1] Write an algorithm, Draw a Flowchart and write a program in CPP to implement Logic Gates.

Algorithm

Step1 :- Start

Step2 :- char menu
int result
int dataValue1
int dataValue2

Step3 :- Display "enter your Boolean operator code(A,O,N,X)

Step4 :- Input menu

Step5 :- switch(menu)

Case 'A':-

i] Display "Enter first Boolean value."

ii] Input dataValue1

iii] Display" Enter second Boolean value"

iv] Input dataValue2

v] If(dataValue1==1 && dataValue2==1)

vi] Result=1

else

result=0

vii] pisplay "show result"

viii] Input result

case O :-

i] Display "Enter first Boolean Value"

ii] Input dataValue1

iii] Display "Enter second Boolean value"

iv] Input dataValue2

v] If (dataValue1 == 1 || dataValue2 == 1)

Yes :- result = 1

NO :- result = 0

vi] Display "show result"

vii] Input result

break

case N :-

i] Display "Enter first Boolean Value"

ii] Input dataValue1

iii] result = ! dataValue 1

iv] Display "show result"

v] Input result

Break

case X :-

i] Display "Enter first Boolean value"

ii] Input dataValue1

iii] Display "Enter second Boolean value"

iv] Input dataValue2

Flowchart

v] If (dataValue1 = ! dataValue1)

Yes :- result = 1

No :- result = 0

vi] Display " show result "

