Date __/__/____ Page No.: _____

Number Systems

- Binary (LL) - 0, 1
- Octal (HL) - 0, 1, 2, 3, 4, 5, 6, 7
- Decimal (HL) - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Hexadecimal (HL) - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  (10) (11) (12) (13) (14) (15)

Base of a number - number of characters available for a number system.

Place value :

$$(9436)_{10} = 9\times10^3 + 4\times10^2 + 3\times10^1 + 6\times10^0$$

Binary ⇒ $\ldots 2^3\ 2^2\ 2^1\ 2^0 . 2^{-1}\ 2^{-2}\ 2^{-3}$

1 1 0 1 . 0 1

Eg → $(1101.01)_2$

$$1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 . 0\times2^{-1} + 1\times2^{-2}$$

$$8 + 4 + 0 + 1 . 0 + \frac{1}{4}$$

$$(13.25)_{10}$$

Eg → $(10111.11)_2$

= 1 0 1 1 1 . 1 1

16 8 4 2 1 . 0.5 + 0.25

= $(23.75)_{10}$

Scanned with CamScanner

Date ___/___/___

Octal → $8^3\ 8^2\ 8^1\ 8^0 \cdot 8^{-1}\ 8^{-2}\ 8^{-3}$

Dec → $10^3\ 10^2\ 10^1\ 10^0 \cdot 10^{-1}\ 10^{-2}\ 10^{-3}$

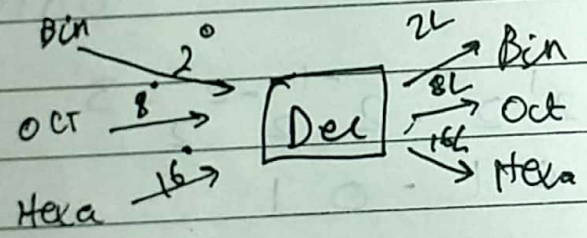
Hex → $16^3\ 16^2\ 16^1\ 16^0 \cdot 16^{-1}\ 16^{-2}\ 16^{-3}$

$8^2\ 8^1\ 8^0\ 8^{-1}\ 8^{-2}$

$(425.13)_8 = (4\times8^2 + 2\times8 + 5\times1 \cdot \frac{1}{8} + \frac{3}{64})_{10}$

$256 + 16 + 5 \cdot \frac{8+3}{64}$

$277 \cdot \frac{11}{64}$



1. Decimal → Binary

$(27.35)_{10} = \cancel{(11011.0101)_2}$ $(11011.0101)_2$

2 | 27 → 1
2 | 13 → 1
2 | 6 → 0
2 | 3 → 1
1 → 1

~~2 | 35 → 1~~
~~2 | 17 → 1~~
~~2 | 8 → 0~~
~~2 | 4 → 0~~
~~2 | 2 → 0~~
~~1 → 1~~

$0.35 \times 2 = 0.7 \rightarrow 0$
$0.7 \times 2 = 1.4 \rightarrow 1$
$0.4 \times 2 = 0.8 \rightarrow 0$
$0.8 \times 2 = 1.6 \rightarrow 1$

Scanned with CamScanner

Date __/__/____ Page No.: _____

2. Decimal To Octal

$(27.35)_{10} = (33.263)_8$

8 | 27 → 3
3 → 3

0.35×8 = 2.80 → 2
0.8×8 = 6.4 → 6
0.4×8 = 3.2 → 3

3. Decimal To Hexadecimal

$(27.35)_{10} = (1B.599)_{16}$

16 | 27 → B
1 → 1

0.35×16 = 5.6 → 5
0.6×16 = 9.6 → 9

4. Octal to Binary

$(456.15)_8 = (100101110.001101)_2$

5. Hexadecimal → Binary

$(2BC1.1A)_{16} = \cancel{(0010\ 1011\ 1100\ 0001.0001\ 1010)_2}$

$(0010\ 1011\ 1100\ 0001.0001\ 1010)_2$

6. Binary → Octal

$(01011011.1011)_2 \rightarrow (267.54)_8$

7. Binary → Hexadecimal

$(01011011.1011)_2 \rightarrow (B7.B)_{16}$

"Cultivation of mind should be the ultimate aim of human existence." —B.R.Ambedkar

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

* Binary Arithmetic

→ Addition :

→ $0+0=0$

→ $1+0=1$

→ $0+1=1$

→ $1+1=0$ Carry 1 (10)

Eg →

```
  1 1 1 1
  1 0 1 1 0 1
+   1 1 0 1 1
-----------
1 0 0 1 0 0 0
```

$(101101)_2 + (11011)_2 = (1001000)_2$

→ Subtraction

Direct Method :

$0-0=1$

$0-1=1$ borrow 1

$1-0=1$

$1-1=0$

Eg →

```
  0 1 0 0 1 1
  1 0 0 1 0 0 0
-     1 0 1 0 1
-----------
  0 1 1 0 0 1 1
```

$(1001000)_2 - (10101) = (0110011)_2$

Scanned with CamScanner

Date ___ Page No.: ___

Eg → $\begin{array}{r} 1000000 \\ -101 \\ \hline 11011 \end{array}$

$(1000000)_2 - (101)_2 = (11011)_2$

1's Complement:

$(1001000)_2 - (10101)_2 = (110011)_2$

A 1001000
B 0010101

~~⇒ A 1001000~~
~~1' ← {B} 0010101~~

⇒ A = 1001000
1' of B = +1101010

$\begin{array}{r} 1001000 \\ +1101010 \\ \hline 10110010 \\ \hline 110011 \end{array}$ (Ans)

Scanned with CamScanner

2's Complement

Add 1 to 1's Complement LSB

$(1001000)_2 - (0010101)_2 = (110011)_2$

~~1101011~~ 1101010

$+ 1$

1101011

$+ \; 1001000$

~~10110011~~

$(110011)_2$

Eg ⇒ $(100000)_2 - (101)_2 = (11011)_2$

1's Complement

100000

$- \; 000101$

111010

~~10111010~~

1

11011

2's Complement

000101 ⇒ 111010 + 1 ⇒ 111011

100000

$+ \; 111011$

~~10111011~~

~~1~~011011

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

→ Multiplication

→ $0 \times 0 = 0$

→ $1 \times 1 = 1$

→ $0 \times 1 = 0$

→ $1 \times 0 = 0$

Eg →

$$
\begin{array}{r}
1011 \\
\times\ 101 \\
\hline
1011 \\
0000\times \\
1011\ \times\times \\
\hline
110111
\end{array}
$$

$(1011)_2 \times (101)_2 = (110111)_2$

→ Division

$$
\begin{array}{r}
101.011 \\
(01)\overline{)11011.000} \\
-101\downarrow\downarrow \\
\hline
00111 \\
101 \\
\hline
1000 \\
101 \\
\hline
00110 \\
-101 \\
\hline
011
\end{array}
$$

Scanned with CamScanner

Date __/__/__ Page No.: ____

Boolean Algebra

→ It was developed by Prof. George Booyle.

→ It works on bivalent logic i.e. 0 & 1
F T

Basic Boolean Operations

AND (x)	OR (+)	NOT (-)
$0 \cdot 0 = 0$	$0+0=0$	
$0 \cdot 1 = 0$	$1+0=1$	$\bar{0} = 1$
$1 \cdot 0 = 0$	$0+1=1$	$\bar{1} = 0$
$1 \cdot 1 = 1$	$1+1=1$	

* $1+1=1$ is boolean operator 'OR', it isn't addition.

Conventional	Boolean
→ 0 → 9	
→ any number of c	0 & 1
→ Each term can be represented by more than 1 alph. $x=y=10$	Only 2 constant. Each term can only be represented by one variable $x=1$, $y=0$

vs:
→ Both can be made into expression

Scanned with CamScanner

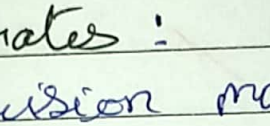
Date ___/___/_____

* Components of Boolean Algebra

1) Logic Gates
2) Boolean Expressions
3) Truth Tables

1) Logic Gates :

Are decision making circuits and building blocks of a circuit.

Eg → 

gate symbol (AND)

There are 3 types of logic gates :

(i) Basic : AND, OR, NOT
(ii) Derrived : XOR, XNOR, ~~NAND, NOR~~
(iii) Universal : NAND, NOR

→ Basic : These are elementary / primary gates & are used for making logic diagrams.

→ Derrived : Combination of two or more basic gates forms a derrived gate.

→ Universal : These are also derrived but they can perform the functions of all basic gates.

2) Boolean Expressions :

They consist of constants, variables with operating signs. There are 2 types of Boolean Expressions.

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

1- SOM-OF-~~PRO~~PRODUCT (sop Σ)

eg → $x = A \cdot B + C \cdot D$

2). ~~Product~~ PRODUCT-OF-SOM (pos Π)

eg → $x = (A+B) \cdot (C+D)$

• → Relation b/n Σ & Π

$$\Sigma = \Pi'$$
$$\Pi = \Sigma'$$

* One is Compliment of the other

3) ~~Truth~~ Truth Tables:

It is a form of table which gives all possible combinations of inputs and outputs, state / levels
(T/F) (0/1)

All possible combinations = 2^n ($n \in N$)

n is number of inputs

* Gray Coding

A	B	C	
0	0	0	= 0
0	0	1	= 1
0	1	0	= 2
0	1	1	= 3
1	0	0	= 4
1	0	1	= 5
1	1	0	= 6
1	1	1	= 7

A-Z	a-z	0-9	Blank
65-90	97-122	48-57	32

If we have no peace, it is bec...

Scanned with CamScanner

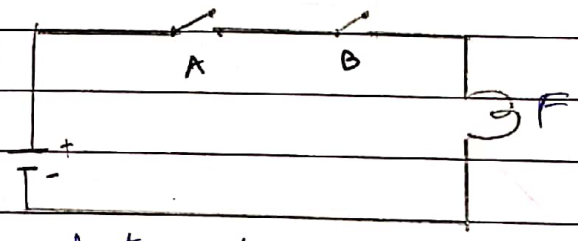
Date ___/___/_____ Page No.: _____

It is a form of coding which has a difference of only 1, between each row & column.

* Logic Gates :

→ Use → Performance → Symbol → Expression → Truth Table

→ AND gate :



electrical circuit

→ Use : Series Connection

→ Performance : Multiplication

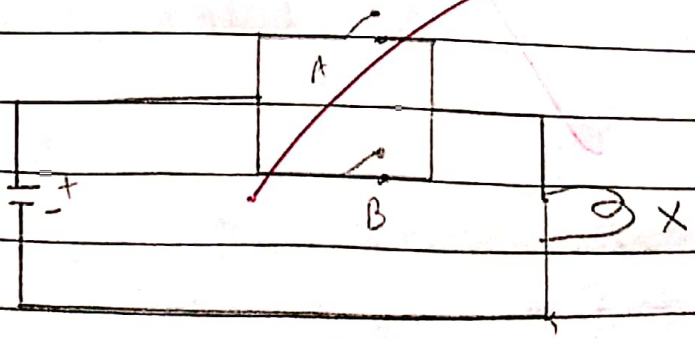
→ Symbol : A, B → AND → F {A (AND) B}

→ Expression : $x = A \cdot B$, $x = AB$, $x = A \cap B$, $x = A \wedge B$ (conjunction)

T.T.

A.	B.	X
0	0	0
0	1	0
1	0	0
1	1	1

→ OR gate : (Inclusive OR)



X	A
1	0
0	1

Scanned with CamScanner

Date ___/___/_____

→ Use : In ll connection

→ Performance : Addition

→ Symbol : A, B → OR → X

→ Expression : $X = A+B$

$X = A \cup B$

$X = A \vee B$

(disjunction)

→ T.T.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

→ NOT gate :

A → P → S → X

→ Use : In transformers, stabilizers

→ Performance : Subtraction / Negation

→ Symbol ~~A → ▷ → X~~ A → ▷ → X

(NOT) A

Comparator

→ Expression : ~~X~~ A'

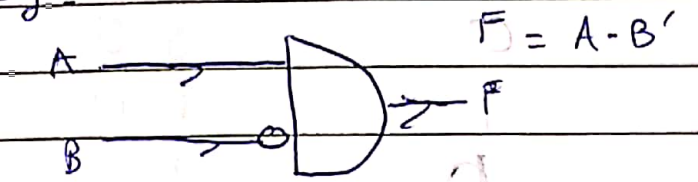
$\bar{A}$

~~~X~~ ~A

→ T.T

A	X
0	1
1	0

Scanned with CamScanner
~~~

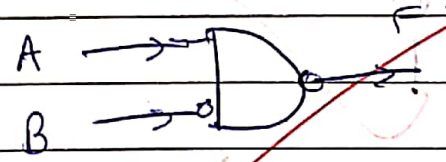
* Comparator :

It is a small circle which can negetiat any input or output at any stage of the circuit. It is used along with some gat

Eg :

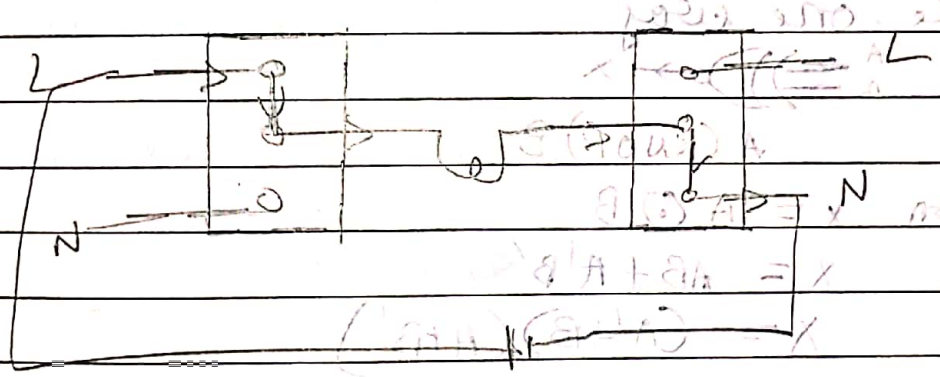


$F = A \cdot B'$



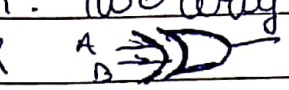
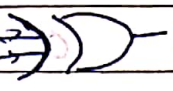
$F = (A \cdot B')'$

→ XOR gate (Exclusive OR)



→ Use : staircase, hall

→ Perform : Two way or dual

→ Symbol  × A (XOR) B 

→ Exp : $X = A \oplus B$

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

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

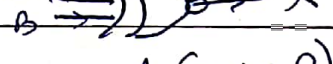
T.T

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

"Do not say, 'It is morning,' and dismiss it with a name of yesterday."
Scanned with CamScanner

for multiple inp, odd no of 1's = 1
even no of 1's = 0

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

→ XNOR gat (Exclusive NOT OR)
→ use: ~~XNOR~~ to find flow of current
→ Performance: one way
→ Symbol: 

A (XNOR) B

→ Expression $X = A \odot B$
$X = AB + A'B'$
$X = (A'+B)(A+B')$

T.T.

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

for multiple inputs, odd no. of 0's = 0
even no. of 0's = 1

~~for~~ for odd number of ~~inp~~ inp, XNOR = XOR
— even ——— XNOR ≠ XOR

Scanned with CamScanner

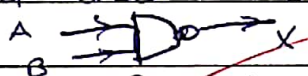
Date ___/___/_____ Page No.: _____

T.T.

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

→ NAND gate (NOT AND)

→ use & performance - same as basic gates

→ Symbol 

A (NAND) B

→ Expression $X = (A \cdot B)'$

$X = \overline{A \cdot B}$

$X = A' + B'$

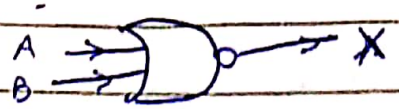
T.T

A	B	$(A \cdot B)'$
0	0	1
0	1	1
1	0	1
1	1	0

→ NOR gate

→ use & performance: same as basic

→ Symbol :



→ Exp: A (NOR) B

$X = (A+B)'$ $X = \overline{A+B}$ $X = A' \cdot B'$

"Do not say, 'It is morning' and dismiss it with a name of yesterday."

