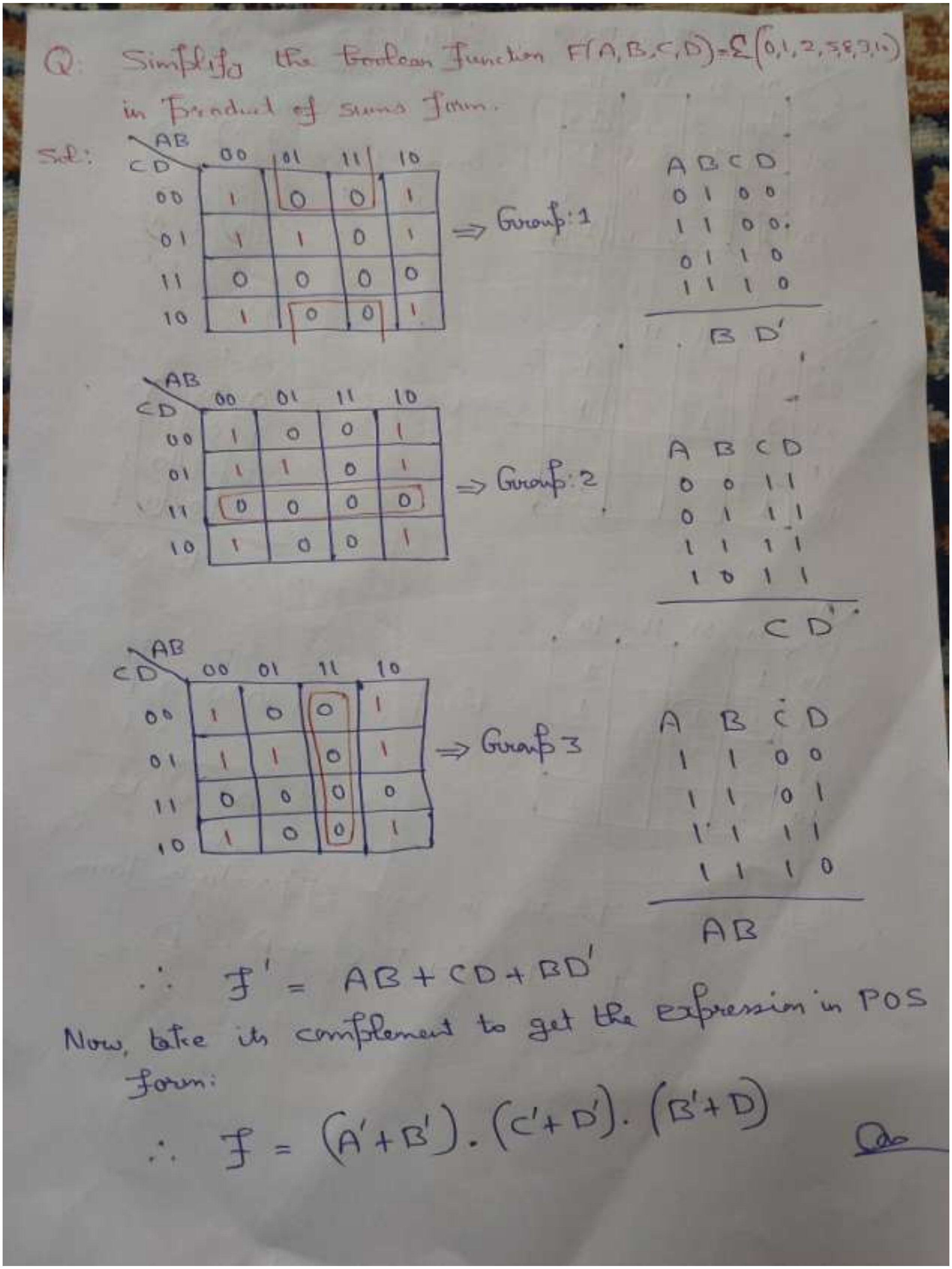
CD \ AB	00	01	11	10
00	1			1
01	1	1		1
11				
10	1			1

$\Rightarrow$ Group: 3

A	B	C	D
0	0	0	1
0	1	0	1
A'		C'	D

$\therefore$ Simplified function in sum-of products form:

$$F = B'D' + B'C' + A'C'D$$



Q: Simplify the Boolean Function $F(A,B,C,D)=\Sigma(0,1,2,5,8,9,10)$ in Product of sums form.