vii] Input result

default

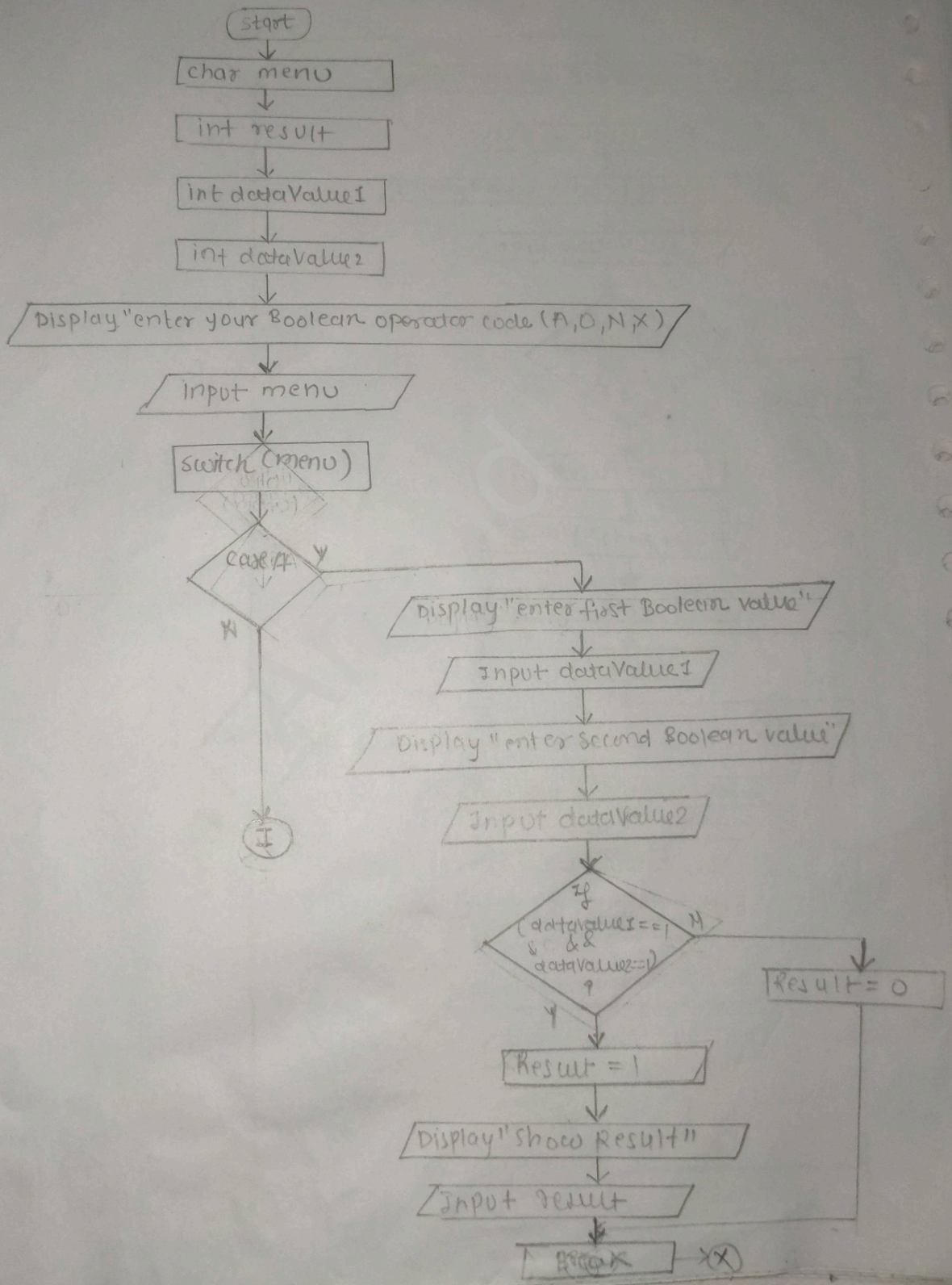
i] result = 0

Step6 :- Input Ignore(2)