Scanned with CamScanner

Date __/__/__ Page No.:

Principle of Duality

To every boolean equation there exists another equation which is dual to the origonal equation. This is achieved by changing ~~AND's to~~ ANDs to ORs and vice versa, 0 to 1 & 1 to 0, however the compliment remains unchanged.

$$A' \cdot B + B'A + 1 = 1$$

$$(A' + B) \cdot (B' + A) \cdot 0 = 0$$

} dual.

Scanned with CamScanner

Date __/__/____ Page No.: _____

Laws

1. Identity law: $A \cdot A = A$, $A \cdot A' = 0$, $A \cdot 0 = 0$, $A \cdot 1 = A$

If a variable is anded or ored with its own identity.

$A \cdot A = A$
$A \cdot A' = 0$
$A \cdot 0 = 0$
$A \cdot 1 = A$

A	A	X	A	A'	X	A	0	X	A	1	X
0	0	0	0	1	0	0	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	1	1

$A + A = A$ $(A \cdot A = A)$, $\bar{\bar{A}} = A$
$A + A' = 1$ $(A \cdot A' = 0)$
$A + 0 = A$ $(A \cdot 1 = A)$
$A + 1 = 1$ $(A \cdot 0 = 0)$

A	A	X	A	A'	X	A	0	X	A	1	X
0	0	0	0	1	1	0	0	0	0	1	1
1	1	1	1	0	1	1	0	1	1	1	1

→ Zero's properties

$A \cdot 0 = 0$
$A + 0 = A$

→ One's properties

$A + 1 = 1$
$A \cdot 1 = A$

→ Compliment properties

$A + A' = 1$
$A \cdot A' = 0$

Binary [illegible]

"Do not say, 'it is morning,' and dismiss it with a name of yesterday."
— Rabindranath Tagore

Scanned with CamScanner

Date __/__/____

2. Commutative law : $A+B = B+A$ } Dual
$B \cdot A = A \cdot B$

3. Associative law : $A+(B+C) = (A+B)+C$ } Dual
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$

4. Distributive law : $A+(B \cdot C) = (A+B) \cdot (A+C)$ } Dual
$A \cdot (B+C) = A \cdot B + A \cdot C$

5. Absorbtion law : $A + A \cdot B = A$ } dual
(Redundant law) $A \cdot (A+B) = A$

6. ~~Idempotent~~ Idempotent law : $A+A = A$ } dual
$A \cdot A = A$

7. Involution law : $\bar{\bar{A}} = A$
$A'' = A$

8. De Morgan's law : $(A+B)' = A' \cdot B'$ } Dual
$(A \cdot B)' = A' + B'$

The complementation of a sum is equal to the product of their individual complements.

A	B	A+B	LHS $(A+B)'$	A'	B'	RHS $(A' \cdot B')$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

hence proved

$1 = A + A$

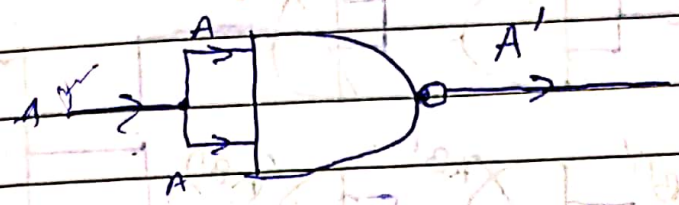
Scanned with CamScanner

Date __/__/__ Page No.: _____

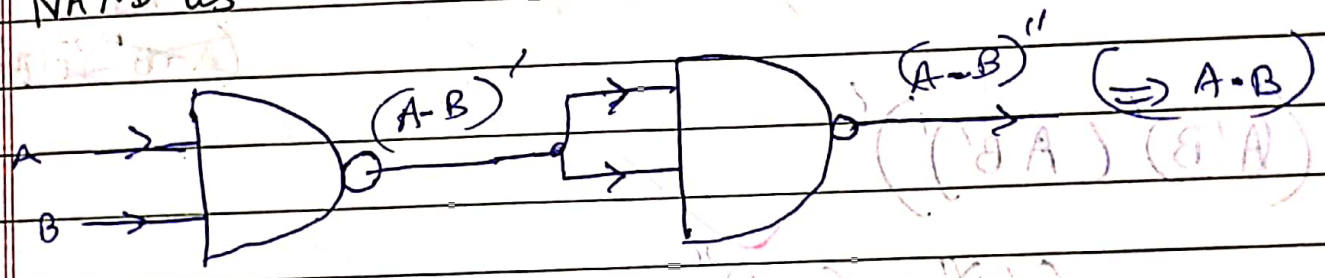
Universility of Gates

→ NAND :

