⇒ Computer Programming :-

⇒ Number system & codes :-

1) RADIX / BASE = no of distinct digits.

2) $(N)_b = (d_{n-1}\, d_{n-2}\, d_{n-3} \ldots d_1\, d_0 \bullet d_{-1}\, d_{-2} \ldots d_{-f}\, d_{-m})$

number, base, msd, Integer portion, Radix pt., fractional portion, LSD

$$0 \le (d_i \text{ or } d_{-f}) \le b-1$$

Face value : actual number in itself
Place value : weight attached to its position

3) Nibble = 4 bits Byte = 8 bits

$K = 1024 = 2^{10}$ $16K = 16384$

Mega : $M = K \times K = 2^{20}$

Giga : $G = K \times K \times K = 2^{30}$

Tera : $T = K \times K \times K \times K = 2^{40}$

4) Binary to decimal : $1011_2 = 1\times2^0 + 1\times2^1 + 0\times2^2 + 1\times2^3$
$= 1 + 2 + 8 = 11$

Decimal to Binary : DOUBLE - DABBLE METHOD :

2	13	1
2	6	0
2	3	1
2	1	1
	0	

$\Rightarrow 1101.$

For fractions :

BCD $\Rightarrow$ $0.1011 \Rightarrow 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$
$= 0.5 + 0 + 0.125 + 0.0625 = 0.6875$

DCB = successive $\times^n$ by 2 eg

f	fx2	rem.	[fx2]
0.6875	1.3750	0.375	1
0.375	0.750	0.75	0
0.75	1.5	0.5	1
0.5	1	0	1

$= 0.1011$

5) Signed Representation and Arithmetic

1's complement Representation ($\bar{A}$) : each 1 replaced by 0 & 0 by 1.
Magnitude remains same but sign changes

↳ for n-bit number, max^m +ve number that can be represented is $(2^{n-1}-1)$ & max^m -ve is $-(2^{n-1}-1)$

2's complement Representation (A') : 2's complement = 1'st complement + 1.

$(11011)_2$ in 2's form is $(-5)_{10}$ as $11011-1 = 1010 \xrightarrow{1^{st}\ comp} 0101 \Rightarrow 5$

↳ for an n-bit number, max^m +ve no is $(2^{n-1}-1)$ & maximum -ve is -2^{n-1}

* Both have msb = 0 for +ve no & msb = 1 for -ve no

sign-magnitude Representation : msb is the sign & rest is the no.
eg $(1010)_2 \Rightarrow (-2)_{10}$ & $(0101)_2 = (5)_{10}$.

Example: $A = 0101$

$A_{(1c)} = 1010 = \bar{A}$ $\quad \bar{A}'_{1c} = 0100$

$A_{(2c)} = 1011 = A'$ $\quad A'_{2c} = 0101 = A$ Ans

$\therefore (A_{2c})_{2c} = A$ Ans

$\therefore$ 2's complement of 2's complement is the no itself

2's complement $+^n$, $-^n$:-

very important in computer hardware

+ no ⇒ signed magnitude form
− no ⇒ 2'c complement form

$\therefore A - B = A + \bar{B}'$

NOTE for 2'c ⇒ If lsb is 0, scan R→L, & retain till 1st 1 & toggle rest
If lsb is 1, toggle all except lsb

1's subtraction :

A - B

A > B
(i) take $\bar{B}$
(ii) add to A
(iii) remove the carry & add at the end

A < B
(i) take $\bar{B}$
(ii) add to A
(iii) No carry but sign is opposite (result in 1c form)

usefulness ⇒ Only ADDER is needed to compute $-n$.

2's subtraction

A - B

A > B
(i) take B'
(ii) add to A
(iii) Ignore the carry, always there

eg $(1111)_2 - (1010)_2$

1111
0110
(1) 0101
ignore ↓
0101

A < B
(i) take B'
(ii) add to A
(iii) No carry. Result is in 2's form.

eg (1000) - (1010)

1000
0110
1110
add ↓ 2c
sign − [0010]

usefulness ⇒ same as 1c $-n$ & also no end-around-carry op^n needed.

* Addition in 2c system : (A+B)

(i) A, B > 0
sign bit of A, B & sum all are 0

(ii) A > 0, B < 0
B in 2c form
ignore carry

(iii) A < 0, B > 0
2c form.
result

(iv) A, B < 0
again ignore carry.

Nothing fancy, just ignore the carry & the result is in 2c form, takes a 2c again to get magnitude if signbit is 1.

Similar for subtraction.

9's complement : subtract each digit from 9.
10's complement : 9's complement + 1.

Just like 1's & 2's complement, in case of 9's subtraction, carry is added around
in 10's subtraction, carry is dropped

7) BINARY CODED DECIMAL (BCD) :

It is a combination of 4 binary digits that represent decimal numbers. eg 8421 code where 8421 indicate the binary weights of 4 bits used to represent decimal digits 0 to 9.

BCD Addition : (i) add using binary addition
(ii) if a 4-bit sum is $\leq 9 \Rightarrow$ valid BCD
(iii) else if sum is >9 or carry is generated add $(0110)_2 = 6$ to the 4 bit sum to get to BCD. If a carry results again, add it to next 4-bit group.

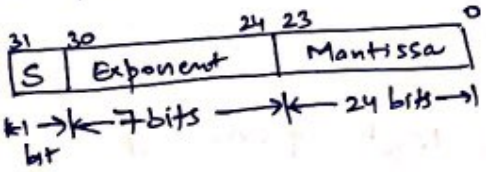
eg

```
  0001  1001        =  (19)
  0001  0100        + (14)
 ------------
  0010  1101  → invalid
        0110
 ------------
  0011  0011  → both valid (33)
```

BCD subtraction : Do using 9's complement or 10's complement
⇓ (EAC) ⇓ (Ignore carry)

8) Double Precision Numbers / Floats :

$x = f \times 10^E$ f: mantissa E: exponent



Boolean Algebra

1) Boolean addition is same as OR
Boolean multiplication is same as AND

2) Absorption Laws : $A + AB = A$ $\quad A + \bar{A}B = A + B$
$A.(A+B) = A$ $\quad A.(\bar{A}+B) = AB$
Also $(A+B)(A+C) = A + BC$

Consensus Laws : $AB + \bar{A}C + BC = AB + \bar{A}C$ [write $BC = BC.1 = BC(A+\bar{A})$]

$(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$ [write $B+C = B+C+0 = B+C+A\bar{A}$]

Proof: $(A+B)(\bar{A}+C)(B+C+A\bar{A})$
$= (A+B)(\bar{A}+C)(B+C+A)(B+C+\bar{A})$
$= (A+B)(A+B+C)(\bar{A}+C)(\bar{A}+C+B)$
$= (A+B)(\bar{A}+C)$ [$\because A.(A+B) = A$].

DeMorgan's Theorems : $\overline{AB} = \bar{A} + \bar{B}$ $\quad \overline{A+B} = \bar{A}.\bar{B}$

for any function complement, do $+ \longrightarrow .$ $\quad . \longrightarrow +$
& complement each of the terms

eg $\overline{(\bar{B}+AC)} = \bar{\bar{B}}.\overline{AC} = B.[\bar{A}+\bar{C}]$

Eg. PT: $BCD + A\bar{C}\bar{D} + ABD = BCD + A\bar{C}\bar{D} + ABC\bar{D}$

$= BCD + A\bar{C}\bar{D} + ABD(C+\bar{C})$ $= BCD + ABCD + A\bar{C}\bar{D} + ABD\bar{C}$

Remember such trick and choose as per guess $= BCD + A\bar{C}(\bar{D} + DB)$

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

$= BCD + A\bar{C}\bar{D} + A\bar{C}B$.