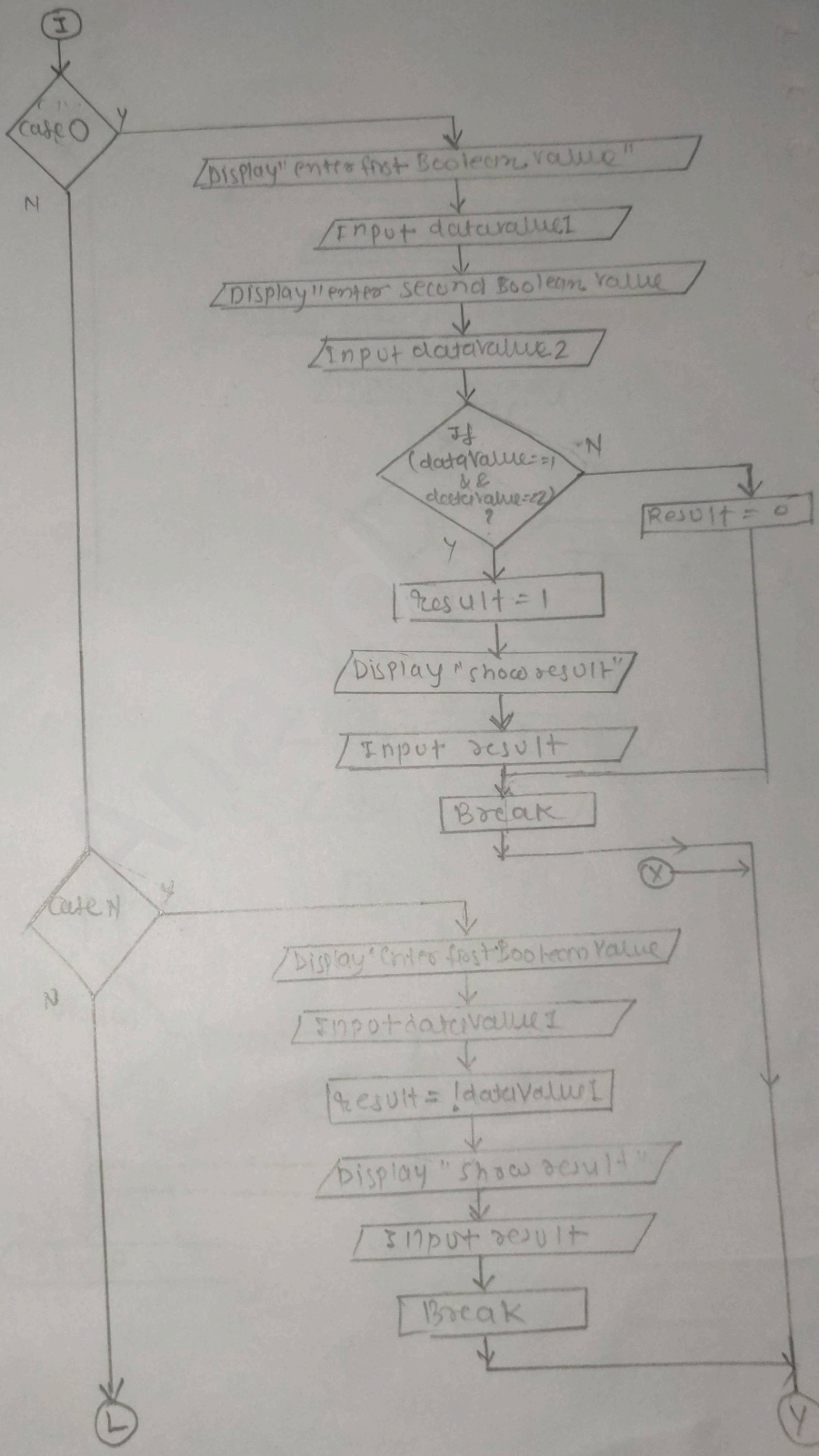
~~Step7 :- return 0~~

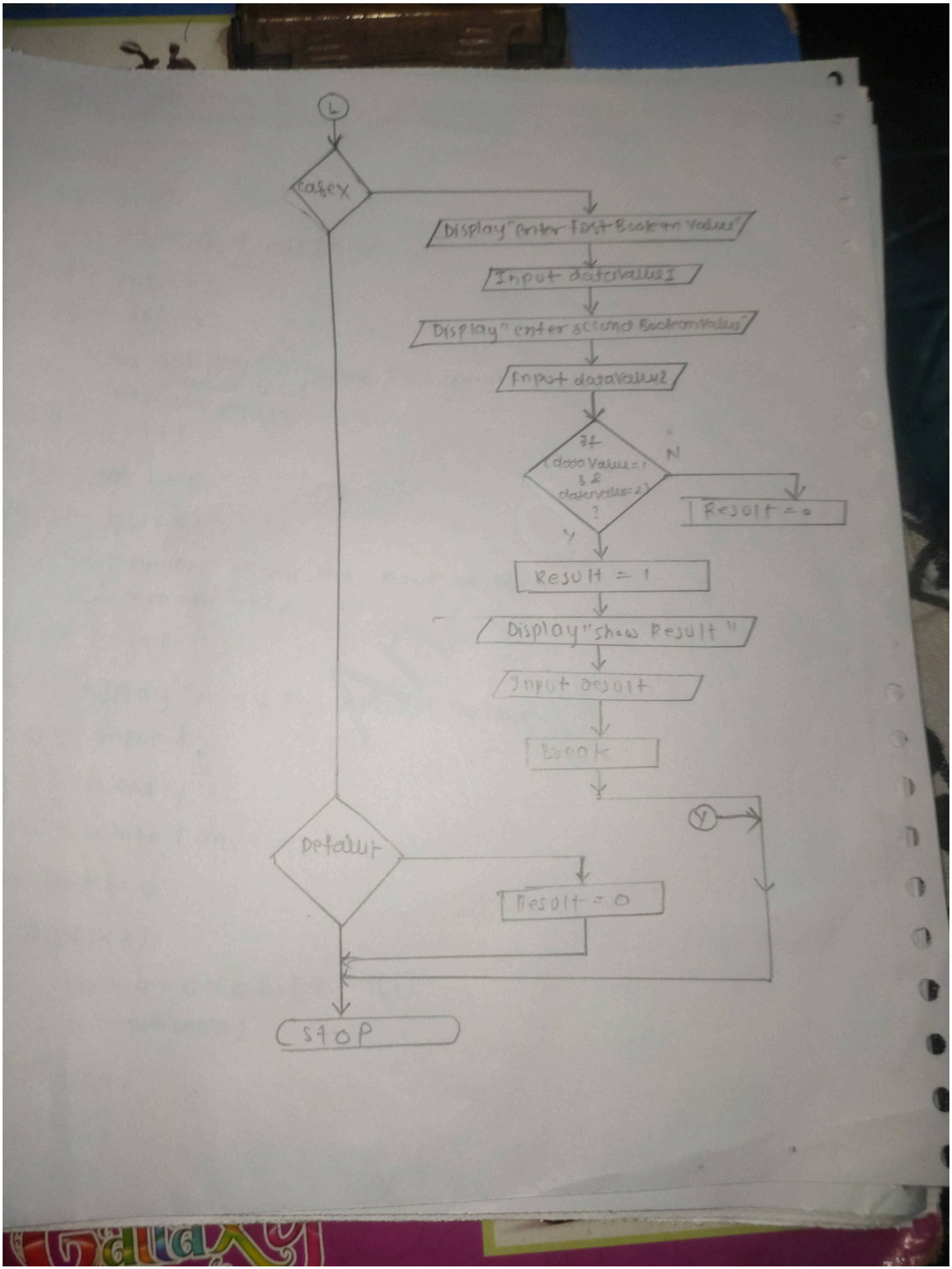
Step8 :- stop

Flowchart


start
char menu
int result
int dataValue1
int dataValue2
Display "enter your Boolean operator code (A,O,N,X)
Input menu
switch (menu)
case A
Y
N
Display "enter first Boolean value"
Input dataValue1
Display "enter second Boolean value"
Input dataValue2
I
If (dataValue1==1 && dataValue2==1) ?
N
Y
Result = 0
Result = 1
Display "show Result"
Input result
Break
X

4] Write an algorithm




L
case x
Display "Enter first Boolean value"
Input datavalue1
Display "enter second Boolean value"
Input datavalue2
If (datavalue1 = 1 && datavalue2 = 2) ?
N
Result = 0
Y
Result = 1
Display "show Result"
Input result
Break
Y
Default
Result = 0
STOP

4] Write an algorithm, Draw a Flowchart and write a program in CPP to implement perceptron Training Algorithm.