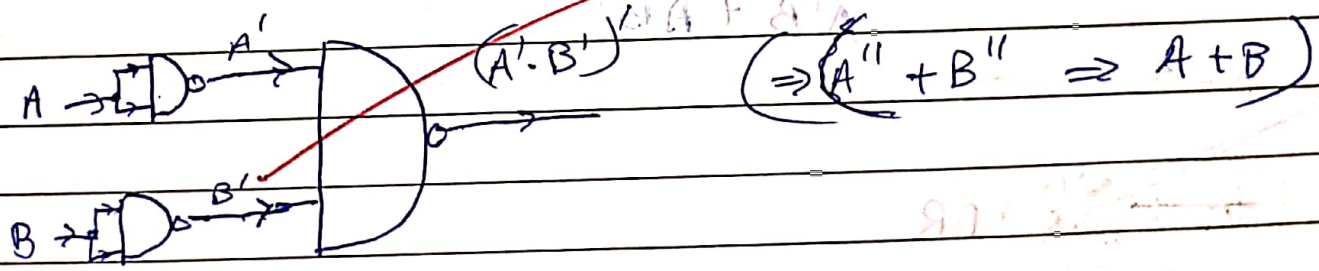
* NAND as NOT



* NAND as AND



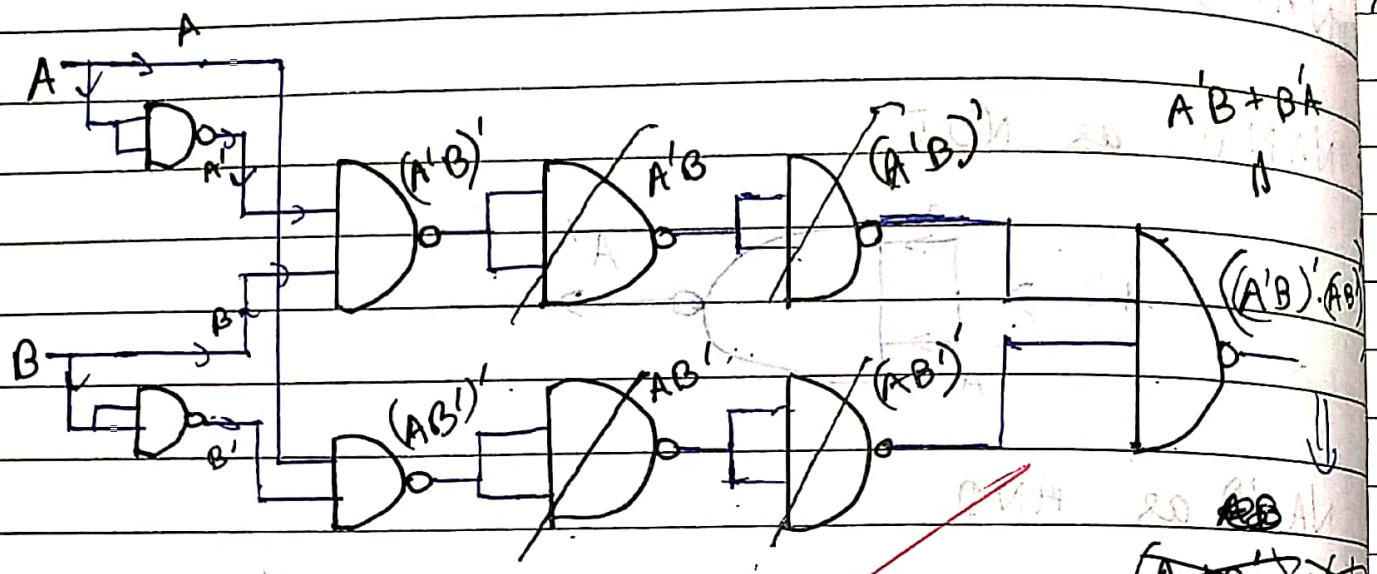
* NAND as OR



Scanned with CamScanner

Date ___/___/_____

*NAND as XOR



$((A'B)'(AB')')'$

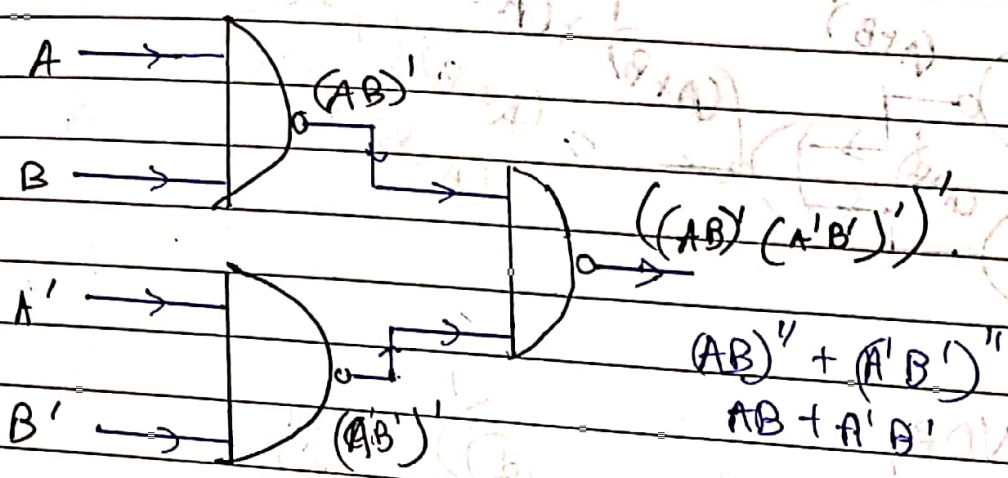
$(A'B)'' + (AB')''$

$A'B + AB'$

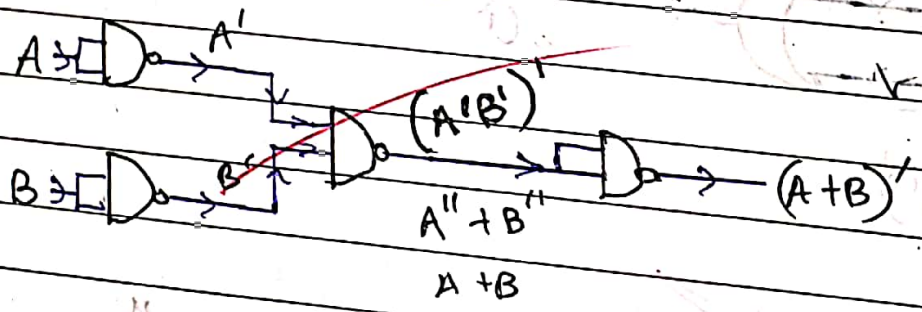
Scanned with CamScanner

Date ___/___/_____ Page No.: _____

* NAND as XNOR : $AB + A'B'$



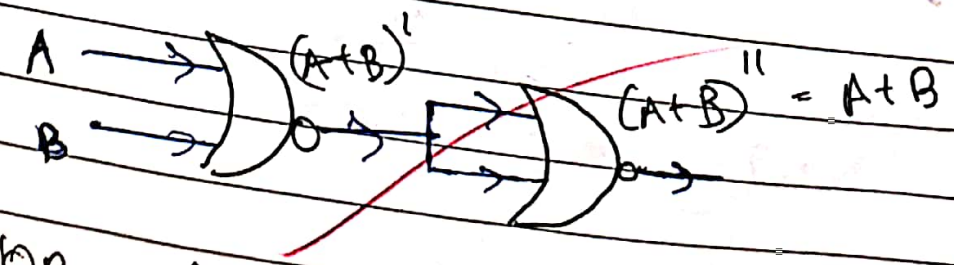
* NAND as NOR : $(A+B)'$



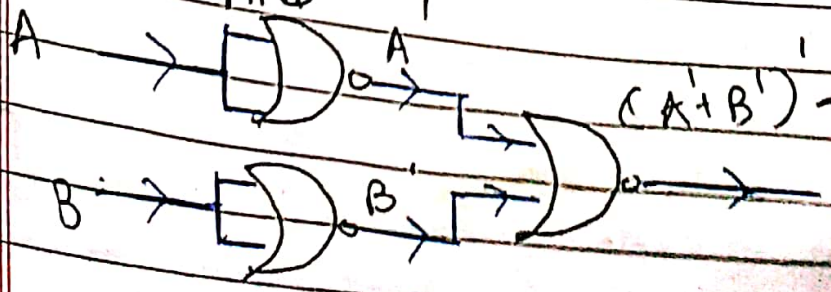
→ NOR

* NOR as NOT

* NOR as OR

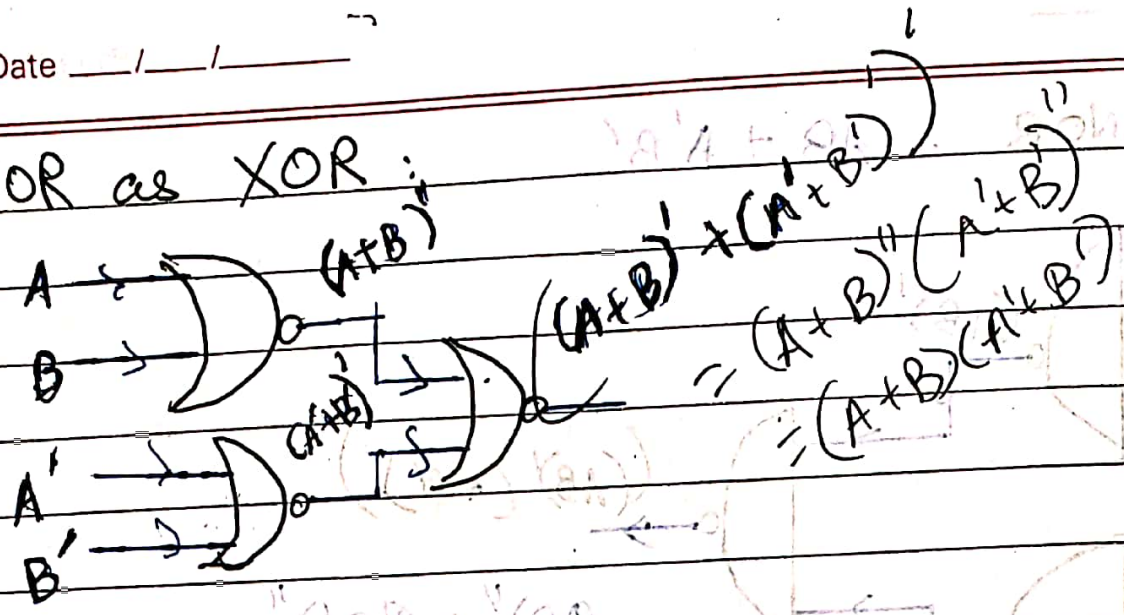


* NOR as AND

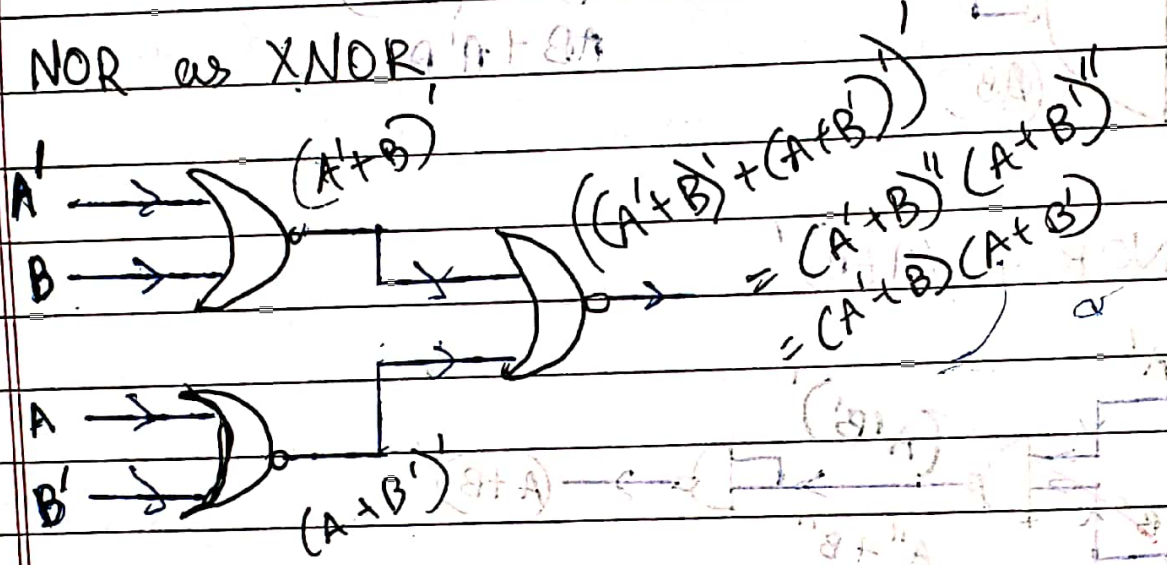


Scanned with CamScanner

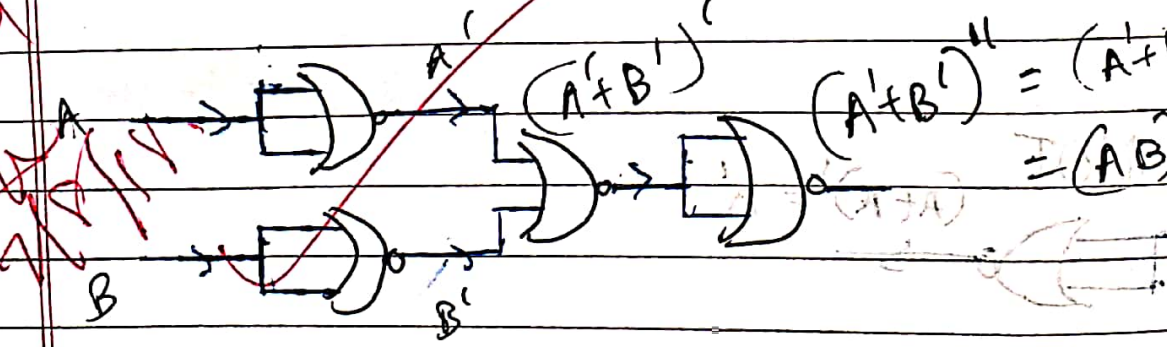
* NOR as XOR :



* NOR as XNOR



* NOR as NAND



Scanned with CamScanner

Date __/__/____ Page No.: _____

Inter Conversions

IT exp ai.

SOP (Σ) AND'S rule

1st rule

A	B	C	X	
0	0	0	0	
0	0	0	1	$A'B'C$
0	0	1	0	
0	0	1	1	$A'BC$
1	0	0	1	$AB'C'$
1	1	0	1	
1	0	1	0	
1	0	1	1	ABC'
1	0	1	0	

min terms (m)

$X(A,B,C) = A'B'C + A'BC + AB'C' + ABC'$ (canonical)

$X(A,B,C) = \Sigma(1,3,4,6)$ (Cardinal)

$F(P,Q,R) = P'R + QR'$ → Cardinal

$= P'R(Q+Q') + QR'(P+P')$

$= P'RQ + P'RQ' + QR'P + QR'P'$

$F(P,Q,R) = \Sigma(0,2,4,7)$ → Cardinal

000 010 100 111

$= P'Q'R' + P'QR' + PQ'R' + PQR$

$= (m_0 + m_2 + m_4 + m_7)$

Scanned with CamScanner

POS (π)

A	B	C	X		POS (π)
0	0	0	1		OR's rule
0	0	1	0	$A+B+C'$	SUM'S rule
0	1	0	0	$A+B'+C$	0's rule
0	1	1	1		
1	0	0	0	$A'+B+C$	
1	0	1	1		
1	1	0	1		
1	1	1	0	$A'+B'+C'$	

max terms (M)

$$X(A,B,C) = (A+B+C')(A+B'+C)(A'+B+C)(A'+B'+C')$$

(Canonical)

Max terms are the sum of all its literals
Min terms are the product of all its literals

$$X(A,B,C) = \pi(1,2,4,7)$$ (Cardinal)

$$X(P,Q,R) = (M_1 \cdot M_2 \cdot M_4 \cdot M_7)$$

Canonical → Sum of min / product of max terms
Cardinal → Decimal equivalents of its terms

$$F(P,Q,R) = (P+Q)(P'+R)$$

$$= (P+Q+0)(P'+R+0)$$

$$= (P+Q+R\cdot R')(P'+R+Q\cdot Q')$$

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

$(P+Q+R)(P+Q+R')(P'+R+Q)(P'+R+Q')$

$F(A,B,C) = \Pi(1,3,4,6)$

001 011 100 110

$\cancel{(A'+B'+C)(A'+B'+C)C}$

$(A+B+C')(A+B'+C')(A'+B+C)(A'+B'+C)$

Scanned with CamScanner

Date __/__/__ Page No.:

Applications of logic gates

→ Half Adder
→ Full Adder
→ Encoders
→ Decoders
→ Multiplexers

use, performance, symbol, exp, TT, Def: It is a combinational Circuit which...

* HALF ADDERs:

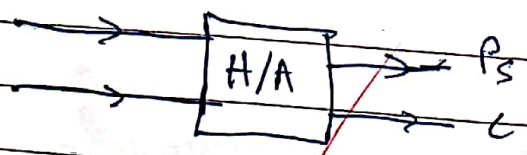
```
 0 0 0 1 1
 1 0 1 1 1
 0 1 0 1 1
 1 1 1 0 0
```

→ **Def:** ... It is a combinational circuit which adds 2 input binary bits & outputs 2 Binary bits.

→ **Use:** Add 2 bits

→ **Performance:** partial addition

→ **Symbol:**



→ T.T.

A	B	~~AB~~	Ps	C
0	0	~~0~~	0	0
0	1	~~1~~	1	0
1	0	~~1~~	1	0
1	1	~~1~~	0	1

$P_S = A'B + AB'$

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

$= \Sigma(1,2)$

$= \Pi(0,3)$

Scanned with CamScanner

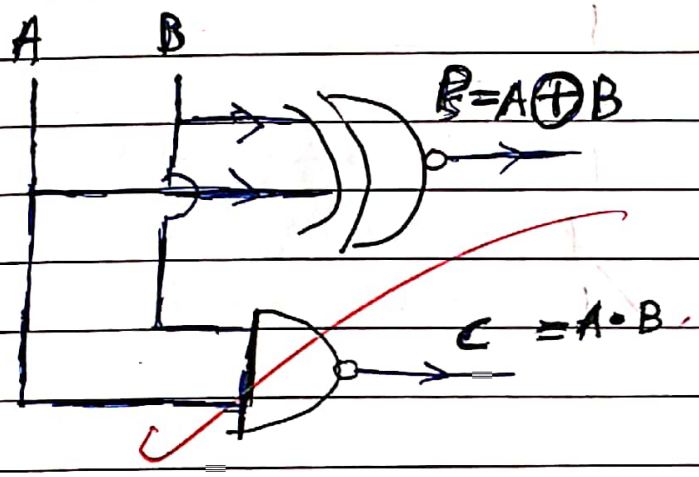
Date ___/___/___

$C = A \cdot B$

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

$C = \Sigma(3)$

$C = \Pi(0,1,2)$



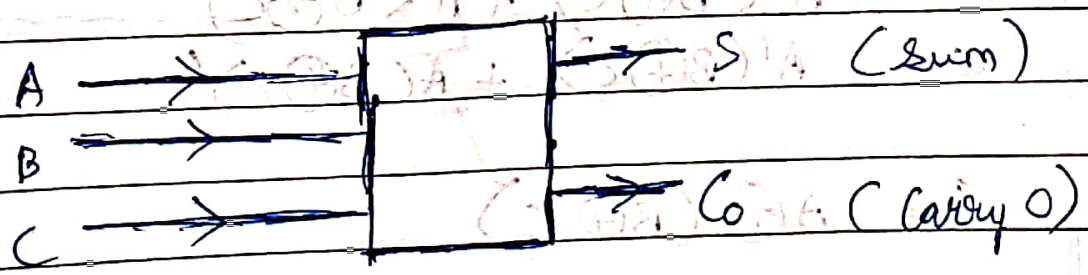
* FULL ADDERS :-

→ Def: ... It is a combinational circuit which adds 3 input binary bits & outputs 2 binary bits.

→ ~~Df~~. Use : Add 3 inputs

→ Performance : Total addition

→ Symbol :



Scanned with CamScanner

Date

→ T.T.

A	B	C	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

→ Exp:

$S = A'B'C + A'BC' + AB'C' + ABC$

$S = A \oplus B \oplus C$

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

$S = \Sigma(1,2,4,7)$

$S = \Pi(0,3,5,6)$

$S = A'(B'C+BC') + A(B'C'+BC)$

$S = A'(B \oplus C) + A(B \odot C)$

$S = A'(B \oplus C) + A(B \oplus C)'$

$S = A \oplus (B \oplus C)$

Scanned with CamScanner

Date ___/___/___ Page No.: _____

$$C_o = A'BC + AB'C + ABC' + ABC$$

$$* \; C_o = AB + BC + AC$$

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

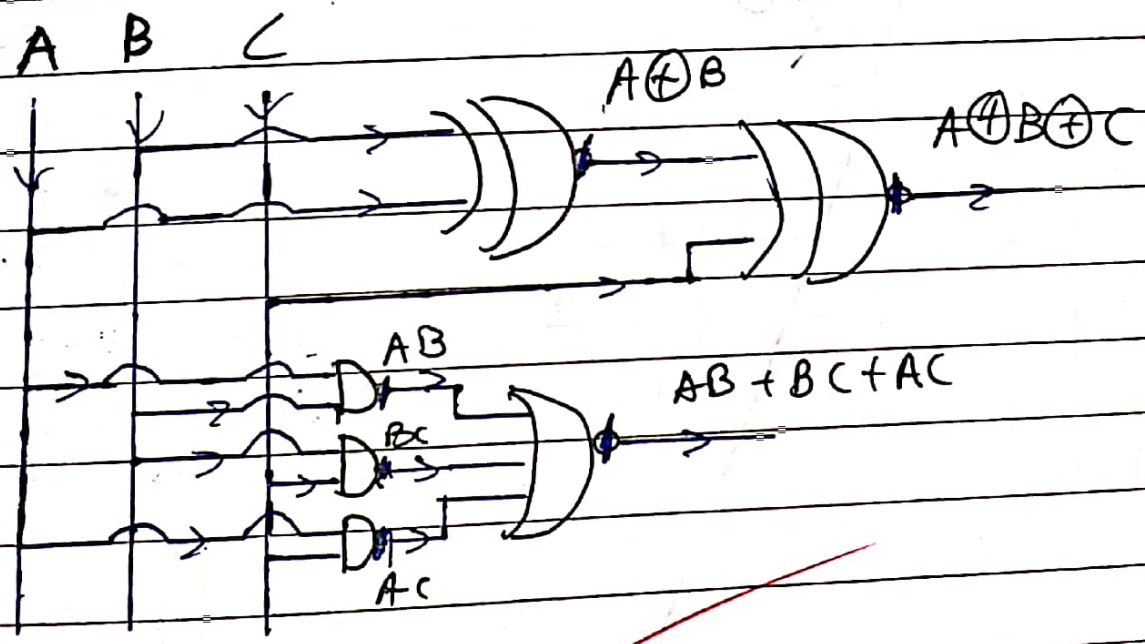
$$C_o = \sum (3,5,6,7)$$

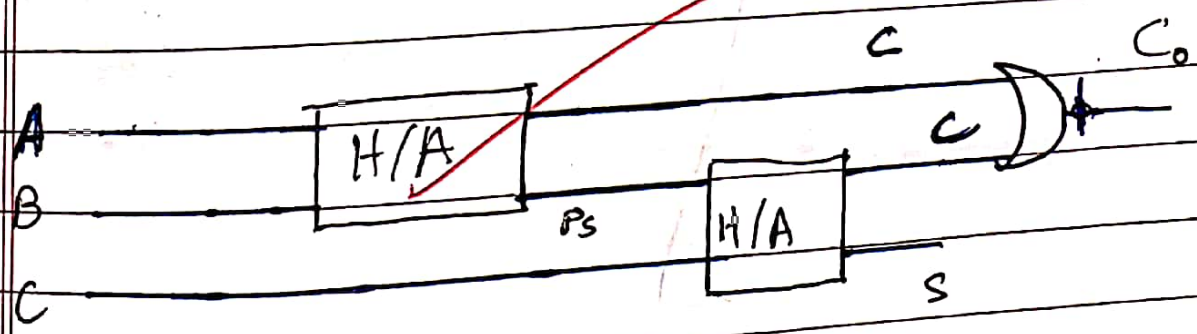
$$= \prod (0,1,2,4)$$

$$C_o = A'BC + ABC + AB'C + ABC + ABC' + ABC$$

$$= BC(A'+A) + AC(B'+B) + AB(C'+C)$$

$$= BC + AC + AB$$





Scanned with CamScanner

Date ___/___/___

Page No.: ___

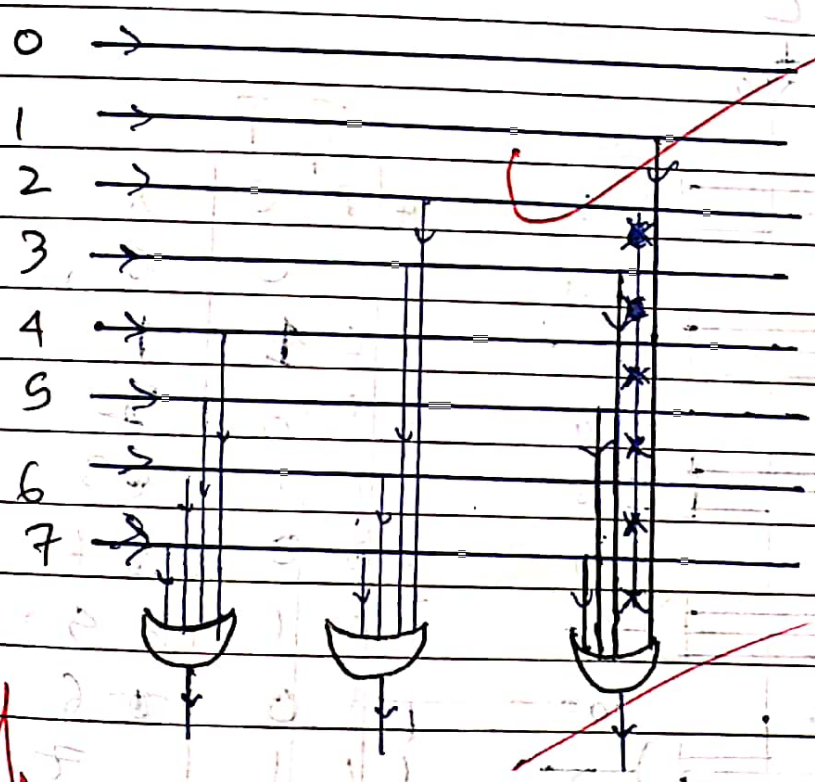
* Encoders :