3) SUM OF PRODUCTS & PRODUCTS OF SUMS :-
$(AB + \bar{A}C + A\bar{C})$ $\quad (A+B)(A+C)(\bar{A}+B)$

(a) Minterm (sounds like ∩ so product) : variables appear in complemented form if they are 0 & uncomplemented form if 1.
⇓
product term

For a K-variable function, only 1 minterm has value 1.
rest $2^K - 1$ have value 0.

(b) Canonical SOP expression :

Steps : (i) Retain all the minterms

(ii) For other terms, check for missing variables & multiply $(x+\bar{x})$ for each missing X

(iii) Multiply all the terms & omit redundant terms

eg $Y(A,B) = A+B = A(B+\bar{B}) + B(A+\bar{A}) = AB + A\bar{B} + \bar{A}B$

(c) Maxterm : (sounds like U so (+))

Value = 1 ⇒ complemented form

Value = 0 ⇒ uncomplemented form

Each maxterm is the complement of a ~~row~~ the corresponding minterm

(d) Canonical POS : Add $x\bar{x}$ for each missing X in the term & expand using $A+x\bar{x} \Rightarrow (A+x)(A+\bar{x})$

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

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

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

4) Karnaugh Map :

Systemically simplifying and manipulating expressions.

Try to group together 1's & reduce adjacent cells using $Ax + A\bar{x} = A$. The bigger the group, the more reduction is there.

Example : $Y(A,B,C,D) = m_1 + m_3 + m_5 + m_7 + m_8 + m_9 + m_{12} + m_{13}$

No single 1
No pair
No octet
2 quads.
All is assigned.

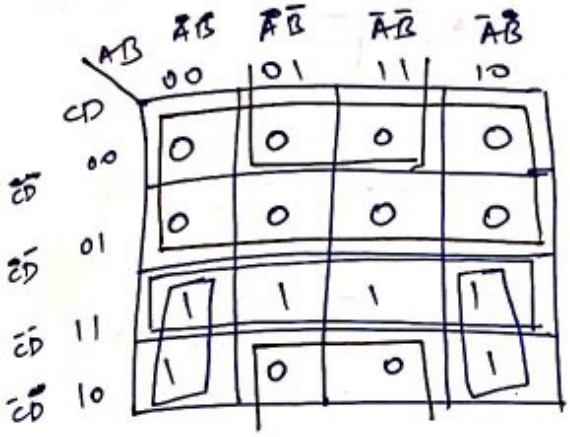
overlap allowed

CD \ AB	$\bar{A}\bar{B}$ 00	$\bar{A}B$ 01	AB 11	$A\bar{B}$ 10
$\bar{C}\bar{D}$ 00	0	0	1	1
$\bar{C}D$ 01	1	1	1	1
CD 11	1	1	0	0
$C\bar{D}$ 10	0	0	0	0

this is redundant as the 1's are already covered

Example: $Y = \Pi(0, 1, 4, 5, 6, 8, 9, 12, 13, 14)$ Maxterm ≡ $\overline{\text{Minterm}}$

∴ given places have 0 in SOP form



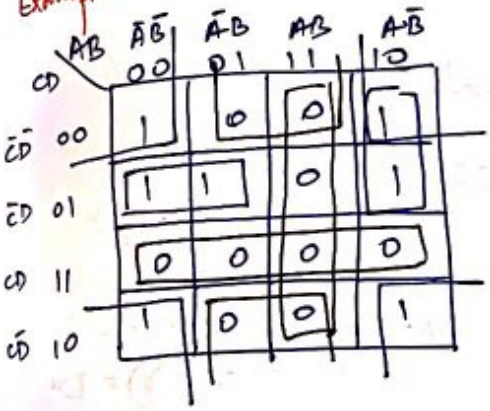
⇒ In POS form, 0 is A
& 1 is A'

octet → c

quad ⇒ $\bar{B}+D$.

∴ $Y = c(\bar{B}+D)$

Example: $F(A, B, C, D) = \Sigma_m(0, 1, 2, 5, 8, 9, 10)$ Write POS & SOP.



SOP = $\underbrace{\bar{B}\bar{D}}_{\text{(quad)}} + \underbrace{\bar{A}\bar{C}D}_{\text{(pair)}} + \underbrace{A\bar{B}\bar{C}}_{\text{(pair)}}$.

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

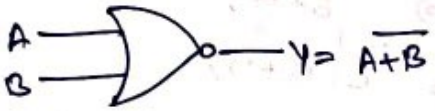
⇒ Logic Gates :

1) OR gate 

AND : 

NOT gate 

NAND : 

NOR = 

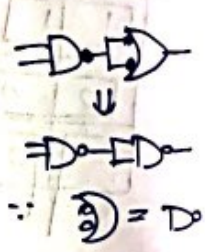
or 

2) NAND & NOR gates are universal gates.

For NAND gate circuit, put in SOP form

Replace 1st level AND by NAND as

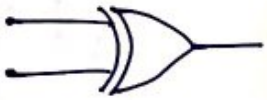
& 2nd level OR by NAND



very important →

FOR NOR gate, put in POS form

& replace all gates by NOR

3) XOR gate = $Y = A \oplus B = A\bar{B} + \bar{A}B$ 

it can perform modulo-2 operation

it can be used as controlled-Inverter eg 

if control = 1 ⇒ $Y = \bar{A}$

if control = 0 ⇒ Y = A

X-NOR gate ∴ 

used for bit comparison (If both bits are similar)

Also parity checking, output is 1 if even no of inputs are 1 else 0.

⇒ Algebra of Logic :

1) Proposition : declarative sentence which is either T or F = do u speak Hindi?
Statement : can be only T or only F eg 3 = 2+4

2) Conjunction : $p \wedge q$ [AND opⁿ]
Disconjunction : $p \vee q$ [OR opⁿ]
Negation : $\sim p$
conditional operator : $p \rightarrow q$
Biconditional : $p \leftrightarrow q$

p	q	$p \rightarrow q$	$p \leftrightarrow q$
0	0	1	1
0	1	1	0
1	0	0	0
1	1	1	1

False → True

3) Tautology : Logical truth / always true.
Contradiction : ~ tautology, ie, always false.
Equivalence : if $p \leftrightarrow q$ is a tautology $p \Leftrightarrow q$

All tautologies are equivalent to each other & all contradictions are equivalent to each other.

4) Some equivalence formulae : $p \Rightarrow p \vee q$ $q \Rightarrow p \vee q$
(Remember through Venn diagram) $p \wedge q \Rightarrow p$ $p \wedge q \Rightarrow q$
$(p \vee q) \wedge \sim p \Rightarrow q$ $(p \vee q) \wedge \sim q \Rightarrow p$.
$(p \rightarrow q) \wedge (q \rightarrow r) \Rightarrow (p \rightarrow r)$
$p \rightarrow q \Rightarrow \sim q \rightarrow \sim p$
$p \leftrightarrow q \Rightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
$p \rightarrow q \Rightarrow \sim p \vee q$

5) Predicates : variable stmt which becomes specific when particular values are assigned to variable.

eg $P(x) : x > 7$

Universal quantification : $\forall x\, P(x)$ ⇓ $P(x_1) \wedge P(x_2) \wedge \dots \wedge P(x_n)$
Existential qualification : $\exists x\, P(x)$ ⇓ $P(x_1) \vee P(x_2) \vee \dots \vee P(x_n)$

6) Normal Form:

Disjunctive normal form: sum of elementary products (Try to get in $\vee$ form (SOP form))

eg. $\sim(p \vee q) \leftrightarrow (p \wedge q)$

$E \Leftrightarrow F \Leftrightarrow (E \wedge F) \vee (\sim E \wedge \sim F)$

$\Rightarrow [\sim(p \vee q) \wedge (p \wedge q)] \vee [(p \vee q) \wedge \sim(p \wedge q)]$