Algorithm:-

step1:- start

step2:- int in[3], d, w[3], q=0

step3:- int i=0

step4:- if(i<3)

Yes:- i) Display "initialize the weight vector w
ii) Input w[i]

No:- goto step(13)

step5:- i=i+1

step6:- int i=0

step7:- if(i<3)

Yes:- Display" enter the input vector i

NO:- goto step(13)

step8:- i=i+1

step9:- Display "enter the desired output

step10:- input d

step11:- int ans=1

step12:- while (ans==1)

Yes:- i) Int i=0

ii) if (i<3)

Yes:- q=q+w[i]*int[i]

NO:- goto step(13)

iii) i=i+1

iv] Display "Desired output is"

v] Display d

vi] Display "actual output is"

vii] Display q

viii] int e

ix] e = d - q

x] Display "error is"

xi] Display e

xii] Display "press 1 to adjust weight else 0"

xiii] Input ans

xvi] If (e<0)

Yes :- i] int i = 0

ii] If (i<3)

Yes :- w[i] = w[i] - 1

No :- goto step(13)

iii] i = i + 1

NO :- If (e > 0)

Yes :- i] int i = 0

ii] If (i<3)

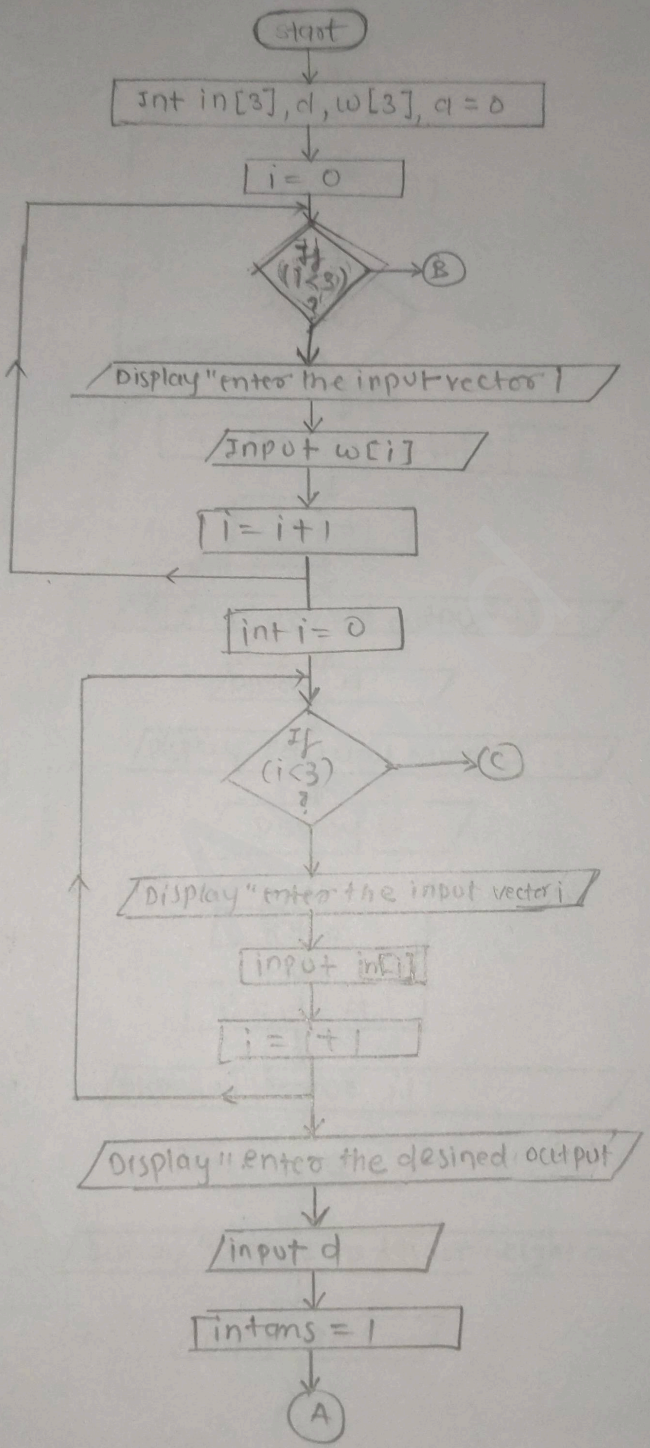
Yes :- w[i] = w[i] + 1

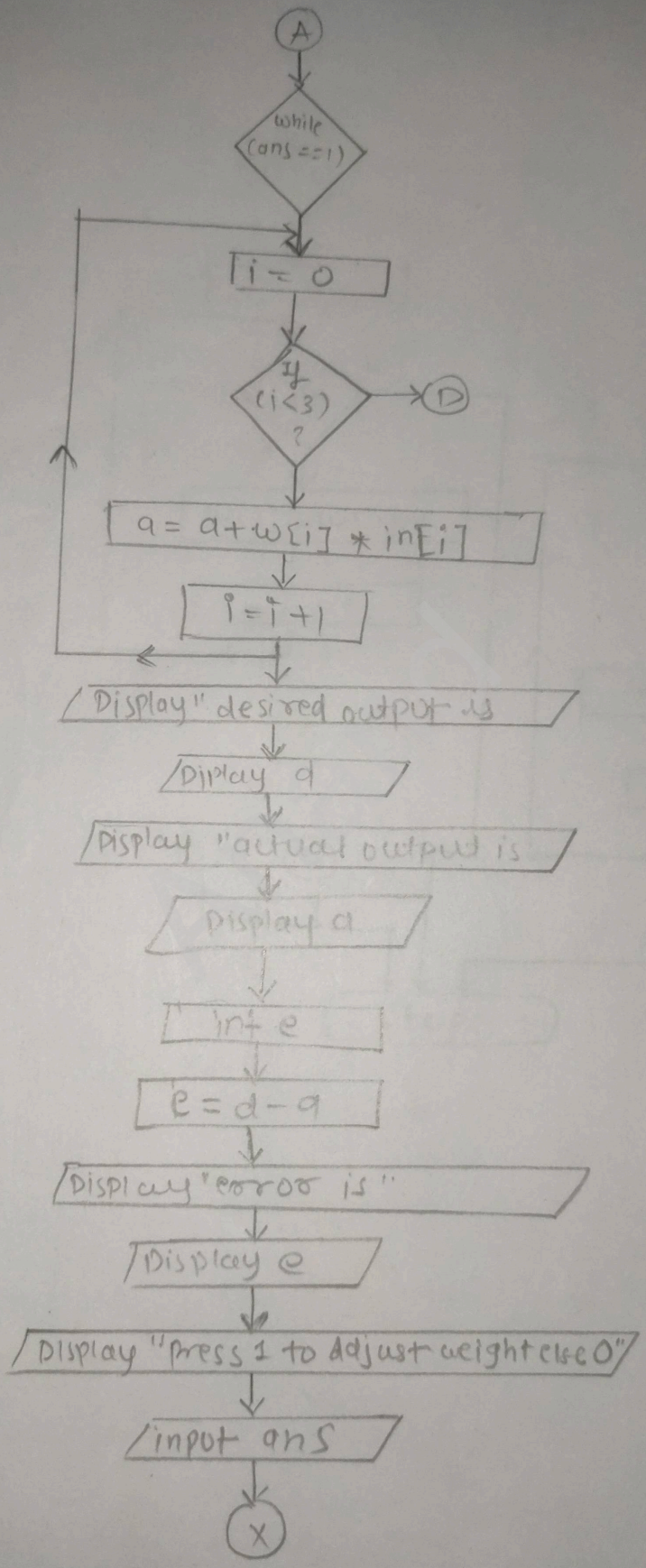
No :- goto step(13)