→ Def: ... inputs 2^n or fewer lines & outputs 'n' lines

→ Use: HLL → LLL

→ Circuit Diagram:

* 8 → 3 (Octal Encoder)



I	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

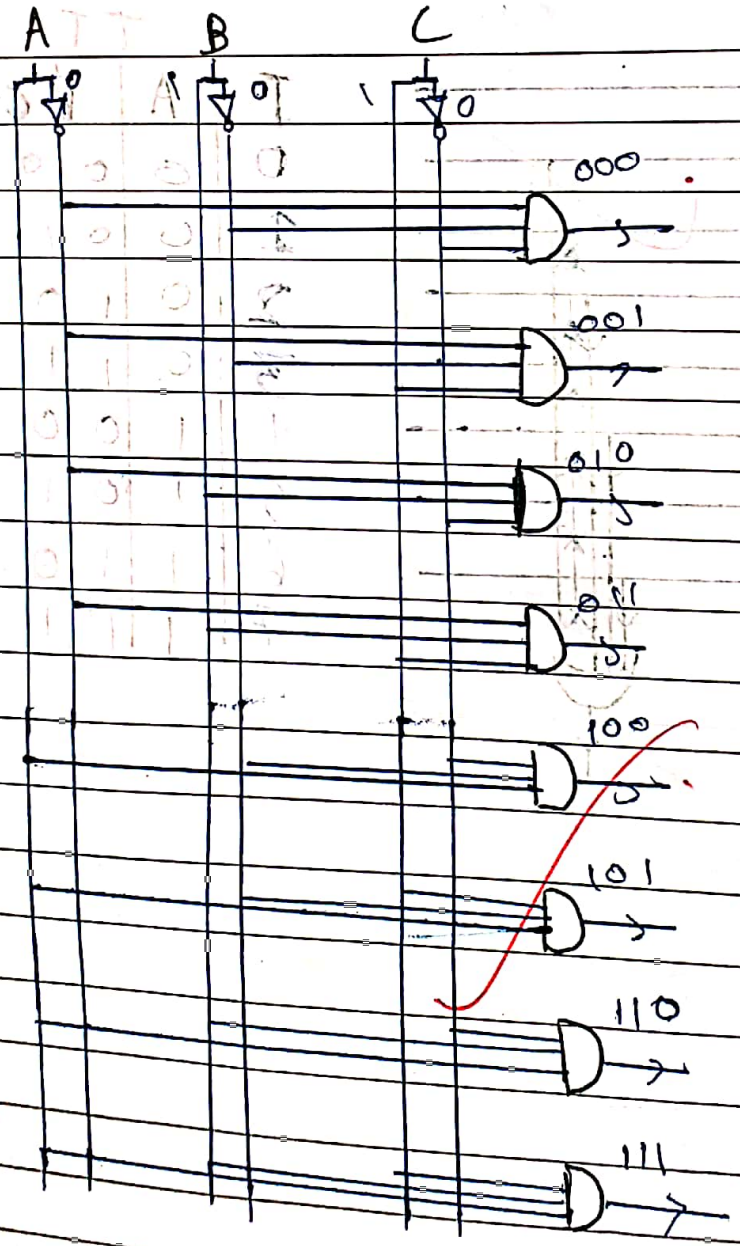
Scanned with CamScanner

* Decoder :

→ Definition : ... which inputs 'n' lines & outputs lines

→ use : LLL → HLL, also used in making of Multiplexer

3 to 8 (octal Decoder)



A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Scanned with CamScanner

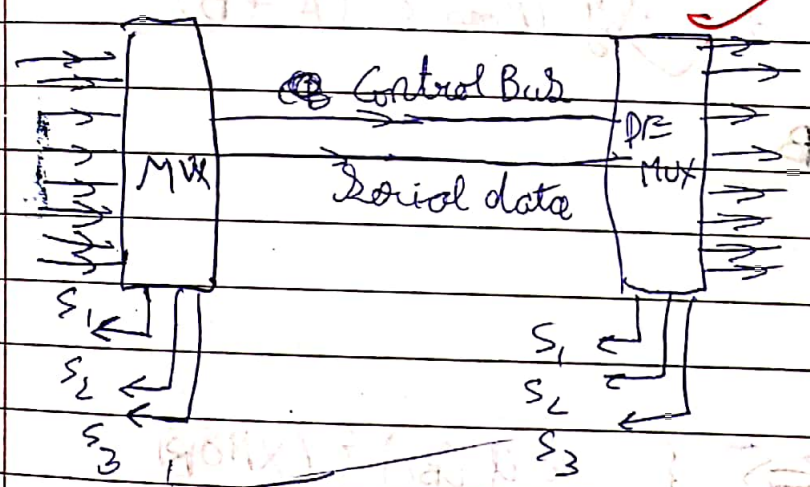
* Multiplexer *

→ Def: ... which inputs ll data and outputs 1 serial data. AKA Data selector, mixer, many to one, ll to serial converter etc.

→ Use: In data transmission

→ Performance: ll to serial

* Block diagram

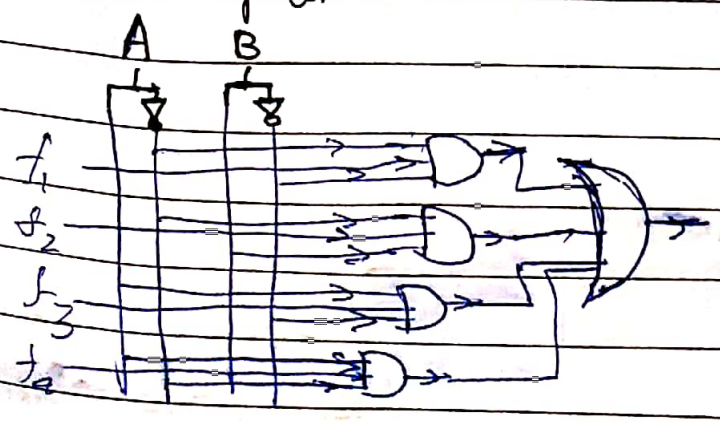


Select lines

Selector Distributor

{Decoder + OR}

* Circuit Diagram



(4:1 MUX)

Scanned with CamScanner

Date ___/___/___

Propositional logic

Page No.:

"A proposition is a statement which can have only 2 possible answers i.e true or false."

* Well formed formulae (WFF):
If a proposition is satisfiable or valid, then it is known as WFF.

* Conjunction ($\wedge$) {$\cap$ & $\cdot$} (AND)
* Disjunction ($\vee$) {$\cup$ & $+$} (OR)
* Negation ($\sim$) {$\sim$ & '$\bar{}$'} (NOT)
* Implication ($\Rightarrow$) {$\therefore$ & if (Mono)} ($A'+B$)

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

* Bi-Conditional $\Leftrightarrow$ { & if (Bi)} (XNOR)

A	B	$A \Leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

A	B	$A \Rightarrow B$	$B \Rightarrow A$	$A \Leftrightarrow B$	$(A \Rightarrow B)(B \Rightarrow A)$
0	0	1	1	1	
0	1	1	0	0	
1	0	0	1	0	
1	1	1	1	1	

$A \Leftrightarrow B = (A \Rightarrow B) \wedge (B \Rightarrow A)$
WFF

* Tautology : All 1s in a proposition output
* Contradiction : All 0s in a proposition output
* Contingency : mixed 1s & 0s in a proposition output

* if $A \Rightarrow B$
  A ~ Has Comp.
  B ~ use E-mail

→ Converse : $B \Rightarrow A$
→ Inverse : $\sim A \Rightarrow \sim B$
→ Contrapositive : $\sim B \Rightarrow \sim A$
→ Chain rule : $A \Rightarrow B$ & $B \Rightarrow C$ then $A \Rightarrow C$
(AKA Modus Ponen)

Scanned with CamScanner

Date __/__/__ Page No

K-Maps

Karnaugh's Maps

It is a form of Mapping which reduces a boolean expression to its simplest form. The expression to be simplified must be in canonical / cardinal form

→ Drawing a K-Map
→ filling
→ grouping (Octal, Quad & pair)
→ Reducing the groups
→ Final Reduced Expression

* Drawing:

→ Sop (Σ)

→ F(A, B) - (2 var)

A \ B	B'	B
A'	0	1
A	2	3

B \ A	A'	A
B'	0	2
B	1	3

→ F(A, B, C) - (3 var)

A \ BC	B'C'	B'C	BC	BC'
A'	0	1	3	2
A	4	5	7	6

AB \ C	C'	C
A'B'	0	1
A'B	2	3
AB	6	7
AB'	4	5

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

→ $F(A,B,C,D)$ - (4 var)

AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$	1 (0)	1 (1)	0 (3)	0 (2)
$A'B$	1 (4)	0 (5)	0 (7)	1 (6)
AB	1 (12)	0 (13)	0 (15)	0 (14)
AB'	1 (8)	0 (9)	1 (11)	1 (10)

* Filling

→ Sop (Σ)

$F(A,B,C,D) = \Sigma(0,1,4,6,8,10,11,12)$

* Grouping

→ Octal (8 ones) - It will eliminate 3 var

→ 1 1 1 1 → 1 1 1 1 → 1 1 1 1 1 1 1 1

→ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

→ Quad - It eliminates 2 var

→ 1 1 1 1 → 1 1 1 1 → 1 1 1 1 → 1 1 1 1 → 1 1

→ 1 1 1 1 1 1

"No real change in history has ever been achieved by discussions." –Subhash chandra bose

Scanned with CamScanner

→ Pair → eliminates 1 var.

→ (1 1)

→ (1 1)

→ 1 1

→ 1

* Examples

→ $F(P,Q,R,S) = \Sigma(0,2,6,7,8,10,14,15)$

PQ \ RS	R'S'	R'S	RS	RS'
P'Q'	1 0	0 1	0 3	1 2
P'Q	0 4	0 5	1 7	1 6
PQ	0 12	0 13	1 15	1 14
PQ'	1 8	0 9	0 11	1 10

$Q_1 = (m_0 + m_2 + m_8 + m_{10}) = Q'S'$

$Q_2 = (m_6 + m_7 + m_{14} + m_{15}) = QR$

$F(P,Q,R,S) = Q'S' + QR$

→ $F(A,B,C,D) = (0,1,2,3,6,7,11,14,15)$

AB \ CD	C'D'	C'D	CD	CD'
A'B'	1 0	1 1	1 3	1 2
A'B	0 4	0 5	1 7	1 6
AB	0 12	0 13	1 15	1 14
AB'	0 8	0 9	1 11	0 10

$Q_1 =$

$Q_2 =$

$Q_3 =$

$F(A,B,C,D) = A'B' + B(A+C)$

"manifest the divinity of the right of everyone." —Sri Ramakrishna Paramahamsa

Scanned with CamScanner

Date ___/___/___ Pa

→ $F(A,B,C,D) = \Sigma(1, 3, 7, 8, 9, 10, 11, 12, 14)$

AB \ CD	$C'D'$	$C'D$	CD	CD'
$A'B'$	0 0	1 1	1 3	0 2
$A'B$	0 4	0 5	1 7	0 6
AB	1 12	0 13	0 15	1 14
AB'	1 8	1 9	1 11	1 10

$Q_1 =$

$Q_2 =$

$P_1 =$

$F(A,B,C,D) = B'D + AD' + A'CD$

A \ BC	$B'C'$	$B'C$	BC	BC'
A'	1	1	1	1
A	1	1	1	1

$F(A,B,C) = 1$

→ POS (T)

→ $F(A,B)$ - 2-var

A \ B	B	B'
A		
A'		

Scanned with CamScanner

Date ___/___/___

→ F(A,B,C) - 3-var

A \ BC	B+C	B+C'	B'+C'	B'+C
A	0	1	3	2
A'	4	5	7	6

→ F(A,B,C,D)

AB \ CD	C+D	C+D'	C'+D'	C'+D
A+B	0	1	3	2
A+B'	4	5	7	6
A'+B'	12	13	15	14
A'+B	8	9	11	10

* 0s are grouped & filled not 1s

1	0	0	1
0	0	0	0
1	0	1	1
1	0	0	1

Scanned with CamScanner

0	1	0	0
1	1	0	0
0	1	0	0
0	1	0	0

Scanned with CamScanner

Date __/__/____ Page No.:

Complexity

Complexity: It is a measurement/ growth rate of an algorithm.

→ Big 'O' Notation: It is the Unit of measurement of an algorith w.r.t. time & space factors

* Types of Complexities:

→ Best Case Complexity: least no. of executions required for an algo.

→ Worst Case Complexity: max no. of executions required for an algo.

→ Avg Case Complexity: It is neither min nor max no. of executions required for an algorithm

Eg → Simple Program:

```
Statement-1
S-2
S-3
:
S-4
```

O(1) (B)
O(n) (W)

Eg → Conditional Program

```
if(<>){
  //block
}
```

O(n) (W) O(1)(B)

```
elseif(<>){
  //block
}
```

O(n) O(1)(B)

Scanned with CamScanner

Date __/__/____ Page No.: _____

```
else {
  //body
}
```
$O(n)$ (W) $O(1)$ (B)

Eg → Nested if :

```
if (—) {
    if (—) {
        //body
    }
    //body
}
```
$O(n)$ (W)
$O(1)$ (B)

Eg looping :

```
for (int i = 1; i <= n, i++) {
    //body
}
```
$O(n)$

```
for (int j = 1, j <= m, j++) {
    //body
}
```
$O(m)$

Total : $O(n+m)$ or $O(n) + O(m)$

if $m = n$, complexity $= O(n)$

*: Units : Big 'O', Big 'Ω', Big 'Θ' (Big 'Oho')

Hardware

Software

Scanned with CamScanner

Date __/__/____

Eg Nested loops :

```
for (int i=1; i<=n ; i++){
    for (int j=1, j<=m; j++){
        //body
    }
    //body
}
```

$O(nm)$

if $m=n$, $O(n^2)$

```
for (int k=1; k<=m; k++){
    //body
}
```

$O(n^2+m)$

if $m=n$,

$O(n^2+n)$

$O(n^2)$

* dominent term is always taken : Dom term is highest order variable

$O(\log n)$ is used for fraction

Eg→ fractional is only for Binary Search

Scanned with CamScanner

Date __/__/__ Page No.: _____

Infix to Postfix & Prefix

→ Postfix:

A* (B+C) +D/(E+F)

A* BC+ +D/ EF+

ABC+* + DEF+/

ABC+* DEF+/+

A*B/C

AB*/C

AB*C/

→ Prefix:

A* +BC +D/ +EF

*A+BC +/D+EF

+*A+BC/D+EF

→ Postfix:

A* (B+C)/E + (F-G)

A* BC+ /E + FG-

ABC+* /E + FG-

ABC+*E / + FG-

ABC+*E/ FG-+

(A*B-C)/(F/(F+F))

(AB*-C)/(F/EF+)

AB*C- / FEF+/

AB*C-FEF+//

Scanned with CamScanner

Date __/__/__

Address Calculation

→ Row : Horizontal
→ Column : Vertical
→ b_r : row index at Base
→ b_c : Column index at Base
→ i
→ j
→ Width : Space/Memory

Columns → 0 1 2 3 4 5

Rows →

0						
1						
2						
3				ij		
4						
5						

* Row Major

Base = b_r
row : ith
no. of rows before ith row : $i - b_r$

$$ij = [(i - b_r)c + j - b_c]\,w + BA$$

* Column Major

$$ij = [(j - b_c)r + i - b_r]\,w + BA$$

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

Eg1> Arr[-4 → 7, 3 → 8] requires 4 bytes of storage for each cell. If BA = 1024, Calc. Adress at Arr[5][6] row majorwise

$i,j = 1024 + 4[(5+4)6 + 6 - 3]$

$= 1024 + 4(54+3)$

$= 1024 + 4 \times 57$

$= 1252$

Eg2> A[M][M'] is stored in memory, W = 4B if BA for A[1][1] is 1500 & adress at A[4][5] is 1608 find m, column majorwise.