$= [(\sim p \wedge \sim q) \wedge (p \wedge q)] \vee [(p \vee q) \wedge (\sim p \vee \sim q)]$

$= [(p \vee q) \wedge \sim p] \vee [(p \vee q) \wedge \sim q]$

$= [p \wedge \sim p] \vee [q \wedge \sim p] \vee [p \wedge \sim q] \vee [q \wedge \sim q]$

$= (q \wedge \sim p) \vee (p \wedge \sim q)$

Conjuctive Normal form: product of elementary sums (POS form)
Try to get in $\wedge$ form

$E \leftrightarrow F \Leftrightarrow (\sim E \vee F) \wedge (\sim F \vee E)$

Principal forms: terms are either minterms or maxterms only

eg. pdnf for $p \vee q \Rightarrow [p \wedge (q \vee \sim q)] \vee [q \wedge (p \vee \sim p)]$

$= (p \wedge q) \vee (p \wedge \sim q) \vee (q \wedge p) \vee (q \wedge \sim p)$

$= (p \wedge q) \vee (p \wedge \sim q) \vee (q \wedge \sim p)$

Example: write pcnf & pdnf for $p \rightarrow [(p \rightarrow q) \wedge \sim(\sim q \vee \sim p)]$

(i) pdnf:

$p \rightarrow [(p \rightarrow q) \wedge \sim(\sim q \vee \sim p)]$

$= \sim p \vee [(\sim p \vee q) \wedge (q \wedge p)]$

$= \sim p \vee [\{\sim p \wedge (q \wedge p)\} \vee \{q \wedge (q \wedge p)\}]$

$= \sim p \vee (q \wedge p)$

$= (\sim p \vee (q \vee \sim q)) \wedge (q \wedge p)$

$= (\sim p \wedge q) \vee (\sim p \wedge \sim q) \vee (q \wedge p)$

(ii) pcnf:

$p \rightarrow [(p \rightarrow q) \wedge \sim(\sim q \vee \sim p)]$

$\sim p \vee [(\sim p \vee q) \wedge (q \wedge p)]$

$= (\sim p \vee \sim p \vee q) \wedge (\sim p \vee (q \wedge p))$

$= (\sim p \vee q) \wedge (\sim p \vee q) \wedge (\sim p \vee p)$

$= (\sim p \vee q)$

⇒ Algorithms :

Input → Assignment → Decision → Repetition → Output

* BISECTION METHOD :

1. Read x_0, x_1
2. Read ϵ
3. $y_0 \leftarrow f(x_0)$
4. $y_1 \leftarrow f(x_1)$
5. $i \leftarrow 0$
6. if $y_0 y_1 > 0$, follow 14 to 16
   else follow 7 to 14
7. if $|(x_0 - x_1)/x_1| > \epsilon$, follow 8 to 11
8. $x_2 \leftarrow (x_0 + x_1)/2$
9. $y_2 \leftarrow f(x_2)$
10. $i \leftarrow i+1$
11. if $y_0 y_2 > 0$ $x_0 \leftarrow x_2$
    else $x_1 \leftarrow x_2$
12. Print 'Solution cgt to a root'
13. Print i, x_2, y_2 : goto 16
14. Print 'Initial values unsuitable'
15. Write x_0, x_1, y_0, y_1
16. Stop.

* GAUSS-ELIMINATION METHOD

1. Read n
2. Read matrix a_{ij}
3. Rearrange st $a_{11} \neq 0$
4. Eliminate x_1 from all eqn by normalising 1st eqn by a_{11}
5. Subtract $j1$ x 1st equation from j^{th} equation. Repeat for $j = 2$ to n
6. Eliminate x_2 similarly. Rearrange if $c_{22} = 0$
7. Get x by back substitution.

* REGULA FALSI METHOD

1. Read x_0, x_1
2. Read ϵ & max it^n n
3. $f_0 \leftarrow f(x_0)$
4. $f_1 \leftarrow f(x_1)$
5. for $i = 1$ to n
6. $x_2 \leftarrow (x_0 f_1 - x_1 f_0)/(f_1 - f_0)$
7. $f_2 \leftarrow f(x_2)$
8. If $|f_2| < \epsilon$, print 'sol^n is cgt' & print x_2, f_2 & go to 13.
   else go to 9.
9. if $f_2 f_0 < 0$ $x_1 \leftarrow x_2$, $f_1 \leftarrow f_2$
   else $x_0 \leftarrow x_2$ $f_0 \leftarrow f_2$
10. Next i
11. Print 'Doesn't converge in n iterations'
12. Print x_2, f_2
13. Stop.

* GAUSS-JORDAN METHOD

1. Read n
2. Read matrix a_{ij}
3. check all the pivot values Rearrange if they are zero
4. Normalise rows
5. Eliminate unknown
6. Print Results.

* NEWTON-RAPHTSON METHOD

1. Read x_0
2. Evaluate $f(x_0), f'(x_0)$
3. $x_1 \leftarrow x_0 - \frac{f(x_0)}{f'(x_0)}$
4. If $|x_1 - x_0| < \epsilon$ go to 6 else continue
5. $x_0 \leftarrow x_1$ Go to 3
6. Stop.

* GAUSS-SIEDEL METHOD

1. Read n, a_{ij}, b_i
2. Read initial x_i
3. If $a_{ii} = 0$, ~~goto~~ print 'a_{ii} is 0' & goto 6
4. Start iterations to calculate $x_j = \frac{b_i - S}{a_{jj}}$ $i = 1$ to n where $S = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j$
5. If largest difference b/w successive $x_i < \epsilon$, then go to 7
6. Repeat iteration 4.
7. Print Root
8. Stop

* Newton's Forward/Backward Formula

1. Read n, x_i & y_i
2. Read x (say a)
3. Construct difference table.
4. Compute $u = (a - a_0)/h$
5. Expand N-F/B formula & substitute the values
6. Print the value at $x = a$
7. If you want to repeat go to 2.
8. Stop.

* Euler's method:

1. Read a, b, y
2. Read no of pts n
3. Find $h = (b-a)/n$
4. Assign $x = a$
5. Assign $i = 1$
6. If $x >= (b+h)$, goto 9 o/w goto 7
7. Evaluate Eulers formula $y_{i+1} = y_i + hf(x_i, y_i)$ & print y_{i+1}
8. Increment i & goto 6. & $x = x+h$
9. Stop

* Lagrange Method

Same way as N-F/B.

* Simpson's 1/3rd Rule:

1. Read Initial value a
2. Read final value b
3. Read no of divisions n
4. Find width = $h = \frac{b-a}{n}$
5. Assign $m = \frac{n}{2}$
   $s = 0$ & $x = a$
6. Expand simpsons Rule as $\frac{h}{3}(y_1 + 4y_2 + 2y_3 + 4y_4 + 2y_5 + \dots + 2y_{n-1} + 4y_n + y_{n+1})$
7. put the values & evaluate
8. Print result
9. Stop.