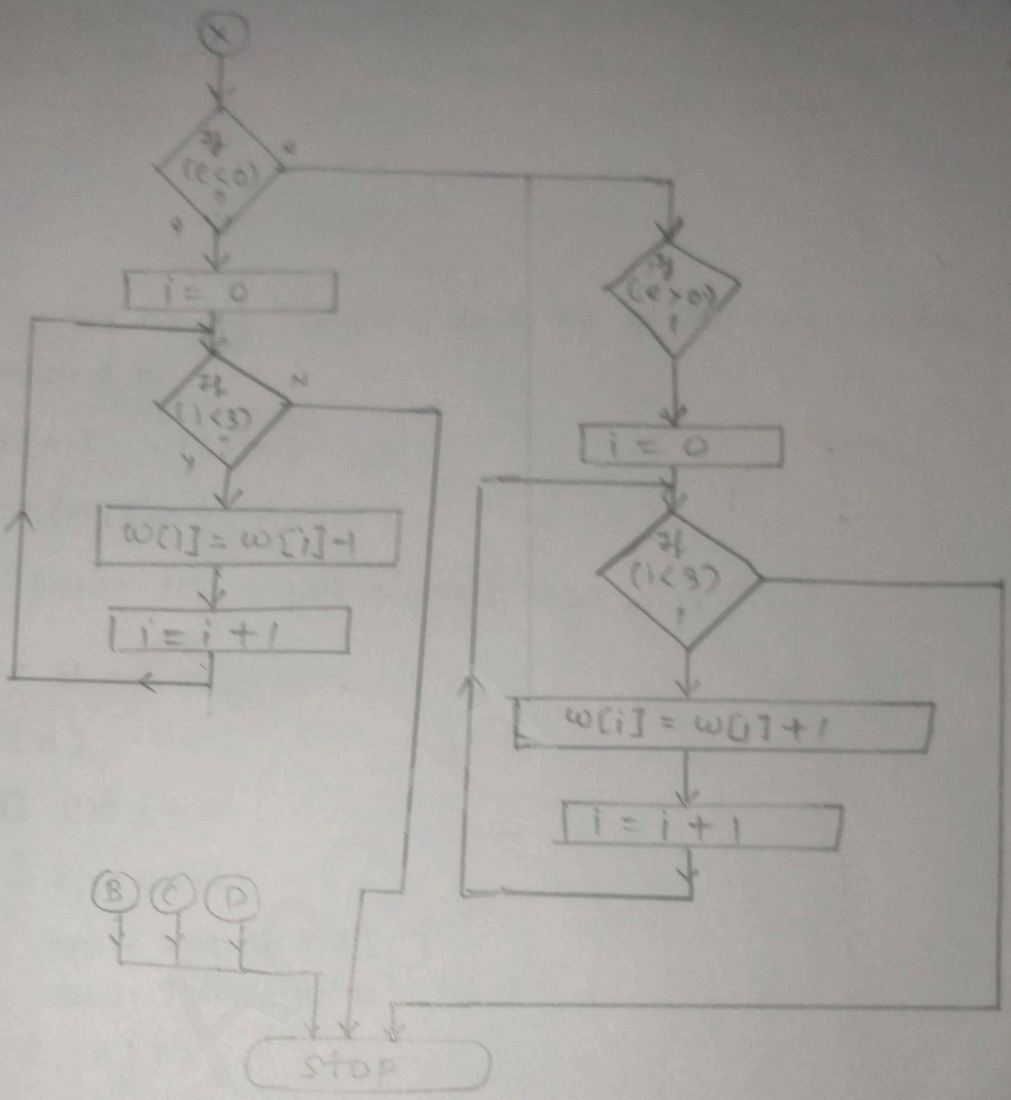
iii] i = i + 1

step13 :- Stop

Flowchart


start
int in[3], d, w[3], a = 0
i = 0
If (i<3) ?
B
Display "enter the input vector"
Input w[i]
i = i+1
int i = 0
If (i<3) ?
C
Display "enter the input vector i
input in[i]
i = i+1
Display "enter the desired output
input d
intams = 1
A


A
while (ans==1)
i = 0
If (i<3) ?
D
a = a+w[i] * in[i]
i = i+1
Display "desired output is
Diplay d
Display "actual output is
Display a
int e
e = d-a
Display "error is "
Display e
Display "press 1 to Adjust weight else 0"
input ans
X


A
if (e<0)
Y
N
i = 0
if (i<3)
Y
w[i] = w[i]-1
i = i+1
if (e>0)
Y
i = 0
if (i<3)
Y
w[i] = w[i]+1
i = i+1
B
C
D
STOP

5] Write an algorithm, Draw a flowchart and write a program in CPP to implement Hebb's rule.