$1608 = 1500 + 4[4m + 3]$

$108 = 4(4m+3)$

$27 = 4m+3$

$24 = 4m$

$m = 6$

Scanned with CamScanner

Date __/__/__ # Trees

Page No.:

* Tree: A tree is a subset containing nodes or edges such that one of the nodes is designated as root

* Types of Trees:

→ 'A Tree': Is a type of tree in which a node can have any number of daughter / subnodes.

→ Binary Tree: Is a type of tree where each node may have at most two daughter / sub nodes. ~~no 2 nodes are identical~~

→ 2-Tree: Is a type of tree where every node has two daughter nodes.

→ All 2-Trees are Binary trees but not all Binary trees are 2-Trees

→ Free-Tree: Is a type of tree in which roots cannot be distinguished

* Characteristics of a tree:

→ It is acyclic or irreversible

→ No two nodes can be identical.

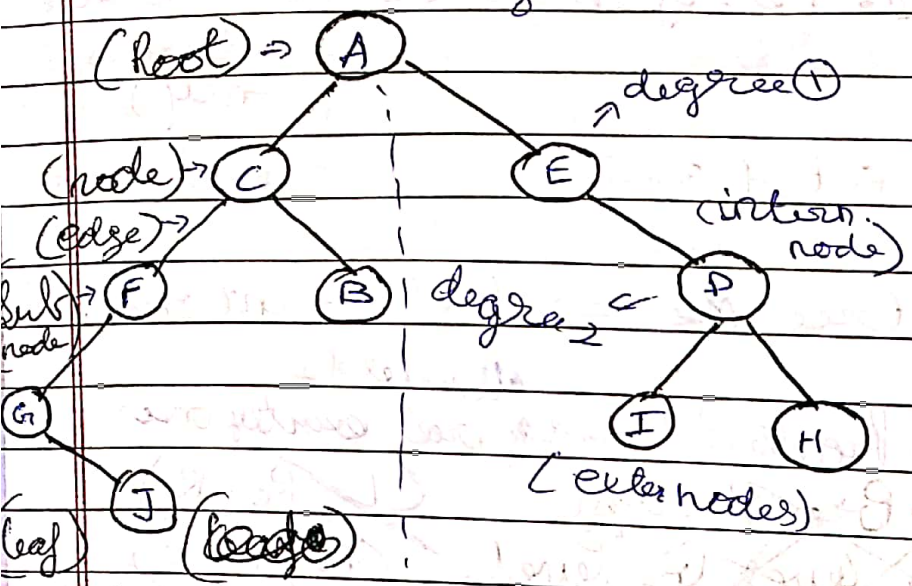
→ There is only one unique path b/n 2 nodes

→ It is a recursive data structure

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

Size = 10



level 0 {generat

level 1

level 2

level 3

level 4

Left subtree (size 5) Right subtree (size 4)

Nodes which have subnodes & parent nodes → Internal nodes.

Nodes which have no subnodes are called external nodes (aka leaf)

A root node is neither external nor Intern

Path b/n A & H

A - E - D - H

[height of D = 1 depth = 2]

Path length = 3 (no. of edges)

* Height of a tree : The longest path length b/n the root & the leaf.
(height = 4)

* Depth of a tree : Same as height for tree but diff. from nodes.

height of leaves = 0

depth of root = 0

Scanned with CamScanner

Date ___/___/_____ Page No.:

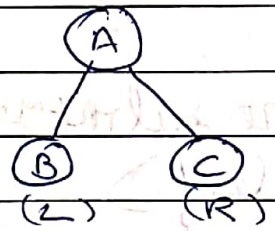
* Degree of a node: no. of subnodes. (immediate subnodes only)
leaf node = 0.

* Size = internal + ext. + root.

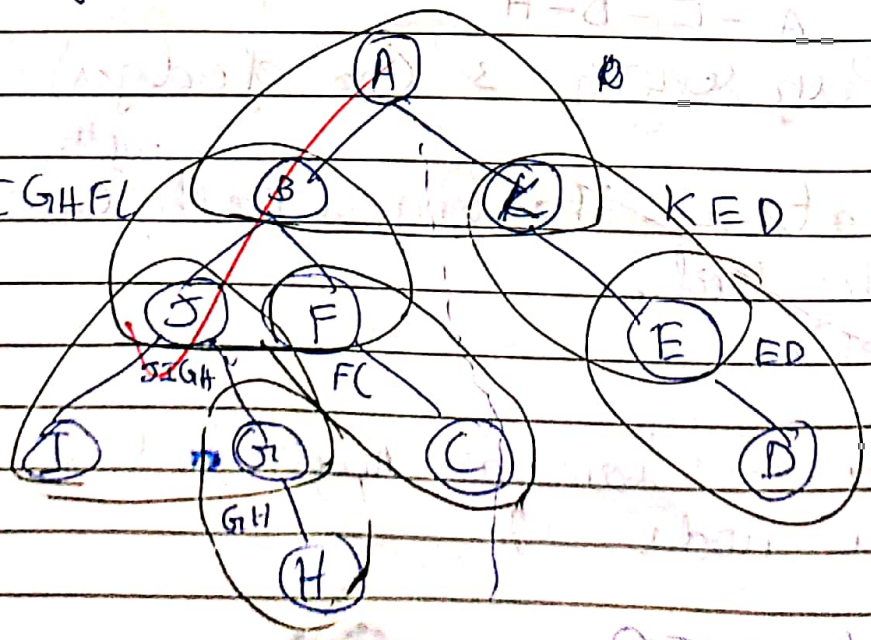
Size of left sub-tree ⇒ don't count root.

* Tree traversal: Process of visiting all nodes of a tree exactly once
→ In order (LRR₀) → ~~Pre order~~ (L, R₀, R)
→ post order (LRR₀) → ~~level by level.~~ (R₀, L, R)
→ Pre order (R₀LR) (R₀, L, R)
→ level by level (aka breadth first)

depth first.



Pre order = ABC
in order = BAC
Post order BCA
level by level ABC



Preorder: ABJI GHFLKED

Scanned with CamScanner

Date __/__/____ Java Page No.: _____

* Escape sequences : To print non graphical chars & key chars on screen.

A few eg. →

\n → new line (Enter)

\t → H-tab

\\ → backslash

\" → double quotes

\' → single quotes.

Eg →

1. Sopln("Don't"); ✗

Sopln("Don\'t"); ✓

2. Sopln("she said to him "eat your food""); ✗

Sopln("She said to him\"eat your food\""); ✓

Scanned with CamScanner

Date ___/___/___ Page No

* Data Types:

Inb. → Primitive Data types: one adress, one value
eg → int, char, etc (~~*~~ X String)

User Def. → Non-Primitive Data types: one address, many values
eg → int[], String etc. {but type is same}
access → a[0], a[1].

→ Class: one adress, many different types of values
(reference data types)
eg → (class) → object)
access → a.x, b.y
(Implicit / explicit)

Java

(Re)

Char sets
- Source - (Graphical, keyboard)
- Escape Sequences - (non-graphical)
  - "\n" → Enter
  - "\t" → 8 spaces (Horizontal tab)
  - "\\" → Backslash
  - "\"" → double quote
  - "\'" → Single quote

  } graphical but still special char.

They are not 2 char but only 1 char.

* Tokens: are the smallest individual parts of a java program.

→ Identifiers
→ literals
→ Keywords
→ Operators

Special char: (These are NOT token)
→ Terminators (;)
→ Seperators (, & ⊬ , \t)
→ punctuators ([{ " (:)

{ } — Encapsulation (Class)

→ Identifiers - variable names

→ May only contain: A-Z | $ | _ | 0-9 ← not start w/

Can start w/

→ Case SeNsITIVE. (if x ff ✓)

(reserved words → [illegible])

Scanned with CamScanner

→ Literals: Constants (data)

- Numeric Literals (0-9)
- Char literals
- String Literals
- Boolean literals (True & False)
- Byte literals (0-127) (→ Boolean) misuse
- null literals (nulls)

BCDI & EBCDIC

~~→ Extended Binary Coded Decimal~~

numeric < floating point (7.0) Real
Integer (7)

char c = 'B';

c → actually Numeric in nature. int a = 'B';
66 a = 66

But, S.o.pln(c); → B. S.o.pln(a); → a

7 → numeric
'7' → char
"7" → String

→ Keywords: words which have a special meaning in program

→ operators: All operators are signs & symbols.
a + b
↑ operator ← operand

Scanned with CamScanner

Date __/__/____ Page No.: _____

(only operators w/o a symbol ⇒ new & this)

→ unary operator: one operand. a++, b--

→ Binary operator: 2 operands. a+b, a-b, a/b etc.

→ Ternary operator: 3 operands.

```
max = (a>b) ? a : b ;
```

↑ one operand ↑ 2 operands.

⇕

```
if(a>b)
    max = a ;
else
    max = b;
```

```
max = (a>b) ? (a>c) ? a : c : (b>c) ? b : c
```

⇕

```
if (a>b) {
    if (a>c)
        max = a
    else
        max = c
}
else if (b>c)
    max = b
else
    max = c
```

⇕

```
if (a>b) {
    if (a>c)
        max = a;
    else
        max = c;
}
else if (b>c) {
    max = b;
}
else
    max = c;
```

Scanned with CamScanner

Date __/__/____ Page No.:

* Types of operators:

→ Arithmetic: +, -, *, /, %

→ Relational: >, <, >=, <=, !=, ==

→ Logical: (!, &&, ||)

→ Assignment: (=) LHS → var ← & → RHS → " / Constant. LHS → var/expn

→ Increment / Decrement: (++ / --)

Postfix / Prefix

use then do / do then use

→ Shorthand:

c = c+1; => c+=1; (Counter)

x = x*1; => x*=1; (How many)

c=c+x; => c+=x; (accumulator) (How much)

+=, -=, *=, /=, %=

* Wrapper Classes

Classes which define primitive data types.

Data type	Wrapper Class
byte	Byte *
short	Short
int	Integer
float	Float
long	Long
double	Double
char	Character.
(boolean)	Boolean

* Extended Binary Coded Decimal Code for Decimal Interchange

Scanned with CamScanner

Date __/__/____ Page No.: _____

Data Types

a → Identifier

[10] → Literal

→ Data Type

→ Inbuilt (one adress / one Value) a [10] (eg glass)

→ Array (one adress / many values of n types) a [10|20|5|19] 0 1 2 3
(eg tray of glass)

→ Class → (one adress / many values of ~~n~~ types)

a [76 05] "Index" ['A'] 0 (eg full meal)

* Implicit: done by the program auto when lower order data → Higher order data. (4 → 8 bit)

* Explicit: done by the programer specifically when Higher order → lower order data (8 → 4 bit)

Character (char) → 2 bytes

→ Static Memory → Memory opened during compilation before execution. int a = 10;

→ Dynamic Memory → Memory used only on execution. "new" Keyword required. (BETTER!)

Integer a = new Integer(10);
capital Cap

* Boxing: passing of static into dynamic: int b = 10;
Integer a = new Integer(b);

* Unboxing dynamic to static: Integer a = new Integer(10);
int c = a;

Scanned with CamScanner

Date ___/___/___

* Java Statements:

→ Input
→ Output
→ Conditional
→ loop
→ Jump / Branch

- ~~Input : 3 ways :~~
~~→ before exe : initialization~~
~~→ At the time of exe : Buffer / Scanner~~
→ during exe

- Input : 3 ways :
→ before exe : initialization
→ At the time of exe : parameters
→ during exe : Input Stream

→ Initialization :
int a = 5 ;
int b = 10;

→ Parameters :
public static int sum (int a, int b) {
}

→ Input Stream : *.io ; *.scanner ;

- Drawback : alphanumeric only.
Convert from String to datatype.

java.lang ; => Only pkg imported by default.

Scanned with CamScanner

Date __/__/__ Page No.:

Classes and Objects

* Functions: a program capable of performing a specific task.

→ Polymorphism: Function overloading

→ Syntax:

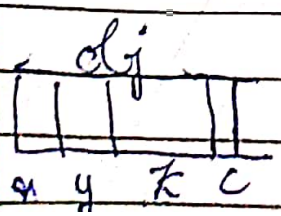
<access specifier> <return type> <name>(<Parameter>) {
    // Body
}

→ A function may/maynot have <parameters>. It may or maynot have a <return type>. Incase no return type, void is used.

* Classes: Collection of data & functions.

→ Keyword used to define non primitive data type: class.

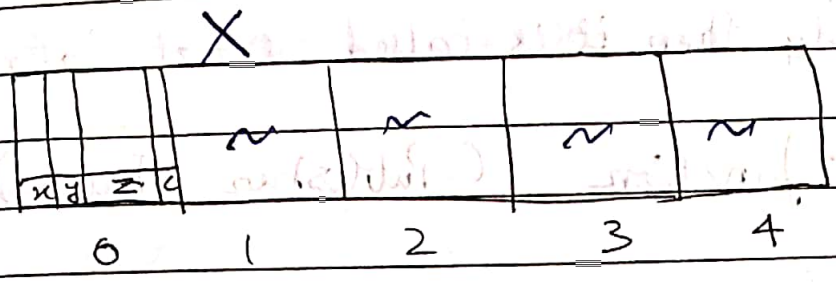
```
class Abc {
    int x, y;
    double z;
    char c;
}
Abc obj = new Abc();
```



Scanned with CamScanner

Date __/__/____ Page No.: _____

Abc x[] = new Abc[5];



* Data: Conventionally, class names start w/ capital letter.

```
//only import before class
class Abc {
    //encapsulated class
}
//nothing here
```

* Within a class, we can only have data & functions i.e. characteristics (data) & behaviour (functions)

Program: A set of Instructions which process data to give an output.

All data comprises of 3 parts:

<data type>	<Identifier>	<data>
{Register}	{label}	{notes}

* All data is of 2 types — Global (static, non static) / ~~Stat~~ Local

Date ___/___/___

→ Local Data:

If the user provides the data to the function directly then it is called local data.

User $\xrightarrow[\text{(local)}]{\text{data}}$ function (Publisher → Teacher)

"Local data belongs to a particular function/method and cannot be shared"

Eg → import java.util.Scanner;

```
Class Abc {
    int a,b;
    void data (double x) {
        Scanner sc = new Scanner (System.i
        int p = sc.nextInt();
    }
    void sum() {
        int c = 10;
    }
}
```

Here, p, c, x are local & a, b are global.

It is used when only one function in class.

Scanned with CamScanner

→ Global Data:

If the user provides the data to the class & class provides it to the functions then it is global data.

User $\xrightarrow[\text{(global)}]{\text{data}}$ [class] → f_1, f_2, f_3

"Global data belongs to the entire class & can be shared."

It is used when multiple func. in class.

Eg →

```
import java.util.Scanner;
class Abc {
    static double z;        // static class vars (common)
    int a, b;               // non static Instance vars
                                ↳ (unique)
    int c = 10;             // static
    void data(double x) {
        int d = 10;
    }
    void sum() {
        Scanner sc = new Scanner(System.in);
        int p = sc.nextInt();
    }
    int q;
}
```

On Compilation:

c	sum()	data()	z
10	p	d x	

Personal use

← Common use

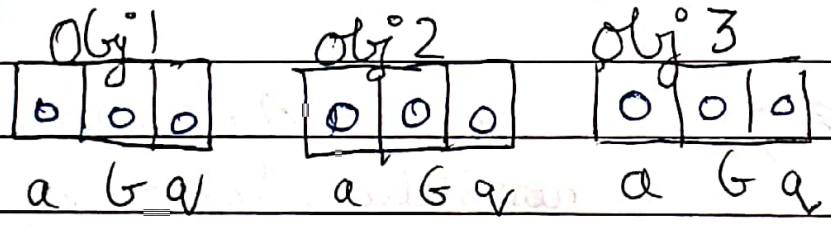
Scanned with CamScanner

→ Static: Memory alloted during complilation. It is permanent & can't be removed. (new)

→ dynamic/non-static: They are alloted during execution. It is temporary & can be removed (delete)

```
Abc obj1 = new Abc();
Abc obj2 = new Abc();
Abc obj3 = new Abc();
```

↓



Each object has its own copy of the non static variables i.e. they are different for different objects. ~~The static~~

There is only 1 copy of the static variables however, they are common/same for all objects

```
final double z = 10;        //static
```

This variable has been declared w/ final thus it can't be changed now.