* Runge-Kutta Method:

1. Read initial values a, b, y
2. Read no of points 'n'
3. Find h.
4. Assign $x = a$
5. Compute k_1, k_2, k_3, k_4 for RK formula & find $y = y + \frac{(k_1 + 2k_2 + 2k_3 + k_4)}{6}$
6. Increment $x = x + h$
7. Continue 5 & 6 till we get to $x_n = b$
8. Stop

* Trapezoidal rule

1. Read x_1, x_2 end & ϵ
2. $h \leftarrow x_2 - x_1$
3. $S_1 \leftarrow f(x_1) + f(x_2)/2$
4. $I_1 \leftarrow h \times S_1$
5. $S_0 \leftarrow 0$
   $I_0 \leftarrow 0$
6. $i \leftarrow 1$
7. If $\left|\frac{I_1 - I_0}{I_1}\right| < \epsilon$ go to 16. else continue
8. $S_0 \leftarrow S_1$
9. $x \leftarrow x_1 + h/2$
10. for $j = 1$ to i
    $S_1 \leftarrow S_1 + f(x)$
    $x \leftarrow x + h$
    next j
11. $i \leftarrow 2 \times i$
12. $h \leftarrow h/2$
13. $I_0 \leftarrow I_1$
14. $I_1 \leftarrow h \times S_1$
15. Print I_1, h
16. Stop

Flow Chart :

(i) Assignment

(ii) Decision

(iii) Input / output

(iv) Terminal

(v) off page connector

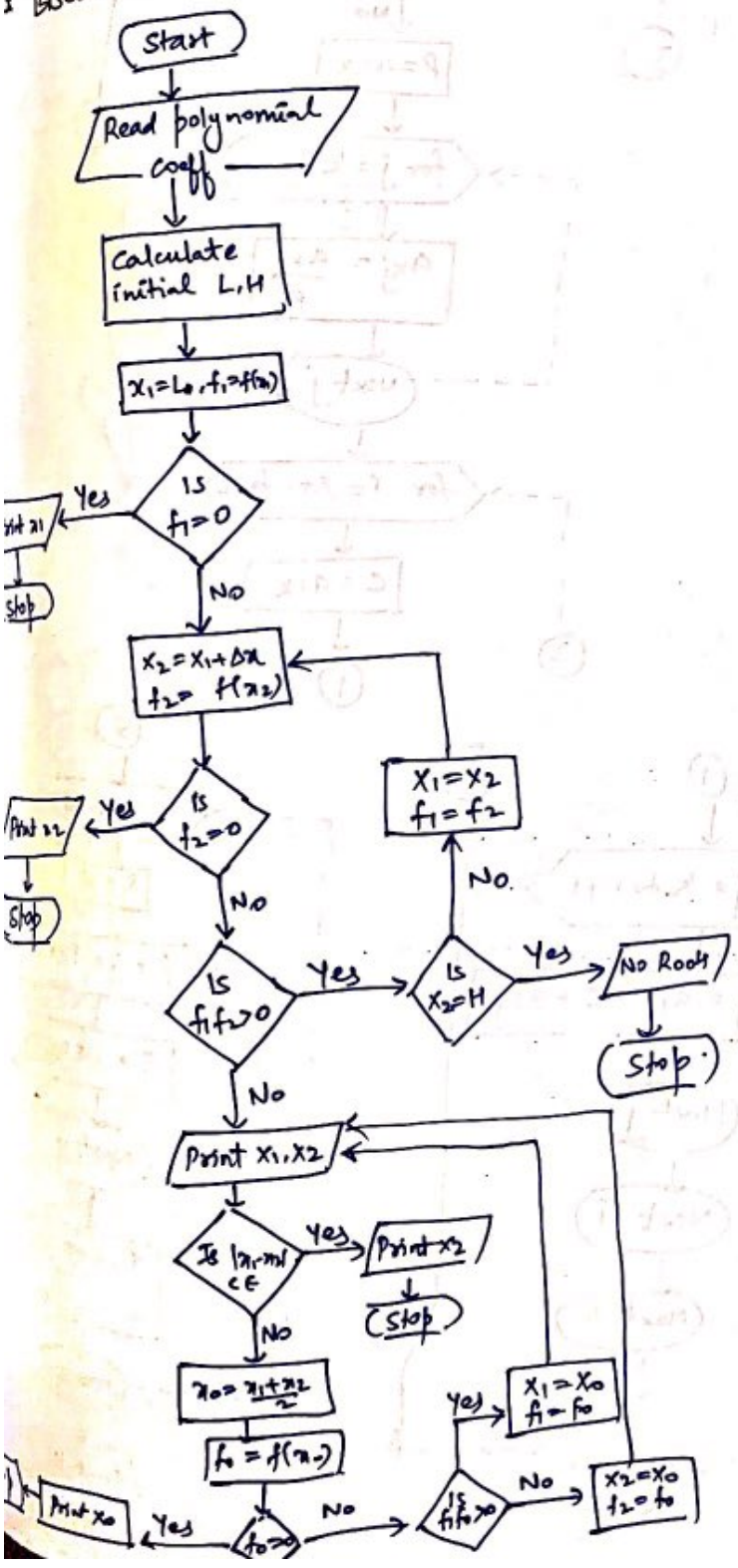
(vi) Modification

(vii) Group instruction

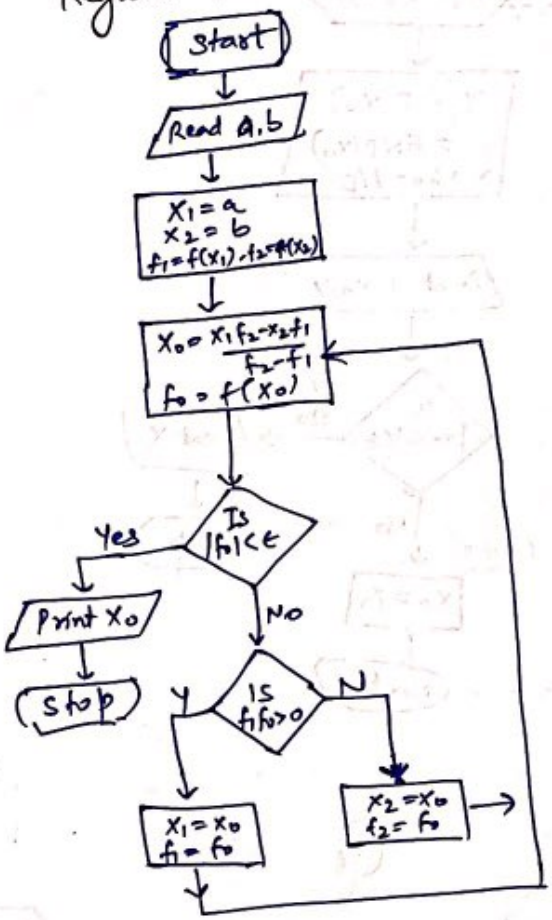
(viii) Connection

(ix) Module

* Bisection Method

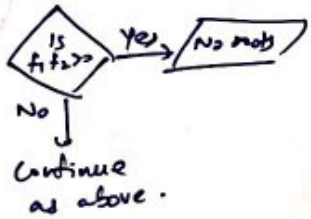


* Regula Falsi

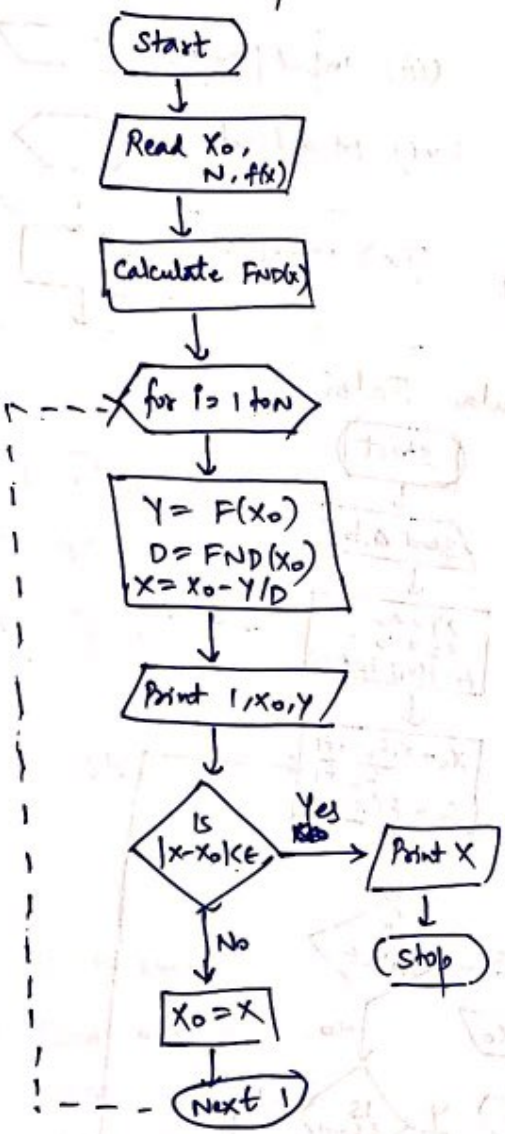


This chart assumes presence of a root b/w a & b.