Sol:

AB CD	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	0	0	0
10	1	0	0	1

⇒ Group: 1

A	B	C	D
0	1	0	0
1	1	0	0
0	1	1	0
1	1	1	0

BD'

AB CD	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	0	0	0
10	1	0	0	1

⇒ Group: 2

A	B	C	D
0	0	1	1
0	1	1	1
1	1	1	1
1	0	1	1

CD

AB CD	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	0	0	0
10	1	0	0	1

⇒ Group 3

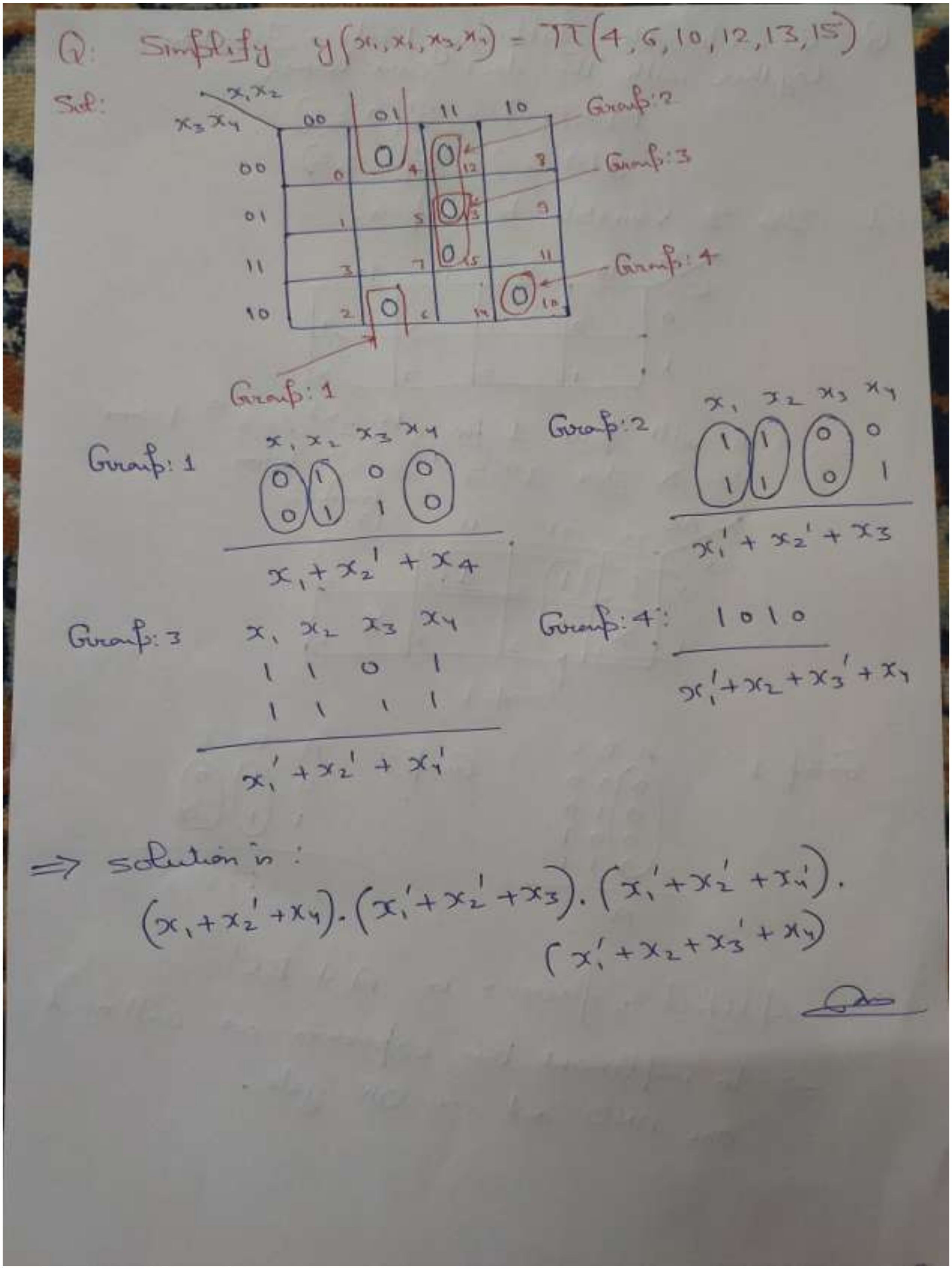
A	B	C	D
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0