Scanned with CamScanner

Date __/__/____ Page No.: _____

* Functions:

```
import java.util.Scanner;

Class Abc {
    Static double z;
    int a, b;
    ~~void~~ int sum() {
        return 10+20;    // can access a,b {not static}
    }
    Static int data() {
        Scanner sc = new Scanner(System.in);
        int p = sc.nextInt();
        return p;        // a static function can't access a, b
                                (non static)
    }
    public static void main() {
        int c = data();
        // no need to make objects to call
        // static functions!
        Abc obj = new Abc();
        int d = ~~obj.~~ obj.sum();
        // need object to call non static functions
    }
}
```

difference b/n static & non static.

→ * Static: Object Independent. Do not need objects to run. Because of this, they can't use non static vars.

→ Non-Static: Object dependent. They need objects to run. They can use non-static vars.

Scanned with CamScanner

→ Access Specifiers:
They tell us which classes can access a func. of a given class.

→ public: can be accessed by all classes. even ~~outside~~ derived classes.

→ private: accessable only by members of its own class.

→ protected (Not in Java): accessable by own class ~~itself~~ & extended classes.

When no access specifier is given, data is made private & func. made public.

→ Setter & getter ~~mo~~ methods:
to assign values manually, a method called setter method must be made:

```
Abc obj = new Abc();
void setData() {
    a = 10;
    b = 20;
    c = 9.6;
}
```

```
Abc() {
    a=10;
    b=20;
    c=9.6;
}
```

```
Obj.setData();
```

to get values of an object, a getter method must be used.

```
Obj.getData();
```

"A Setter method sets value to the instance var"

Scanned with CamScanner

Date __/__/____ Page No.: _____

* Diff. b/n setter & ~~getter~~ constructor is that the constr is called only once at the time of obj. creation automatically. A method is called manually and can be called any number of times

→ main function: It calls the other ~~other~~ methods and creates object. It does not perform any calculations conventionally.

It calls methods in order:

→ get data → call
→ call ✓ order → disp ✗ order
→ disp() → getData

X

5	19	9.8
a	b	c

int d = X.b; ⇒ d: 19

// the this keyword refers to the address of the current // object.

```
Class Abr {
    int a, b;
    double c;
    void f1() {                 // formal para
        this.a = 12;            // can't use X.a here
        this.b = 13;
        this.c = 17.8;
    }

    static void f2() {
        Abr X = new Abr();
        X.a = 2;
        X.b = 3;
        X.c = 5.6;
        X.f1();                 // actual para.
    }
}
```

"No real change in history has ever been acheived by discussions." —Subhash chandra bose

Scanned with CamScanner

Date ___/___/_____

Change f1() ⇒

```
void f1(AGr X, ALr Y) {   // formal
    // I can use 9 vars in here!
    // a, b, c of X, Y & this.
}
...
Q.f1(P, R);   // Actual, pass by ref.
...
AGr f1() {
    return this;
}
...
Q = R.f1();   // Q = R;
```

Scanned with CamScanner

```
import java.util.Scanner;
//4 types of setter methods.
class Abc {
    int a, b;
    double c;
    void getData(){
        a=10;
        b=20;
        c=9.8;
    }
    void getData(int p, int q, double r){
        a=p;                    //formal parameters
        b=q;                    //method prototype
        c=r;
    }
    void getdata(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter vals");
        a = sc.nextInt();
        b = sc.nextInt();
        c = sc.nextInt();
    }
    void getData(Abc x){
        a = x.a;
        b = x.b;
        c = x.c;
    }
    static void main(){
        Abc p = new Abc();
        Abc q = new Abc();
        Abc R = new Abc();
        p.getData();  //actual parameters
```

"No real change in history has ever been acheived by disruption" - Subhash chandra bose

Scanned with CamScanner

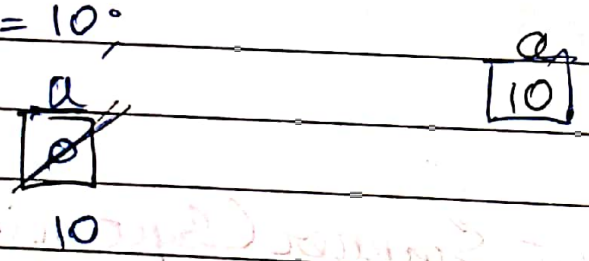
```
        Q.getData(2,9,17.6);
        R.getData(Q);
    }
}
```

//replace methods <u>void getData</u> w/ ACr

```
ACr P = new( );
ACr Q = new(2,9,7.8);
ACr R = new(Q);
```

int a;
a = 10;

int a = 10;



* If we do not make a constructor, default constructor is there which initializes all objects w/ default values.

→ Constructor:
It is a special method used to allot memory to an object & initialize its data members.

→ Naming rules for constructors:
→ Same as class name!
→ It has no returning type, not even void!

Scanned with CamScanner

→ Types of Constructors

- Default Constructor: A constructor with no parameters & no body. It is always created ~~regardless of you~~.

```
Abc () {          //default Const.
}
```

All default ∈ Non parameterized.

- Non-Parameterized: A constructor w/ no parameters but has a body.

```
Abc () {
    a = 2;
    b = 7;
    c = 10.2;
}
```

parameterized: A Constructor w/ both parameters and a body.

```
Abc (int a, int b, double c) {
    this.a = a;
    this.b = b;
    this.c = c;
}
```

Copy constructor: A constructor which initializes an object w/ another object.

```
Abc ( Abc P) {
    this.a = P.a;
    this.b = P.b;
    this.c = P.c;
```

Scanned with CamScanner

Date ___/___/___ Page No.

Constructor	Method
→ Name: same as class name.	can be any name except class name.
→ No return type, not even void	return type is always specified.
→ No need to call, it is invoked automatically.	It must be called by the user manually.
→ only invoked once at time of object creation.	can be invoked an ∞ no. of times.

→ Constructor overloading: If more than one type of constructor is defined w/i a class, it is called.... Eg →

```
class Point {
    double x, y;
    Point (double xx, double yy) {
        x = xx;
        y = yy;
    }
    void midPt (Point P, Point Q) {   // formal
        this.x = (P.x + Q.x)/2;
        this.y = (P.y + Q.y)/2;
    }
    void disp () {
        System.out.println("x = " + x);
        System.out.println("y = " + y);
    }
```

Scanned with CamScanner

Date __/__/____ Page No.: ____

```
line:
1   static void main() {
2       Point A = new Point(3.4, 5.9);
3       Point B = new Point(7.6, 10.1);
4       Point C = new Point();
5       C.midPt(A, B);      //actual
6       C.disp();
7   }
}
```

OUT:

x = 5.5,
y = 8.0

→ Formal: The parameters used at time of definition of function (in method prototype).

→ Actual: The parameters passed in a function at the time of calling the function.

Change midPt.:

```
Point midPt(Point P) {
    Point Q = new Point();
    Q.x = (x+P.x)/2;
    Q.y = (y+P.y)/2;
    return Q;
}
```

Change main (line:5) C = A.midPt(B);

Scanned with CamScanner

Date __/__/____ Page No.:

line

```
class Time {
    int h, m;
    void getTime (int h, int m) {
        this.h = h;   // local var is given
        this.m = m;   // preference to this.
    }
    void addTime (Time t1, Time t2) {
        this.m = t1.m+t2.m;
        this.h = this.m / 60;
        this.m % = 60;
        this.h += t1.h + t2.h;
    }
    void disp () {
        System.out.println ("hours =" + this.h);
        System.out.println ("minutes =" + this.m);
    }
    static void main () {
        Time A = new Time ();
        Time B = new Time ();
        Time C = new Time ();
        A.getTime (7, 40);
        B.get Time (5, 35);
        C.addTime (A, B);
        C.disp ();
    }
}
```

Scanned with CamScanner

Date __/__/____ Page No.: _____

① Change addTime():

```
Time addTime(Time t) {
    m += t.m;                    // can also make local obj;
    h += m/60;
    m %= 60;
    h += t.h;
    return this;
}
```

Change line 23:

```
C = A.addTime(B);
```

② Change getTime()

```
Time(int h, int m){
    this.h = h;
    this.m = m;
}
```

remove lines 21-22

Change lines 18-19

```
18 Time A = new Time(7, 40);
19 Time B = new Time(5, 35);
```

'\u0000' → default char value.

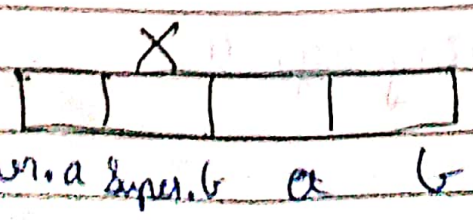
null → string default.

Scanned with CamScanner

Date __/__/__ Page No.

Inheritance

```
class Abr {
    int a, b;
    static double c;
    void f1 (int a) {
        // a, b, c, this.a
    }
    ~~static~~ void f2() {
        // a, b, c, this.a
    }
}

class Pqr : extends Abr {        // path of
    int a, b;                    // location different
    ~~static double c;~~
    void f1 (int a) {
        // a          local
        // this.a, b  global
        // super.a  super.b, c  base
    }
    static void main () {
        Pqr X = new Pqr();
        X.f1 (10);               // derived class
        X.f2 ();                 // base class
        X.super.f1 (10);         // base class
    }
}
```



Scanned with CamScanner

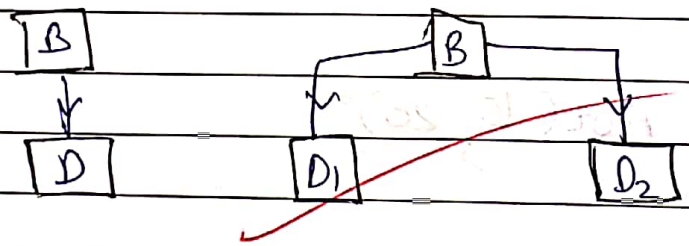
this - keyword overides local over global.
Super - Keyword overides derived class over base.

* Def^n: The properties of 1 class aquired by another class.

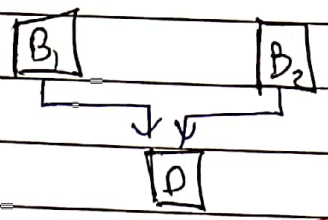
The class sharing its properties is called the Super/base class
The class receiving the properties is called the extended/sub/derived class.

* Types of Inheritance: A derived class with ~~only~~ ...

→ Single: Only 1 base class.



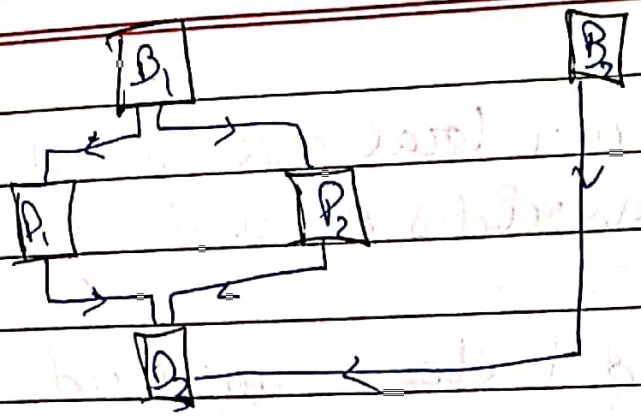
→ Multiple: more than one base class.



→ Multi-level: a base class which is iself a derived class.

→ Hybrid (or hierarchical): more than one base class of different levels.

Scanned with CamScanner



Note*: Java only supports single & multi.

* Constructor w/ Inheritance.

```
class Abc {
  int a, b;
  Abc (int x, int y) {
      a = x;
      b = y;
  }

  Abc P = new Abc(10, 20);
```

P

10	20
a	b

lines

```
1  class Pair {
2      double c;
3      int d;
4      Pair (int aa, int b, double c, int d) {
5          super(aa, b);
6          this.c = c;
7          this.d = d;
8      }
9  }
```

*must preceed global

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

① Change line 5:

5 super();

② Change line 5:

5 a = aa; // must precede derived

+ super.b = b; // ~~local~~ globals.

Scanned with CamScanner

Date __/__/____ Page No.

(line)

```
1  abstract class Account {
2      double P;
3      int an;
4      Account (double pp, int x) {
5          P = pp;
6          an = x;
7      }
8      void show() {
9          System.out.println("Principle" +P);
10         System.out.println(" Acc. no. " + an);
11     }
12 }
```

abstract class: A class which cannot be instantiated. It can't make its own objects.

```
1  class Simple extends Account {
2      double t, r;
3      Simple (double p, int a, double t, double r)
4          Super (p, a);
5          this.p = p;
6          this.r = r;
7      }
8      double interest () {
9          return (p*r*t) / 100;
10     }
11     void show ( ) {
12         Super. show ();
13         System.out.println (" Time:" + t);
14         System.out.println (" Rate:" + r);
15         System.out.println ("Interest" + interest());
16     }
```

To manifest the divinity is the highest

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

```
    static void main () {
        Simple x = new Simple (7000.26, 7214, 7.6, 8.5);
        x.show();
    }
}
```

Scanned with CamScanner

Date __/__/__ Page No.:

Interface

```
interface Abc {
    // Data -> Real: always static & final
    //  +
    // functions -> abstract: only prototype
    }                  // no use, defined by derived class
                       // itself.
```

Lines

```
1  interface Abc {
2      static final double g = 9.8;
3      double pi = 3.14159;
4      void Area();
5      void volume();
6  }
```

```
1  interface Pqr {
2      int a = 10;
3      double c = 20.6;
4  }
```

```
1  interface Xyz implements Abc, Pqr {
2
3  }
```

Scanned with CamScanner

Interface	Class
→ The data members are static & final while functions are abstract	A class has abstract data members, functions are R & defined.
→ Supports multiple inheritance	does not support multiple inheritance

class Employee implements XyZ {

}

class A extends B implements C, D {

}

A → derived, B → base, C & D → interface.

interface: is an abstract class which contains static and final data members and prototypes of functions which are not defined

Scanned with CamScanner

Date

* Class can use any access specifier while interface can only ~~access~~ public.

Example:

line

```
1  interface Pqr {
2      void area();
3  }
```

```
1  interface Abc {
2      void area();
3  }
```

```
1  class XYZ implements Abc, Pqr {
2      void area();
3      Pqr obj = new XYZ(); // Call area of Pqr
4      Abc obj = new XYZ(); // Call area of Abc
5  }
```

Scanned with CamScanner

Date __/__/__ Page No.: _____

Data Structures:

* Arrangement of data in the memory → data structure.

It can be linear & non-linear.