Algorithm:

Step1 :- Start

Step2 :- int m, n

Step 3 :- Display " enter no. of features and no. of training datasets

Step4 :- Input m and n

Step5 :- int wt1[m], wt2[m]

Step6 :- int input[n][m]

Step7 :- Display " enter the input matrix row wise "

Step8 :- Int i = 0

Step 9 :- If(i<n)

yes :- i] int j = 0

ii] If(j<m)

yes :- input[i][j]

No :- goto step(34)

iii] j = j+1

~~step10 variable~~

No :- goto step(34)

Step 10 :- i = i+1

step 11 :- int target1[n], target2[n]

step12 :- Display " enter the target in binary : "

step13 :- int i = 0

Step 14 :- If(i<n)

yes :- Input target1[i]

No :- goto step(34)

step15 :- i = i + 1

step16 :- Display "enter the target in bipolar :

step17 :- Int i = 0

step18 :- If (i < n)

Yes :- Input target2[i]

No :- goto step (34)

step19 :- i = i + 1

step20 :- int i = 0

step21 :- If (i < m)

Yes :- i] wt1[i] = 0

ii] wt2[i] = 0

No :- goto step (34)

step22 :- i = i + 1

step23 :- Int j = 0

step24 :- If (j < n)

Yes :- i] Display "########j = "

ii] Int i = 0

iii] If (i < m)

Yes :- i] wt1[i] += (input[j][i] * target1[j]

ii] Display "weight1 at i = " << i << " is " << wt1[i]

iii] wt2[i] += (input[j][i] * target2[j])

iv] Display "wt2 at i = " << i << " is " << wt2[i]

No :- goto step (34)

iv] i = i + 1

No :- goto step (34)

step25 :- j = j + 1

step26 :- Display " ***** output ***** \nafter 1 epochs
binary weights".

step27 :- int i = 0

step28 :- if (i<m)

Yes :- Display " wt1[i]

No :- goto step (34)

step29 :- i+1 = i

step30 :- Display "\nafter 1 epoch : bipolar weights:"

step31 :- int i = 0

step32 :- if (i<m)

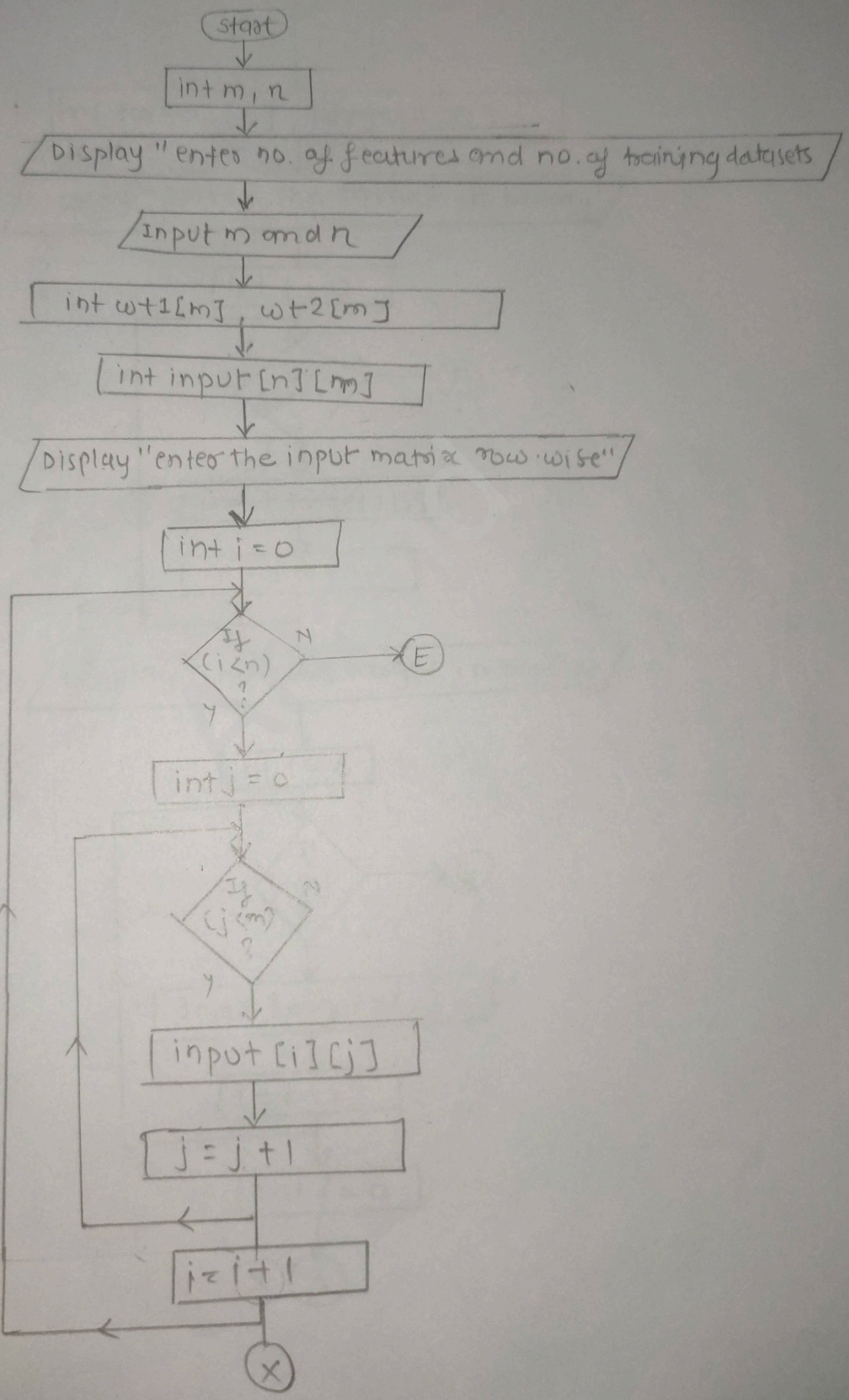
Yes :- Display " wt2[i]"