AB

$\therefore \; F' = AB + CD + BD'$

Now, take its complement to get the expression in POS form:

$\therefore \; F = (A'+B').(C'+D').(B'+D)$



Q: Simplify $y(x_1,x_2,x_3,x_4) = \Pi(4,6,10,12,13,15)$

Sol:

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	0	0 4	0 12	8
01	1	5	0 13	9
11	3	7	0 15	11
10	2	0 6	14	0 10

Group: 1 Group: 2 Group: 3 Group: 4

Group: 1

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ \hline \end{array}$$

$x_1 + x_2' + x_4$

Group: 2

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ \hline \end{array}$$

$x_1' + x_2' + x_3$

Group: 3

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$x_1' + x_2' + x_4'$

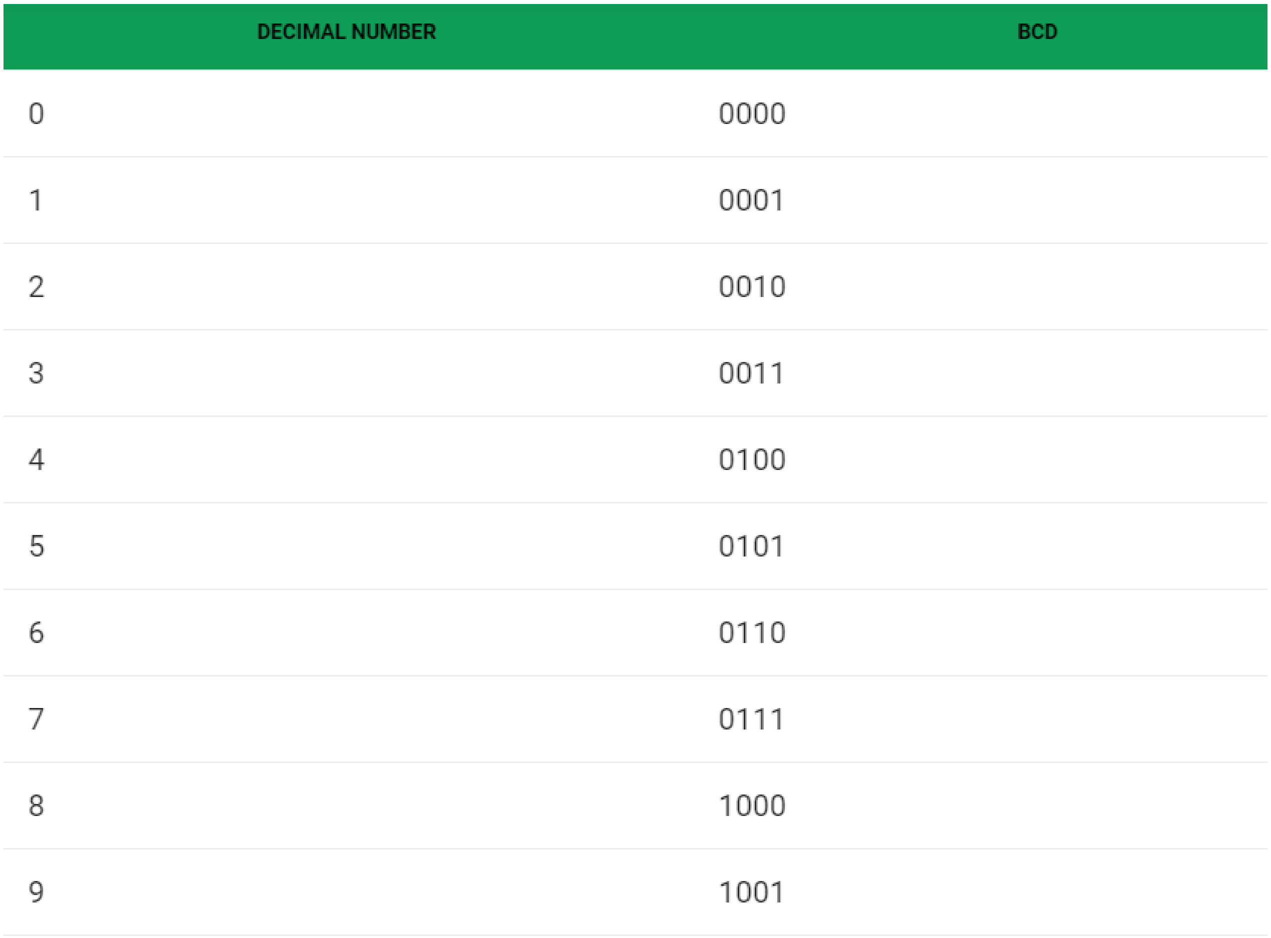
Group: 4: 1 0 1 0

$x_1' + x_2 + x_3' + x_4$

⇒ solution is :

$$(x_1 + x_2' + x_4) . (x_1' + x_2' + x_3) . (x_1' + x_2' + x_4') . (x_1' + x_2 + x_3' + x_4)$$

- **Don’t Care Condition:**
    - Binary Coded Decimal (BCD):
        - BCD is another process for converting decimal numbers into their binary equivalents.
        - It is a form of binary encoding where each digit in a decimal number is represented in the form of bits.
        - These are generally used in digital displays.



DECIMAL NUMBER	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example:

Convert (123)10 in BCD:

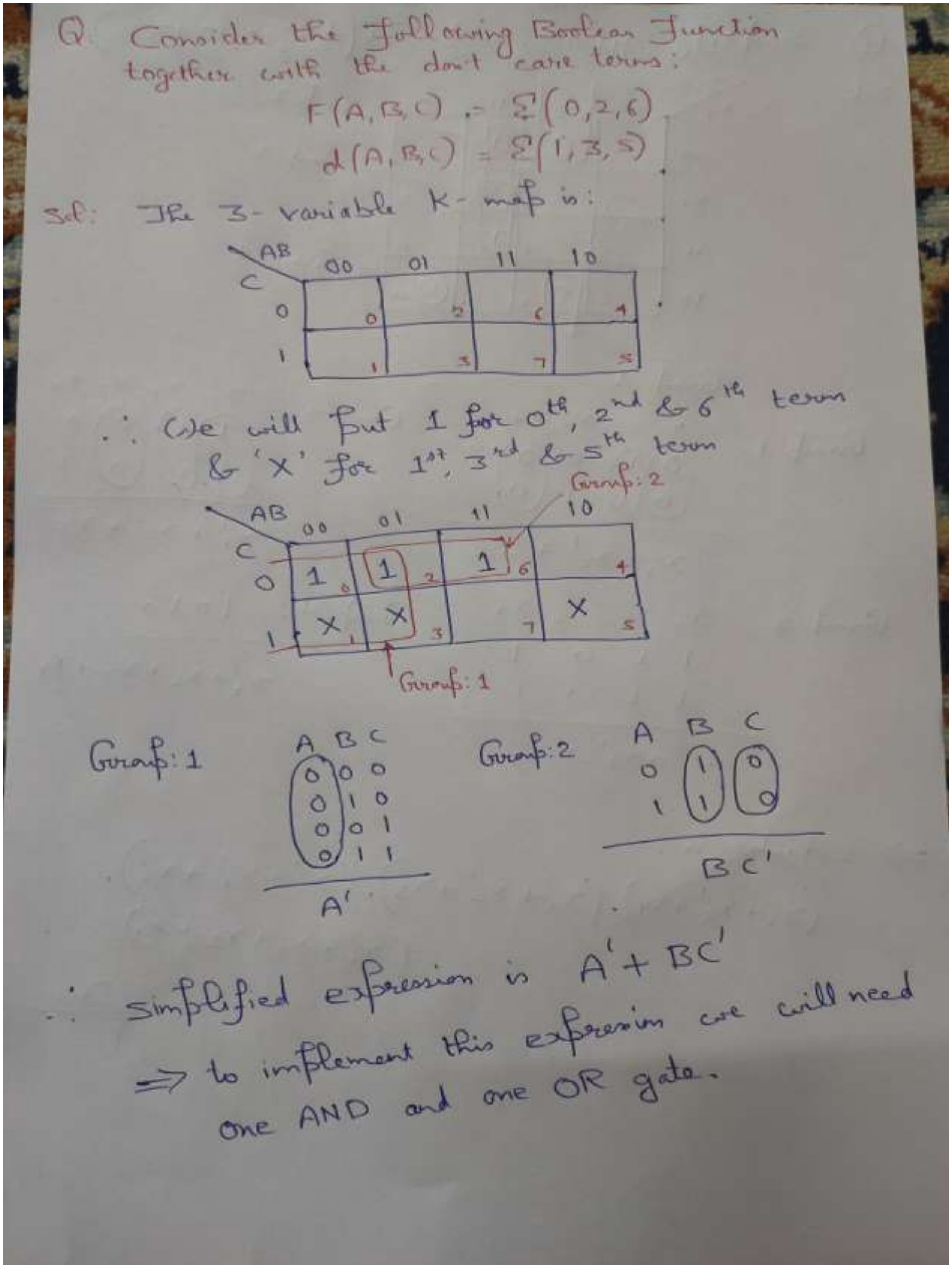
1 → 0001

2 → 0010

3 →0011

So, in BCD is: 000100100011

- **Don’t Care Condition:**
    - The “Don’t care” conditions allow us to replace the empty cell of a K-map to form a grouping of variables.
    - While forming groups of cells, we can consider a “”Don’t care” cell as either 1 or 0 or we can simply ignore the cell.
    - Thus, “Don’t care” condition can help us to form a larger group of cells.
    - It is not always necessary to fill in the complete truth table for some real-world problems. We may have a choice to not fill in the complete table e.g. while dealing with BCD numbers encoded as four bits, we may not care about any codes above the BCD range of (0,1,2,...9). We would not normally care to fill in those codes because those codes (1010,1011,1100,1101,1110,1111) will never exist as long as we are dealing only with BCD encoded numbers. These six invalid codes are don’t cares as far as we are concerned. i.e.
        - We do not care what output out logic circuit produces for these don’t cares.
    - We only use don’t care conditions in a group if simplifies the logic.
    - The cross (x) symbol is used to represent the “don’t care” cell in K-map.
- **Significance of “Don’t care” conditions:**
    - Don’t care conditions have the following significance with respect to the digital circuit design:
        - **Simplification:** these are used to further simplify the Boolean output expression.
        - **Lesser number of gates:** → make the digital circuit design more economical.
        - **Prevention of Hazards:** Don’t care also prevents hazards in digital systems.
- **Example:**



Q. Consider the following Boolean Function together with the don't care terms:

$$F(A,B,C) = \Sigma(0,2,6)$$
$$d(A,B,C) = \Sigma(1,3,5)$$

Sol: The 3-variable K-map is:

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

∴ We will put 1 for 0th, 2nd & 6th term & 'X' for 1st, 3rd & 5th term

Group: 2

C \ AB	00	01	11	10
0	1 0	1 2	1 6	4
1	X 1	X 3	7	X 5

Group: 1

Group: 1

A	B	C
0	0	0
0	1	0
0	0	1
0	1	1

A'