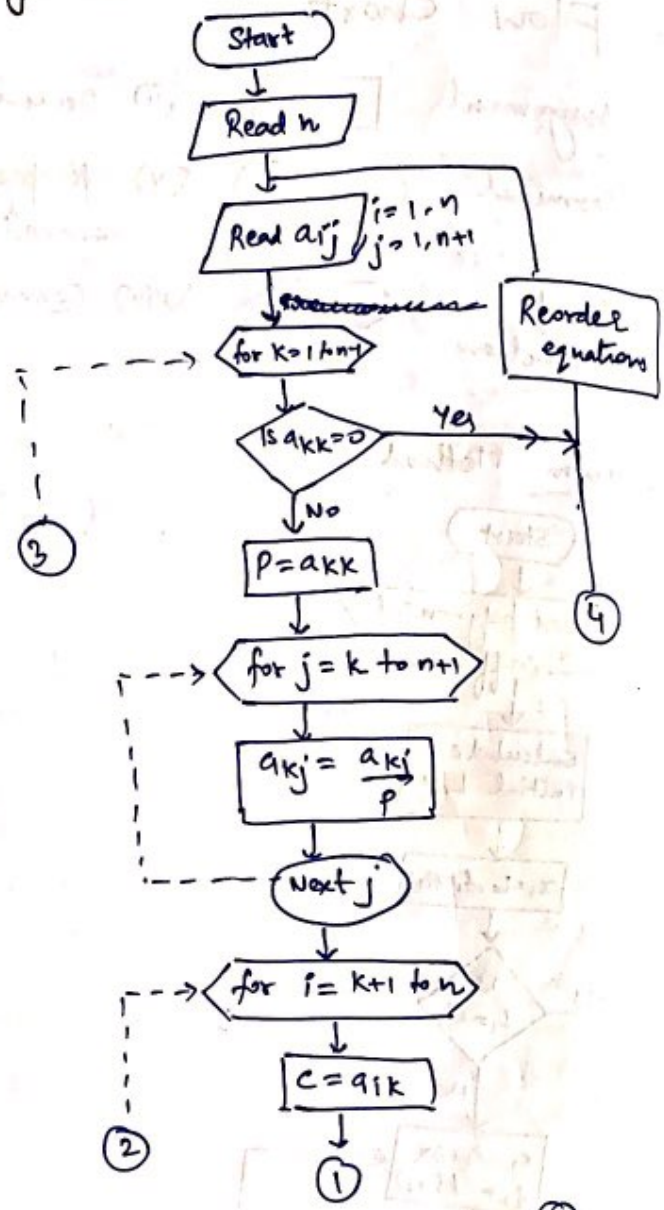
Otherwise add another decision symbol at the start as filter