Linear	Non-linear
Arrays < 10, 20	queues < circular, dequeue
Stacks	link list
queues	tree
(linear)	(graphs) → best (N/C)

Good
30/7/19

5/8/19

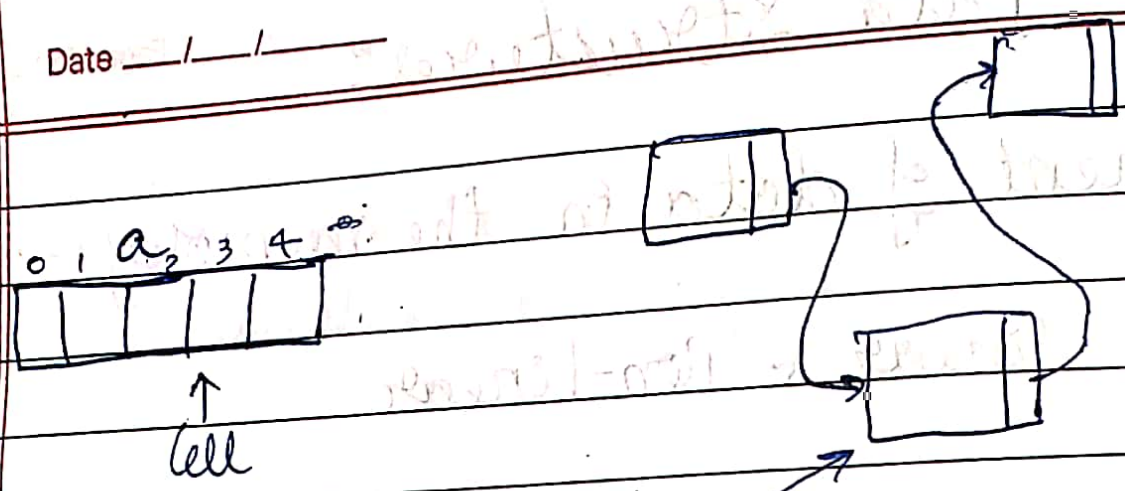
INSPECTED

07/8/19

INSPECTED

Scanned with CamScanner

Date ___/___/___



(data + Reference) letter.

Scanned with CamScanner

Date __/__/__ Page No.: _____

Arrays (1.D)

An array is a linear data structure which consists of the elements of the same data type.

a → address

2	4	9	13	6
0	1	2	3	4

subscript/index

```
int a[] = new int[5];
int []a = { 2, 14, 9, 13, 6 };

int s = a[1] + a[3];
```

s
27

a → variable, has 1 value

a[] → subscripted variable, has many values.

→ direct initialization
→ method parameters
→ Scanner input.

Scanned with CamScanner

Date __/__/____ Page

```
1  import java.util.Scanner;
2  class Array {
3      int a[], n;
4      Array(int nn){
5          n = nn;
6          a = new int[n];
7      }
8      void readData(){
9          Scanner sc = new Scanner(System.in);
10         //System.out.println("Enter no. of e");
11         //n = sc.nextInt();
12         System.out.println("Enter"+n+" el");
13         int c = 0;
14         while(c < n){
15             a[c++] = sc.nextInt();
16         }
17     }
18     void display(){
19         for(int c=0; c<n; c++){
20             System.out.println(a[c]);
21         }
22     }
23     static void main(String ar[]){
24         Scanner sc = new Scanner(System.in);
25         System.out.println("Enter no. of");
26         Array arr = new Array(sc.nextInt());
27         arr.readData();
28         : // Prog
29         arr.display();
```

line 10 & 11 used when no constructor is made & line 26 del.

Scanned with CamScanner

Date ___/___/_____ Page

* linear Search: (O(1) & O(n))

```
int Lsearch (int v){
    int c = -1;
    for (int i = 0; i < n; i ++) {
        if (a[i] == v) {
            c = i;
            break;
        }
    }
    return c;
}
//alternatively
int Lsearch (int v) {
    int c = 0;
    while ( c < n && a[c] != v)
        c++;
    if (c < n)
        return c;
    return -1;
```

Scanned with CamScanner

Date ___/___/___

* Binary Search ($O(1)$ & $O(\log n)$)

```
int Bsearch(int v){
    int l=0, u=n-1, m=(l+u)/2;
    while (l<=m && a[m] != v){
        if (a[m] > v)
            u=m-1;
        else
            l=m+1;
        m = (l+u)/2;
    }
    if (l<=u)
        return m;
    return -1;
}
```

* Bubble sort ($O(n^2)$ & $O(n)$)

```
int Bsort( ) {
    for (int i=0; i<n-1; i++) {
        for (int j=0; j<n-1-i; j++) {
            if (a[i] > a[j+1]) {
                int t = a[j];
                a[j] = a[j+1];
                a[j+1] = t;
            }
        }
    }
}
```

Scanned with CamScanner

* Selection Sort ($O(n)$ & $O(n^2)$)

```
void Ssort () {
  int t, min;
  for ( int i = 0 ; i < n-1 ; i++) {
      min = i;
      for ( int j = i+1 ; j < n ; j++) {
          if (a[min] > a[j])
              min = j;
      }
      if (min != i) {
          t = a [min];
          a[min] = a[i];
          a[i] = t;
      }
  }
}
```

* Insertion & Deletion :

a

2	19	6	18	4	7				
0	1	2	3	4	5	6	7	8	9

$v = 25$ & $k = 2$

$a[i+1] = a[i]$; $\forall\ i \geq k$

~~int~~ a

2	19	6	~~18~~	4	7	
0	1	2	3	4	5	6

$i = 3$.

$a[i] = a[i+1]$;

Scanned with CamScanner

Date ___/___/_____

→ Insertion:

```
void insert(int v, int m) { n++;
    for (int i = n-1; i >= m; i--) {
        a[i+1] = a[i];
    a[m] = v;
}
```

→ void deletion (int v) {

```
    int m = a.Bsearch (v);
    for (int i = m; i < n; i++) {
        a[i] = a[i+1];
    n--;
}
```

* Insertion sort (O(n) & O(n²))

```
void Isort ( ) {
    int t, j;
    for (int i = 1; i < n; i++) {
        t = a[i];
        j = i-1;
        while ( j >= 0 && t < a[j] ) {
            a[j+1] = a[j];
            i--;
        }
        a[j+1] = t;
    }
}
```

Scanned with CamScanner

* Maximum & Minimum of Array:

```
void maxmin() {
    int max = 0, min = 0;
    for (int i = 1; i < n; i++) {
        if (a[max] < a[i])
            max = i;
        if (a[min] > a[i])
            min = i;
    }
    System.out.println("Max =" + a[max]);
    System.out.println("Min =" + a[min]);
}
```

* Merging (w/o reps) & w/reps & Common

a[]

2	0
5	1
11	2
13	3
17	4
21	5

n

+

b[]

1	0
2	1
7	2
17	3
35	4

m

→

c[]

1	0
2	1
5	2
7	3
11	4
13	5
17	6
21	7
35	8
	9
	10

n+m

d[]

1	0
2	1
2	2
5	3
7	4
11	5
13	6
17	7
17	8
21	9
35	10

n+m

e[]

2
17

n+m

Scanned with CamScanner

Date __/__/____

lines

```
void merge (Array P, Array Q) {
    int x = 0, y = 0, z = 0;
    while (x < P.n && y < Q.n) {
        if (P.a[x] == Q.a[y]) {
            a[z] = P.a[x];
            x++;
            y++;
        }
        else if (P.a[x] > Q.a[y])
            a[z] = Q.a[y++];
        else
            a[z] = P.a[x++];
        z++;
    }
    while (x < P.n)
        a[z++] = P.a[x++];
    while (y < Q.n)
        a[z++] = Q.a[y++];
    n = z;
}

static void main (String ar[]) {
    Array P = new Array (10);
    Array Q = new Array (20);
    Array R = new Array (30);
    P.readData();
    Q.readData();
    P.~~readData();~~ Isort();
    Q.Isort();
    R.merge (P, Q);
    R.disp();
}
```

Scanned with CamScanner

→ Alternatives:

* remove lines 4-8 for merging w/ reps
* change lines → for merging on only common (intersectio

-	10	a[z] = Q.a[y++];	+	10	y++;
-	12	a[z] = P.a[x++];	+	12	x++;

* change line 1:

```
1  Array merge(Array P, Array Q){
2          Array Q = new Array(n+P.n);
3          int x = 0, y = 0, z = 0;
:
```

& replace a[z] everywhere by Q.a[z]
& replace Q.a[z] everywhere by a[z]

* Change lines 29 →

```
29 R = P.merge(Q);
```

* Decimal to Binary / Octal / Hex:

```
void dectoAny(int v, int b){
  int x=0, num=v;
  while(num>0){
    a[x++]=num%b;
    num /= b;
  }
  System.out.print("Equivalent of"+v+" (in base b) =");
  for(int i=x-1; i>=0; i--){
    if(a[i]>9)
      System.out.print((char)(a[i]+55));
    else
      System.out.print(a[i]);
  }
}
```

Scanned with CamScanner

Array (2-D)

	0	1	2	3	4	5
0	00	01	02	03	04	05
1	10					
2	20					
3	30					
4	40					

5X6 → array
row X Column.
a[x][y]
↓ ↓
Row Column

```
import java.util.Scanner;
class Array2D {
    int a[][], m, n;
    static Scanner sc = new Scanner(System.in);
    Array2D (int mm, int nn){
        m = mm
        n = nn
        a = new int[m][n];
    }

    void readData() {
        System.out.println("Enter" + (m*n) + "elements");
        for (int i=0; i<m; i++){
            for (int j=0; j<n; j++) {
                a[i][j] = sc.nextInt();
        }}
    void disp() {
        for (int i=0; i<m; i++){
            for (int j=0; j<n; j++) {
                System.out.println(a[i][j] + "\t");
        }
    }

    Static void main (String ar[]){
        Array2D obj = new Array2D(3, 4);
        obj.readData();
        :
        obj.disp();
    }
```


"Education is the manifestation of perfection already in a man." —Swami Vivekananda

Scanned with CamScanner

Date ___/___/_____ Page No.:

* Sum of each row/column.

```
1  void sumrow() {
2      int s = 0;
3      for (int r = 0; r < m; r++) {
4          for (int c = 0; c < n; c++)
5              s = s + a[r][c];
6          System.out.println("Σ of row" + (r+1) + " = " + s);
7      }
8  }
```

For sum of ~~rows~~ columns interchange lines 3 & 4

* Sum of principal diag:

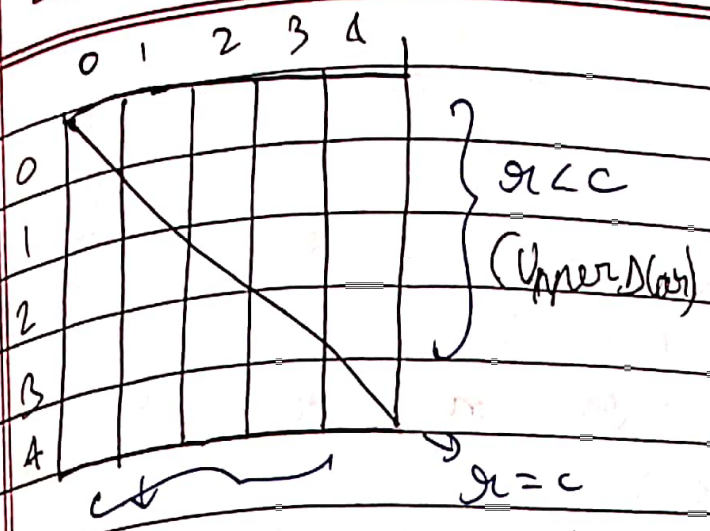
```
1  void sumpdiag() {
2      int s = 0;
3      for (int i = 0; i < n; i++)
4          s += a[i][i];
5      System.out.println("Σ of principal diag = " + s);
6  }
```

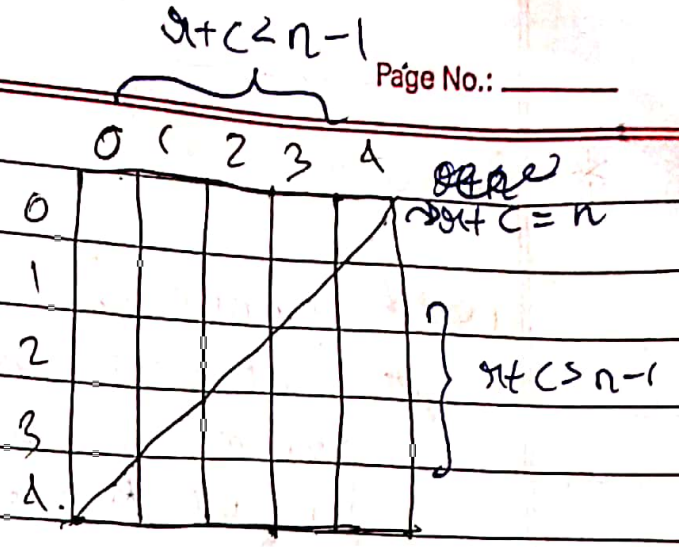
for Σ of rev. diag, change line 4:

4 s += a[i][n-1-i];

Scanned with CamScanner

Date __/__/____ Page No.: _____





~~if(r==4~~

```
if( r==4 || r==0 || c==0 || c==4)
    //boundary element.
else
    // non boundary element.

if ((r==0 || r==4) && ( c==0 || c==4))
    //Corner element
else
    // not corner element.
```

→ Sum of Diags, (4)
→ Sum of boundary, non boundary,
→ Sum of Corners.

Scanned with CamScanner

Date ___/___/_____ Page No.:

* Printing:

```
void print(){
    for( int r = 0; r < m; r++) {
        System.out.println();
        for( int r = 0; r < m; r++){
            if( r <= c)
                System.out.print(a[r][c] + "\t");
            else
                System.out.print( " \t");
        }
    }
}
```

* Addition of 2 Matrices: (order must be ~)
(or Sub)

```
void add( Array2D P, Array2D Q) {
    for( int i = 0; i < m; i++) {
        for( int j = 0; j < n; j++)
            a[i][j] = P.a[i][j] + Q.a[i][j];
    }
}
```

~~Readd~~

Scanned with CamScanner

Date __/__/____ Page No.: _____

* Transpose of Matrix: (order must be inverse).

```
void transpose(Array2D P) {
    for(int i=0; i<m; i++) {
        for(int j=0; j<n; j++)
            a[i][j] = P.a[j][i];
    }
}
```

// alternatively, for square matrix:

```
void transpose( ) {
    for(int i=0; i<m; i++) {
        for(int j=i; j<n; j++) {
            int t = a[i][j];
            a[i][j] = a[j][i];
            a[j][i] = t;
        }
    }
}
```

Scanned with CamScanner

Date ___/___/_____ Page No.: ____

* Multiplication (sizes $m \times n$ & $n \times p$).

```
void multiply (Array 2D P, Array 2D Q){
    for(int i=0; i<m; i++){
        for(int j=0; j<n; j++){ a[i][j]=0;
            for(int k=0; k<P.n; k++){
                a[i][j] += P.a[i][k] * Q.a[k][j];
            }
        }
    }
}
```

* Sorting rows/columns:

~~void sort(int i=0; i<m; i++){~~

~~void mult (Array 2D P, Array 2D Q) {~~

```
void sort () {
    for(int i=0; i<m; i++){
        for(int c=0; c<n-1; c++){
            for(int c2=0; c2<n-1-i; c2++){
                if(a[i][c2] > a[i][c2+1]){
                    int t = a[i][c2];
                    a[i][c2] = a[i][c2+1];
                    a[i][c2+1] = t;
                }
            }
        }
    }
}
```

for sorting colums, replace $a[i][c]$ w/ $a[c2][i]$ & $a[i][c2+1]$ w/ $a[c2+1][i]$

"It is the prime responsibility of every citizen to feel that he is a party." Vardar Vallabhbhai Patel

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

* Sorting diagonals:

```
void sortd() {
    for (int i = 0; i < m-1; i++) {
        for (int j = 0; j < n-1-i; j++) {
            if ( a[j][j] > a[j+1][j+1] ) {
                int t = a[j][j];
                a[j][j] = a[j+1][j+1];
                a[j+1][j+1] = t;
            }
        }
    }
}
```

for ~~sorting~~ sorting reverse diagonal,
replace

a[j][j] w/ a[n-1-j][n-2-j]
a[j+1][j+1] w/ a[n-j][n-j-1]

Scanned with CamScanner

Date ___/___/_____ Page No.: ____

* Sorting Entire Matrix / Parts of the Matrix :

```
0  void SortMatrix () {
1      int x = 0;
2      int G[] = new int[m*n];
3      for(int r = 0; r < m; r++){
4          for (int c = 0; c < n; c++){
5              G[x++] = a[r][c];
6          }
7      }  G.n = x;  // this is imp. for Sorting part of
8      //2D to 1D.          // Matrix :: we are doing m×n always
9      G.BSort();  // during initial of G
10     // from Array class
11     x = 0;
12     for (int r=0; r<m; r++){
13         for (int c=0; c<n; c++){
14             a[r][c] = G[x++];
15     } //1D to 2D
16 }
```

for a part (say upper diag) of matrix change loop condition according to it:
Change lines 3-4 & 12-13:

```
3 [12] for(int r = 0; r<m; r++){
4 [13]     for (int c = r; c < n; c++)
```

Scanned with CamScanner

Date __/__/____

Page No.: _____

* flipping Horizontally (Mirror):

4	1	7	11
5	12	3	13
8	6	2	4

~~11~~	~~7~~	~~1~~	~~4~~
~~5~~	~~12~~	~~3~~	~~13~~
~~8~~	~~6~~	~~2~~	~~4~~

11	7	1	4
13	3	12	5
4	2	6	8

8 6 2 4 ← vertically flip.