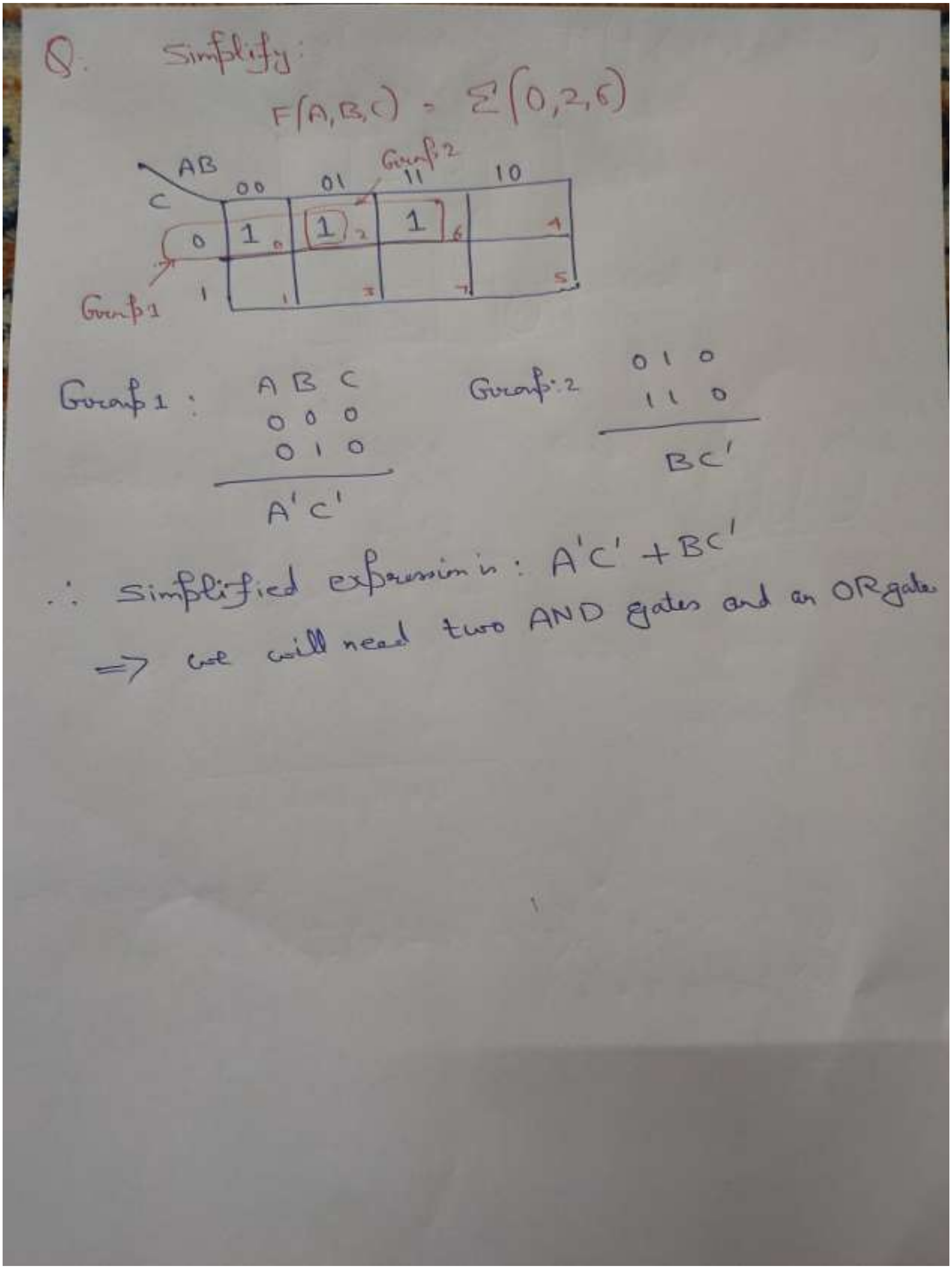
Group: 2

A	B	C
0	1	0
1	1	0

BC'

∴ simplified expression is $A' + BC'$

⇒ to implement this expression we will need one AND and one OR gate.



Q. Simplify:

$F(A,B,C) = \Sigma(0,2,6)$

C \ AB	00	01	11	10
0	1 $_{0}$	1 $_{2}$	1 $_{6}$	$_{4}$
1	$_{1}$	$_{3}$	$_{7}$	$_{5}$

Group 2
Group 1

Group 1 :

A	B	C
0	0	0
0	1	0

$A'C'$

Group : 2

0	1	0
1	1	0

BC'

∴ Simplified expression is : $A'C' + BC'$

⇒ we will need two AND gates and an OR gate