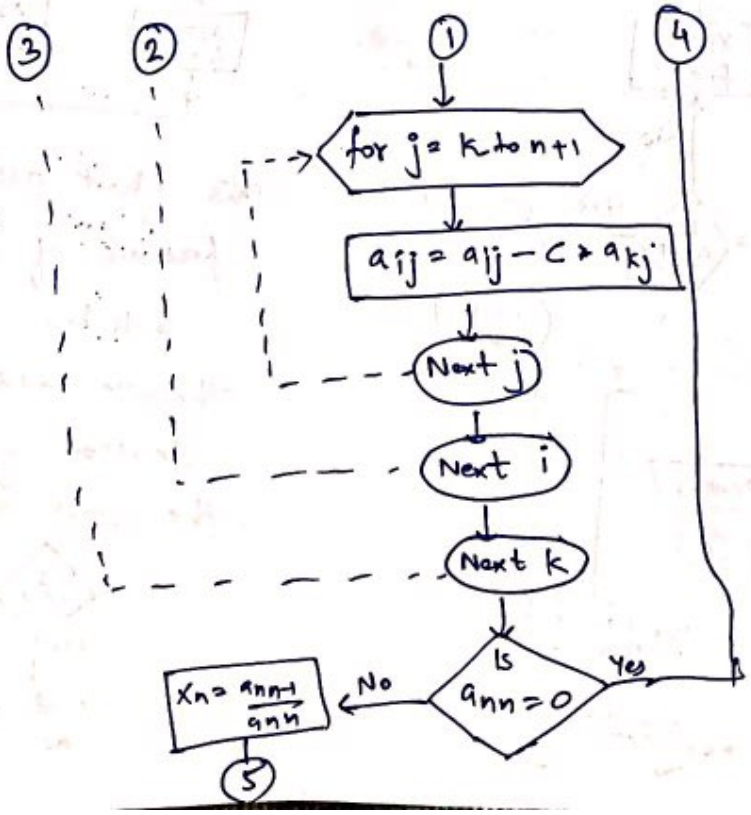


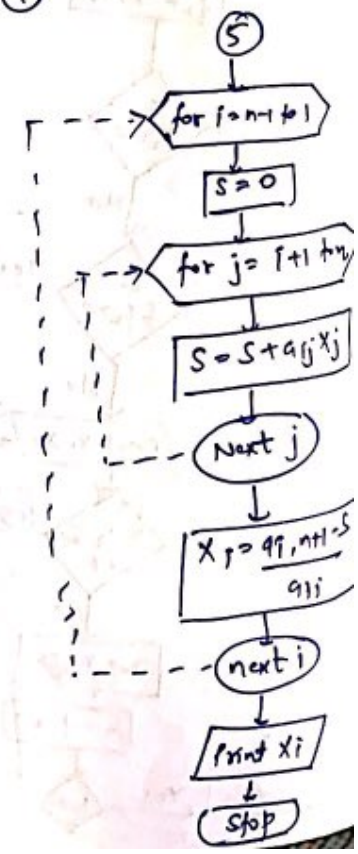
* Newton - Raphson



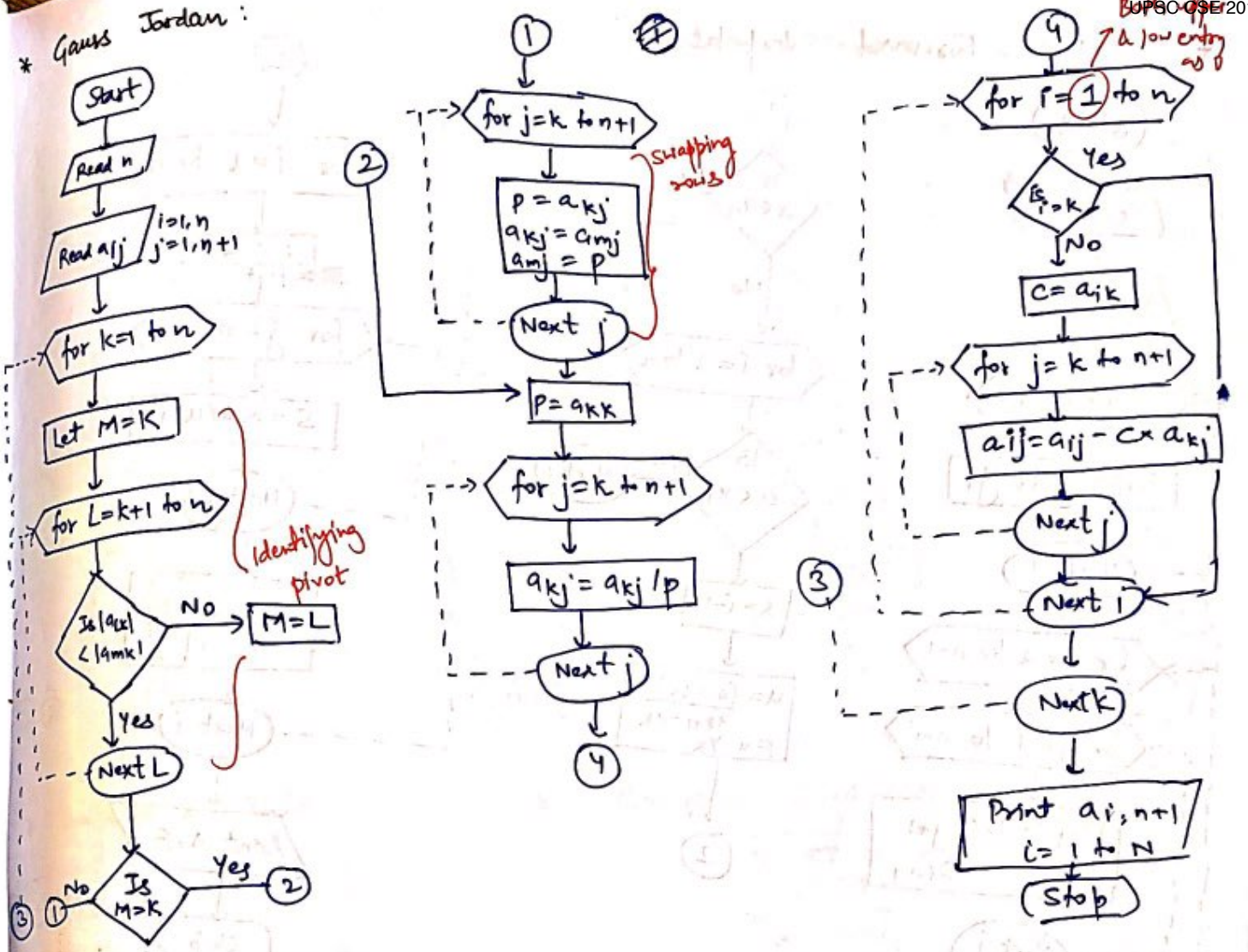
* Gauss - Elimination



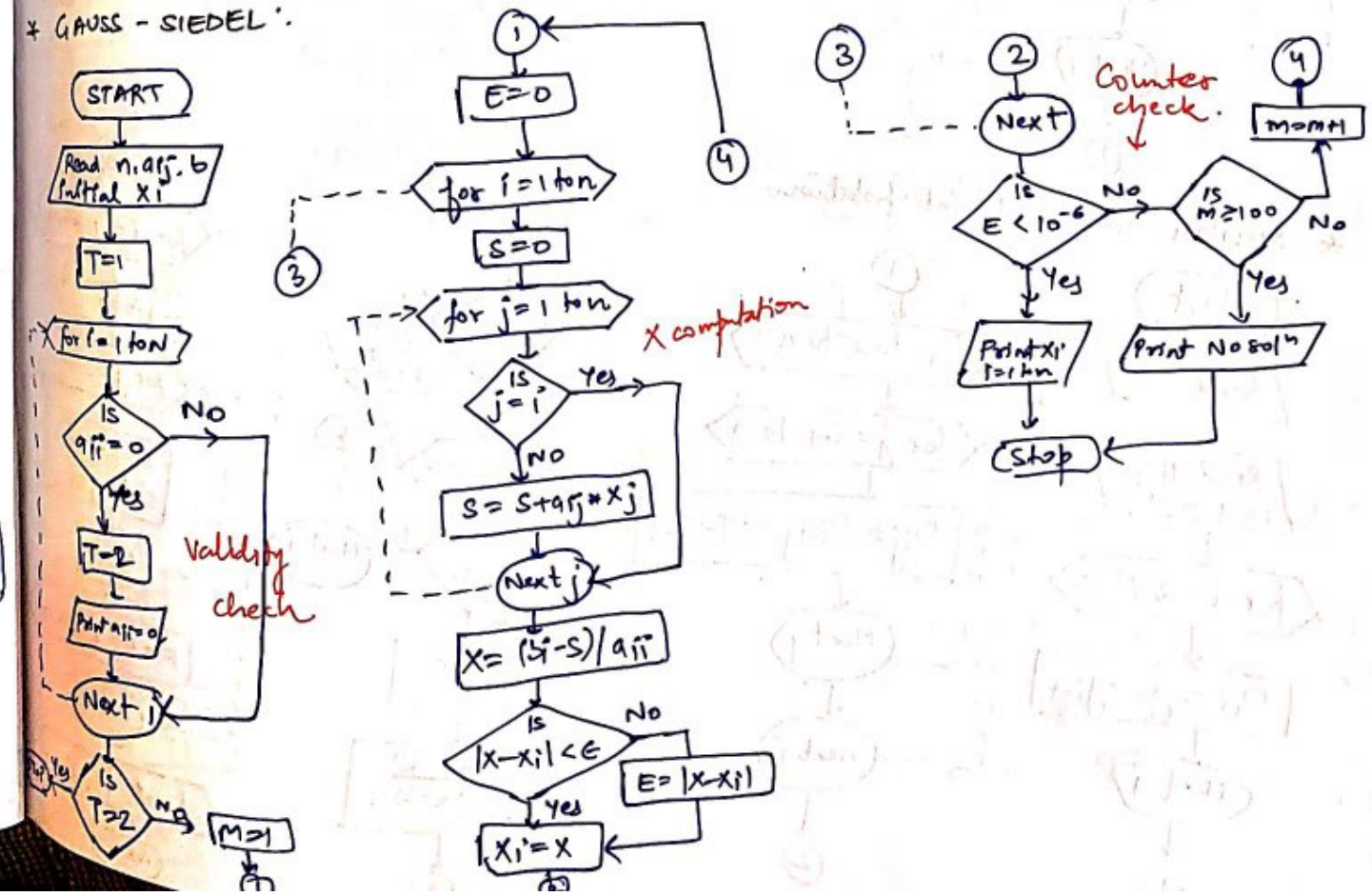