5 12 3 13

4 1 7 11

```
0 void flipH( ) {
1     for (int r = 0; r < m; r++) {
2         for (int c = 0; c < n; c++) {
3             int t = a[r][c];
4             a[r][c] = a[r][n-1-c];
5             a[r][n-1-c] = t;
6         }
7     }
8 }
```

to flip vertically, change lines 4-5:

```
4 a[r][c] = a[n-1-r][c];
5 a[n-1-r][c] = t;
```

Scanned with CamScanner

Date __/__/____ Page No.: _____

Stacks.

(LIFO - Last In First Out)

top = -1 initially.

	3
35	2 = top
15	1
10	0

push → add on end

pop. → remove from end

a → identifier

10 → value

0xff94 → reference

Implementation of ~~array~~ stack w/ array.

∵ there is no pointers in java, we can't actually make a stack, but only show its working by array.

Stack → A stack is a linear data structure w/ 1 end operation. It works on the principle of LIFO (last in first out).

→ Algorithm to push object in a stack:

Step 1 Start

Step 2 if (top == capacity) then stack is "full" "OVERFLOW", exit.
else:

Step 3 top++

Step 4 Stack[top] = value (return)

Step 5 end

→ Algo for pop:

Step 1 start

" 2 if (top <0) then stack is empty "UNDERLOW", exit
else:

" 3 value = stack[top]

" 4 top--

" 5 end.

Scanned with CamScanner

```
class Stack {
    int s[], n, top;
    Stack (int n) {
        this.n = n;
        s = new int[n];
        top --;
    }
    void push (int v) {
        if( top < n - 1 ) {
            s [++top] = v;
        else
            System.out.println ("Stack full");
    }
int ~~void~~ pop() {
        if( top >= 0)
            return s [top--];
        else
            return -999;
    }
}
```

* Note * value is never actually deleted (pop)
∵ there is no delete in ~~to~~ java.

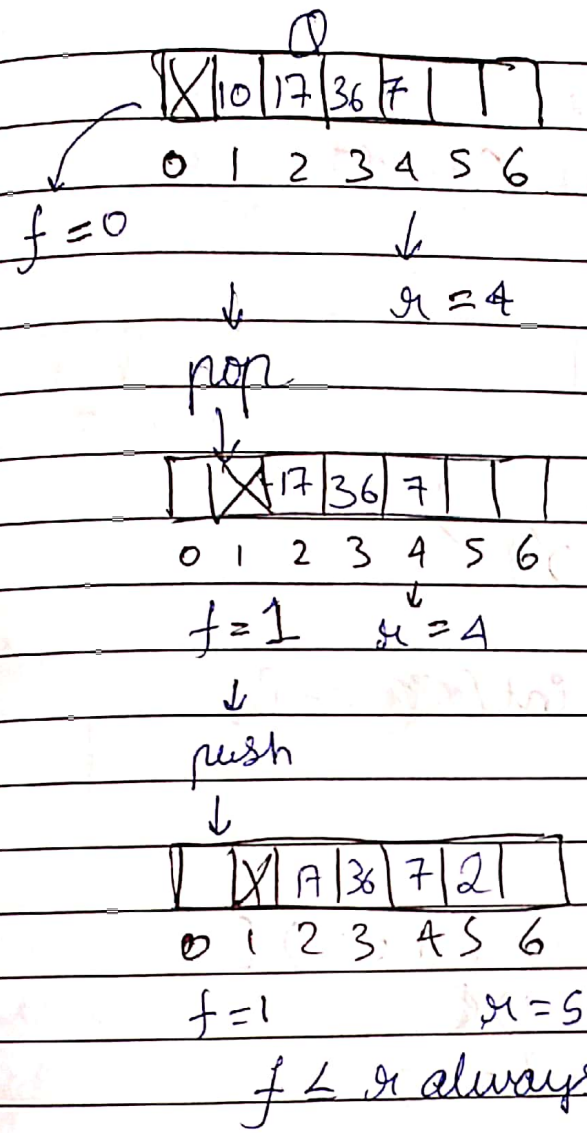
Scanned with CamScanner

Date __/__/____ Page No.: _____

Queue

It is a linear data structure w/ 2 end operation. It works on the principle of FIFO (First In First Out).

* Eg:



f < r always

* Algo for push

step 1 Start
step 2 if (rear = = cap) then "OVERFLOW" exit
else
step 3 rear = rear + 1
step 4 Queue[rear] = Value
step 5 end

Scanned with CamScanner

* Algo for pop:

Step1 Start

Step2 if (front == rear) "UNDERFLOW" exit

else

Step3 front = front + 1

Step4 value = Queue[front]

Step5 end

* Class:

```
class Queue {
    int Q[], f, r, n;
    Queue (int n) {
        this.n = n;
        Q = new int[n+1];
        f = 0;
        r = 0;
    }
    void push (int r) {
        if (r < n-1)
            Q[++r] = r;
        else
            System.out.println("Queue full");
    }
    int pop () {
        if (f != r)
            return Q[++f];
        else
            return -999;
    }
} //display() make urself. (f+1 to r).
```

Scanned with CamScanner

* Dqueue :

		X	14	9		6	
0	1	2	3	4	5	6	7

f=2 r=5

f > 0 else no space

↓

pushfront

↓

	X	7	14	9		6	
0	1	2	3	4	5	6	7

f=1 r=5.

↓

poprear

↓

	X	7	14	9			
0	1	2	3	4	5	6	7

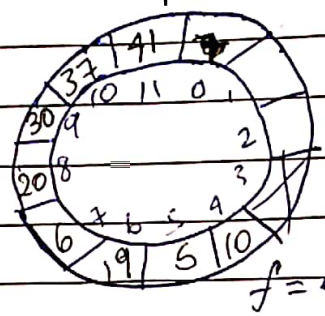
f=1 r=4.

* functions add to class:

```
void pushfront (int x) {
    if (f > 0)
      Q [f--] = x;
    else
      System.out.println ("full");
}

int poprear() {
    if (f == r) {
      System.out.println ("empty"); return -999;
    else
      return Q[r--];
}
```


Scanned with CamScanner

* Queue : → r = 11



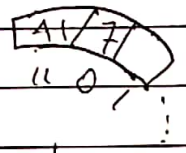
* Cap = 12
(n = 12)
f = 0 & r = 0
r = (r+1) % 12
(n)

f = 3

↓ push

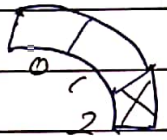
→ if ((r+1) % n == front)
OVERFLOW.

↓



↓ pop × if (f == r)
UNDERFLOW

↓



f = 2
r = 2

* Push:

Step 1 Start
Step 2 if (~~front~~ (rear+1) % Size == front) → overflow exit
else
Step 3 rear = (rear+1) % Size
Step 4 Q[rear] = value.
Step 5 end.

* Pop:

Step 1 start.
Step 2 if (front == rear) "UNDERFLOW" exit
else
Step 3 front = (front+1) % Size
Step 4 value = Queue [front]
Step 5 end.

* Size is given n, but array gives for n+1
∴ 1 is pointer.

Scanned with CamScanner

Date ___/___/_____ Page No.: _____

* functions for this class:

```
void push(int v){
    if ((r+1)%n != f) {
        r = (r+1) %n;
        Q[r] = v;
    }
    else
        System.out.println("full");
}

int pop(){
    if (f != r) {
        f = (f+1) %n;
        return Q[f];
    }
    else
        return -999;
}

//display : for(int i = f; i != r; i = (i+1) % n)
```

∵ we don't know where it starts or ends.

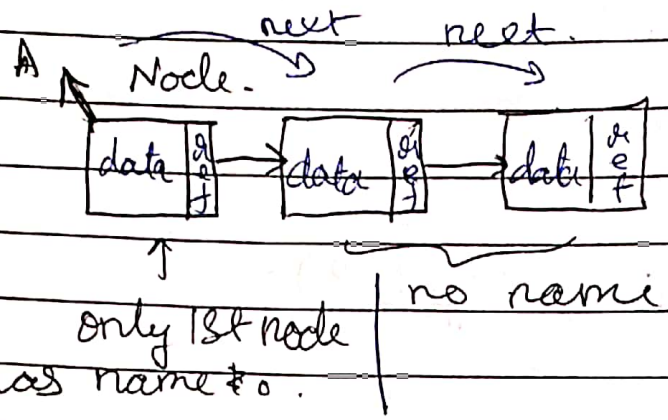
Scanned with CamScanner

Date ___/___/___ Page No.:

link list.

(Single):

A link list is a linear data structure comprising of nodes, w/ each node having a data component & a reference component

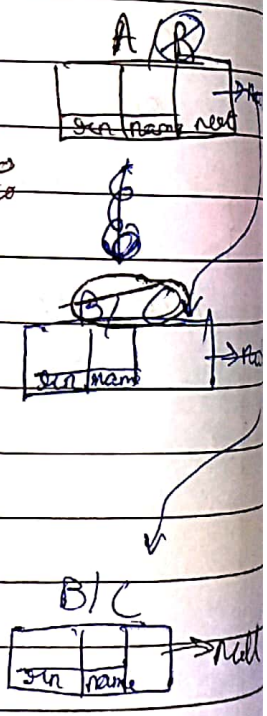


(double link list:) ← [ref1 | data | ref2] →

Creation

```
Node A, B, C;
A = new Node();
A.getData();
A.next = null;
B = A;
```

```
class Node {
    int sn;
    String name;
    Node next;
}
```




```
* i=0 is A */ for (int i = 1; i<n; i++) {
        C = new Node();
        C.getData();
        C.next = null;
        B.next = C;
        B = C;
    } B = A;
```

```
*while (B != null) {
        B.display();
        B = B.next;
    }
```

traversal

Scanned with CamScanner

* Algo: (Traversal)

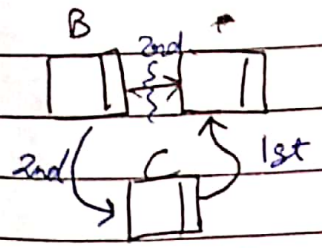
Step 1 Start
Step 2 Set temp pointer (B) to first node.
Step 3 Repeat 4 & 5 until pointer reaches null.
Step 4 // perform task → search for the node, if found disp.
Step 5. move pointer to next node
Step 6 end.

* Searching:

```
void search (Node A, int roll) {
    Node B = A; // use this instead of A (B)
    While ( B != null && B.rn != roll)
        B = B.next;
    if ( B != null)
        System.out.println ("Name: " + B.name);
    else
        System.out.println ("Not found");
}
```

* Insertion

// A = A.insert(V, roll);




```
C.getData();
C.next = B.next;
B.next = C;
```

*Search w/ return type.

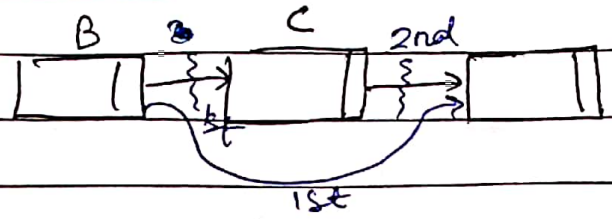
```
Node insert (Node V, int roll) {
    //Search.
    if (this.rn == roll) {
        V.next = this;
        return V;
    }
    Node B = this;
    Node C = B.next;
    While (C != null && C.rn != roll) {
        B = C;
        C = C.next;
    }
    V.next = C;
    B.next = V;
    return this;
}
```


Scanned with CamScanner

Date ___/___/_____

Pag

* Deletion:




```
void delete (int roll){
    Node B = new Node search(roll);
    if (B != null) {
        Node C = new Node();
        B.next = C.next;
        C.next = null;
    }
}
```

```
void delete(int roll){
    Node B = search(roll);
    B = B.next;
}
```

// A = A.delete(r);

```
Node delete( int roll ){
    Node B = this ;
    if( this.rn == roll) {
        B = B.next ;
        return B;
    }
    Node C = B.next;
    while ( C != null && C.rn != roll ){
        B = C;
        C = C.next;
    }
    B.next = C.next ;
    return this;
}
```

Scanned with CamScanner

Date __/__/__ Page No.: ____

Recursion:

(recurring, calling on to itself).

Recursion is calling on to itself.

* Difference b/n Recursion & Iteration:

Iteration	Recursion
→ fast process	slow process.
→ less memory	more memory.

→ Recursion - makes complex programs simpler & easier to answer.

→ Thing which we know from start - "base case"

eg:

recursive: $n! = n(n-1)!$ The repitative case is the recursive case.

base case: $0! = 1$ The pre-known values before any calculation

* Direct Recursion & Indirect Recursion.

```
void A() {
    A();
}
```

```
void A() {
    B();
}

void B() {
    A();
}
```

Scanned with CamScanner

Date ___/___/_____ Page No.

* Finite & Infinite Recursion:

- Finite: A recursive function, having one or more base case which are reachable or attainable.
- infinite: A recursive function w/no base case or a base case which is unreachable or not attainable.

→ Recursion works on the principle of LIFO: last memory opened is the first memory to be closed.

Eg →

```
int factorial (int n) {
    return (n < 2) ? 1 : n * factorial (n - 1);
}
```

120 ← factorial (5) ↖ 120

↳ factorial (4) * 5 ↖ 24 x_5

↳ factorial (3) * 4 ↖ 6 x_4

↳ factorial (2) * 3 ↖ 2 x_3

↳ factorial (1) * 2 ↖ 1 x_2

↳ return 1 x_1

Scanned with CamScanner

Date __/__/____ Page No.: _____

```
1 void test (int n) {
2     if( n != 0) {
3         System.out.println (n); //FIFO
4         test (n - 1);
5         System.out.println (n); //LIFO
6     }
7 }
```

Output:

3
2
1
1
2
3

Replace line 4:

```
4 test (--n);
```

→ OUTPUT

3
2
1
0
1
2

* Sum/product:

```
void sum(int n) {
    return (n == 0) ? 0 : n%10 + sum (n/10);
}
```

↑ change to * for prod.

```
void power ( int a, int b) {
    return ( b == 0 ) ? 1 : a*power ( a, b) ;
}
```

Scanned with CamScanner

Date __/__/____ Page No.:

```
int fibo (int n){
    if (n= = 1)
        return 0;
    else if (n = =2)
        return 1;
    else
        return fibo(n-2) + fibo(n-1);
}
// returns fn
```

* Bsearch:

```
int Bsearch (int l, int u, int v){
```

Date __/__/____ Page No.:

```
* int HCF (int p, int q) {
    if (p = = 0)
        return q;
    else
        return HCF (q%p, p);
}

* void dectoany(int n, int b){
    if (n != 0)
        dectoany (n/b);
        if (n% b > 9)
            System.out.println ((char)(n%b + 55));
        else
            System.out.println (n%b);
}
```

Scanned with CamScanner