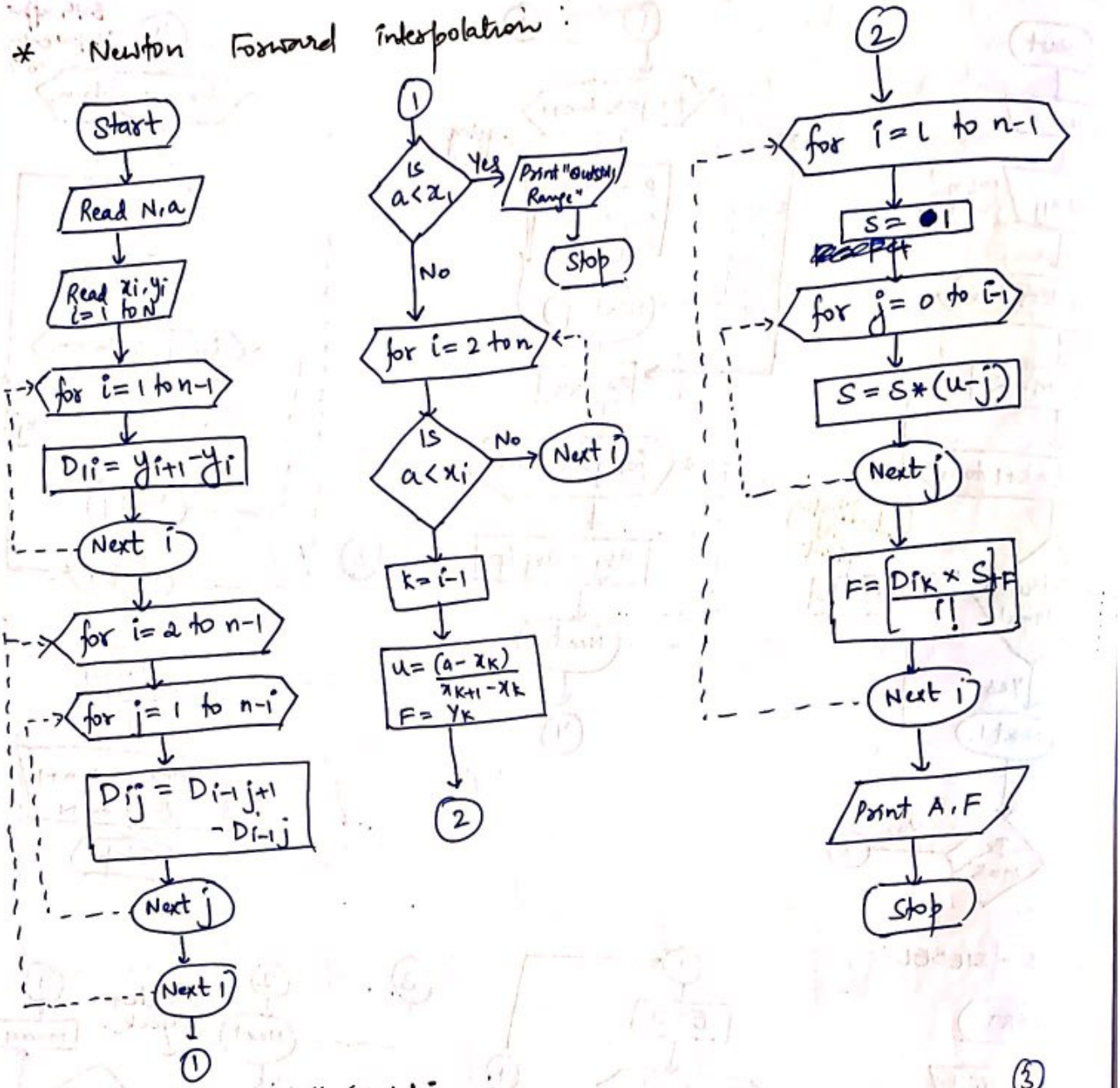
* Gauss Jordan :



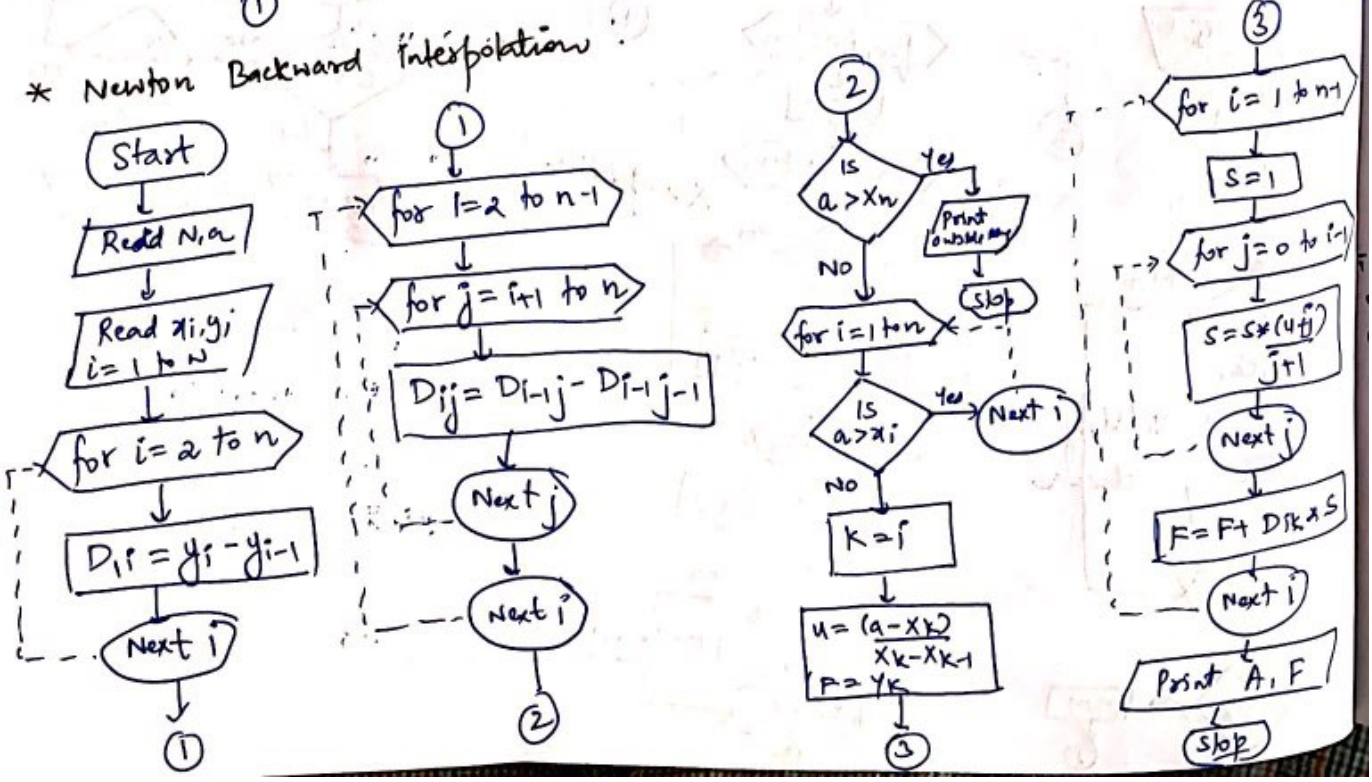
* GAUSS - SIEDEL :



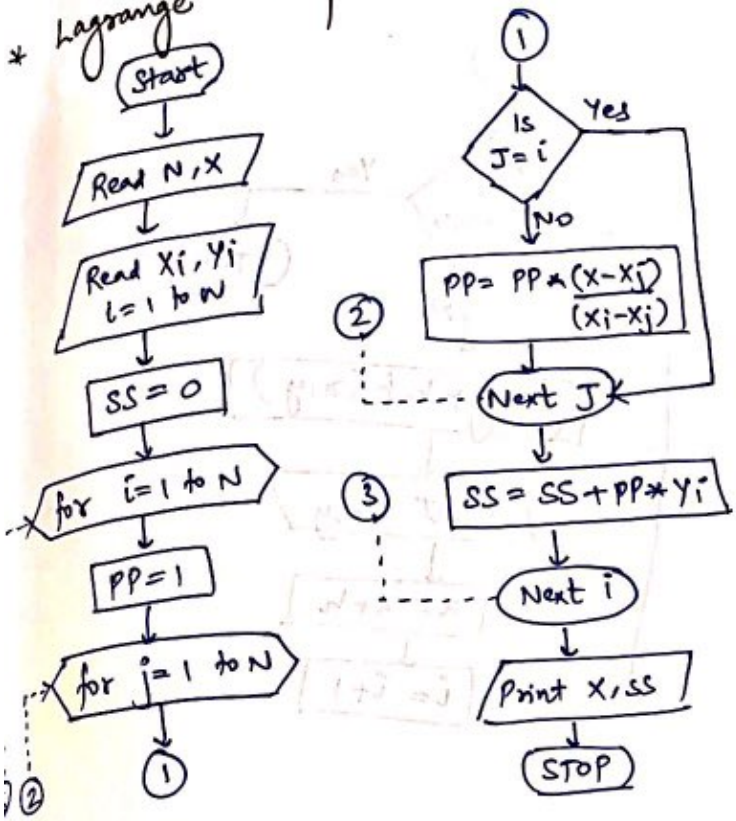
* Newton Forward interpolation :



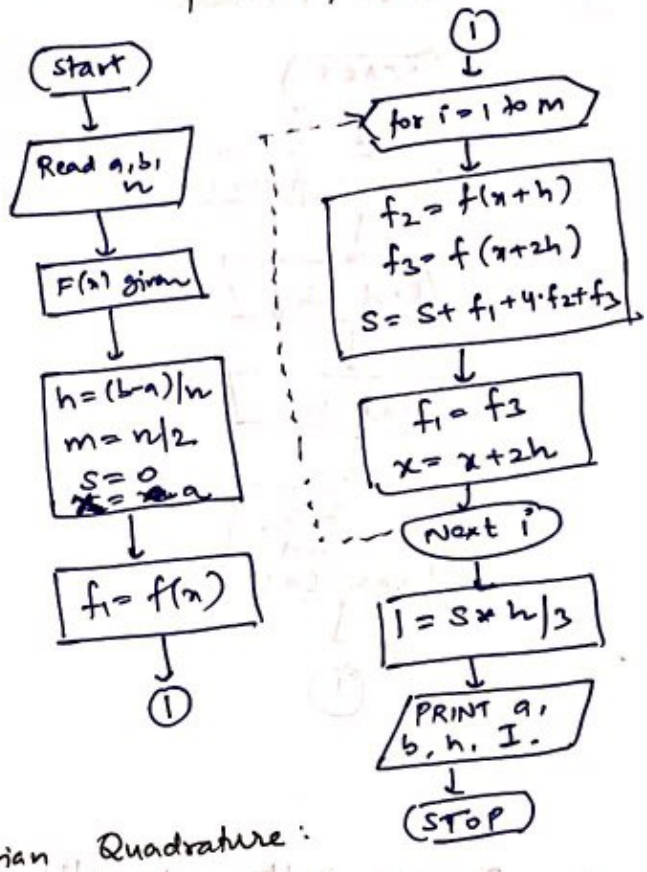
* Newton Backward interpolation :



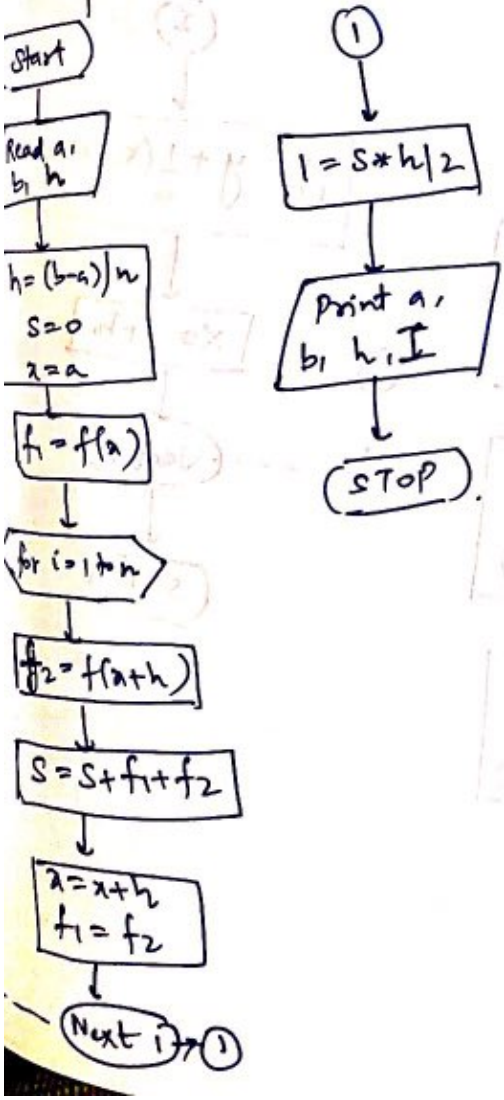
* Lagrange Interpolation :



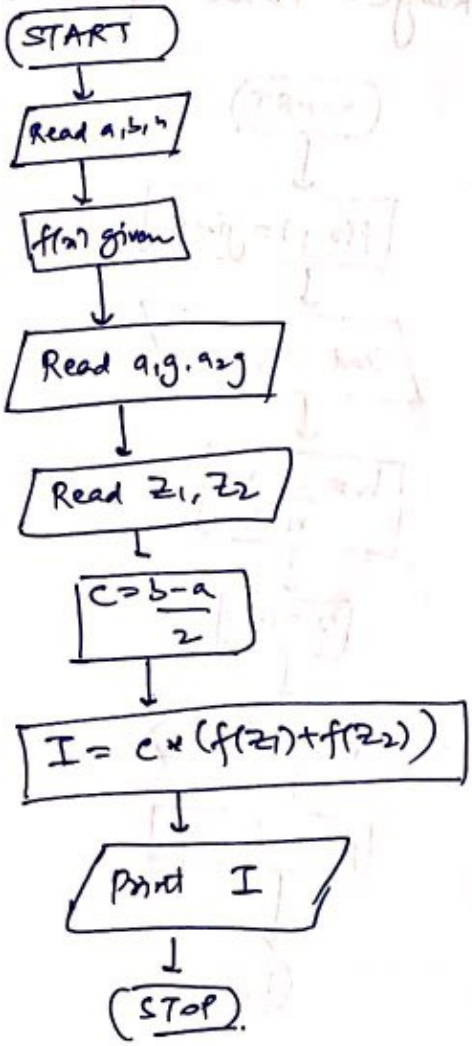
* Simpsons 1/3rd Rule



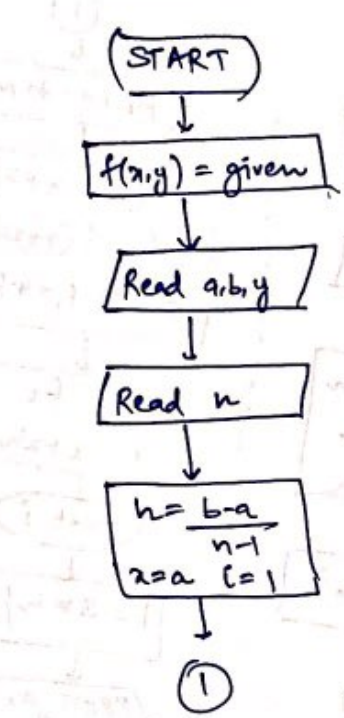
* Trapezoidal Rule

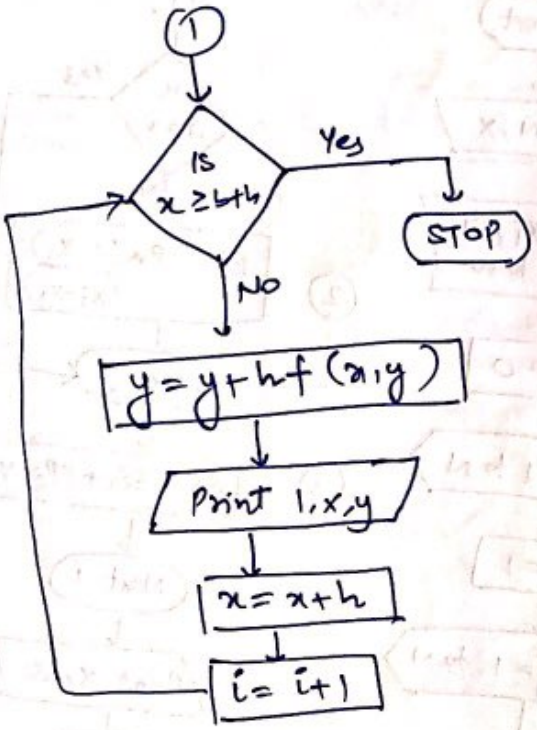


* Gaussian Quadrature :



* Euler's Method :





* Runge- Kutta of order 4 :