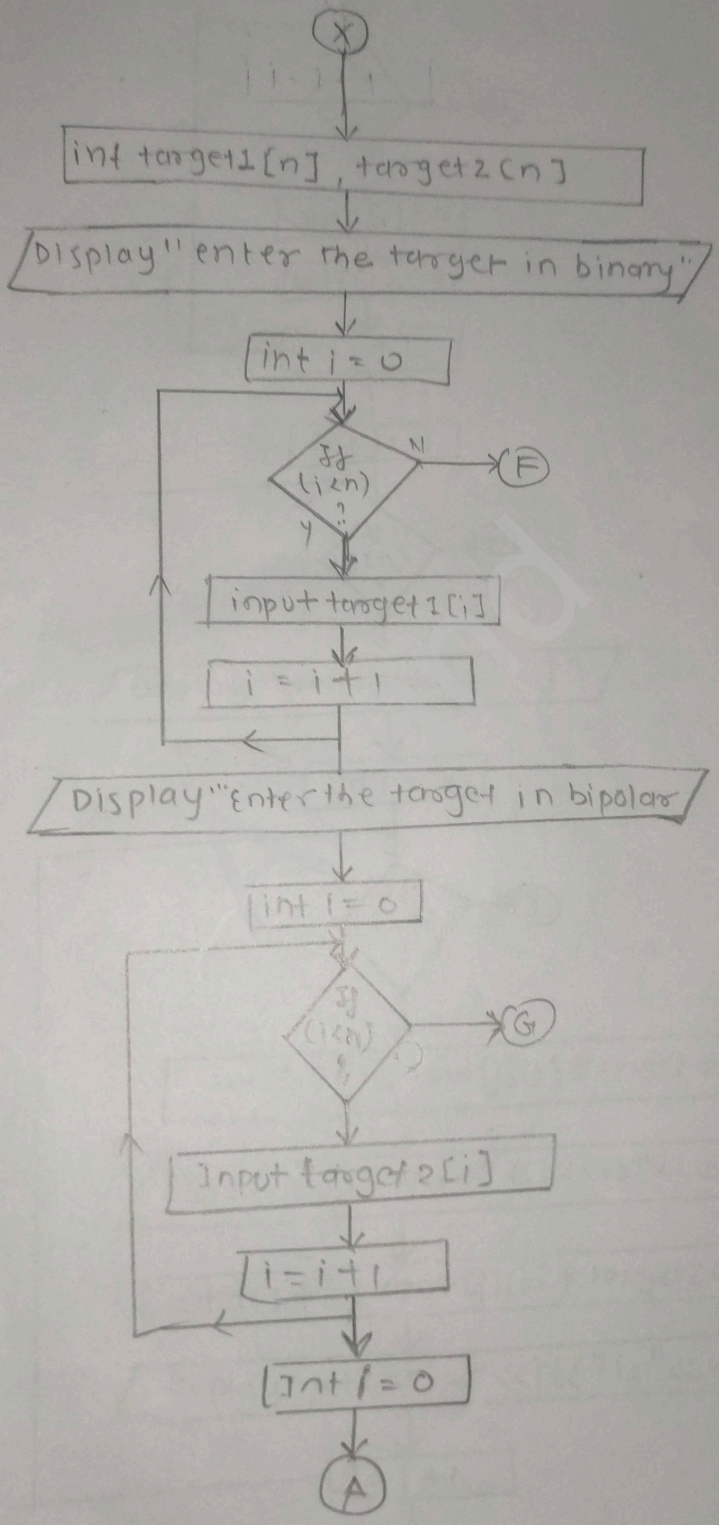
No :- goto step(34)

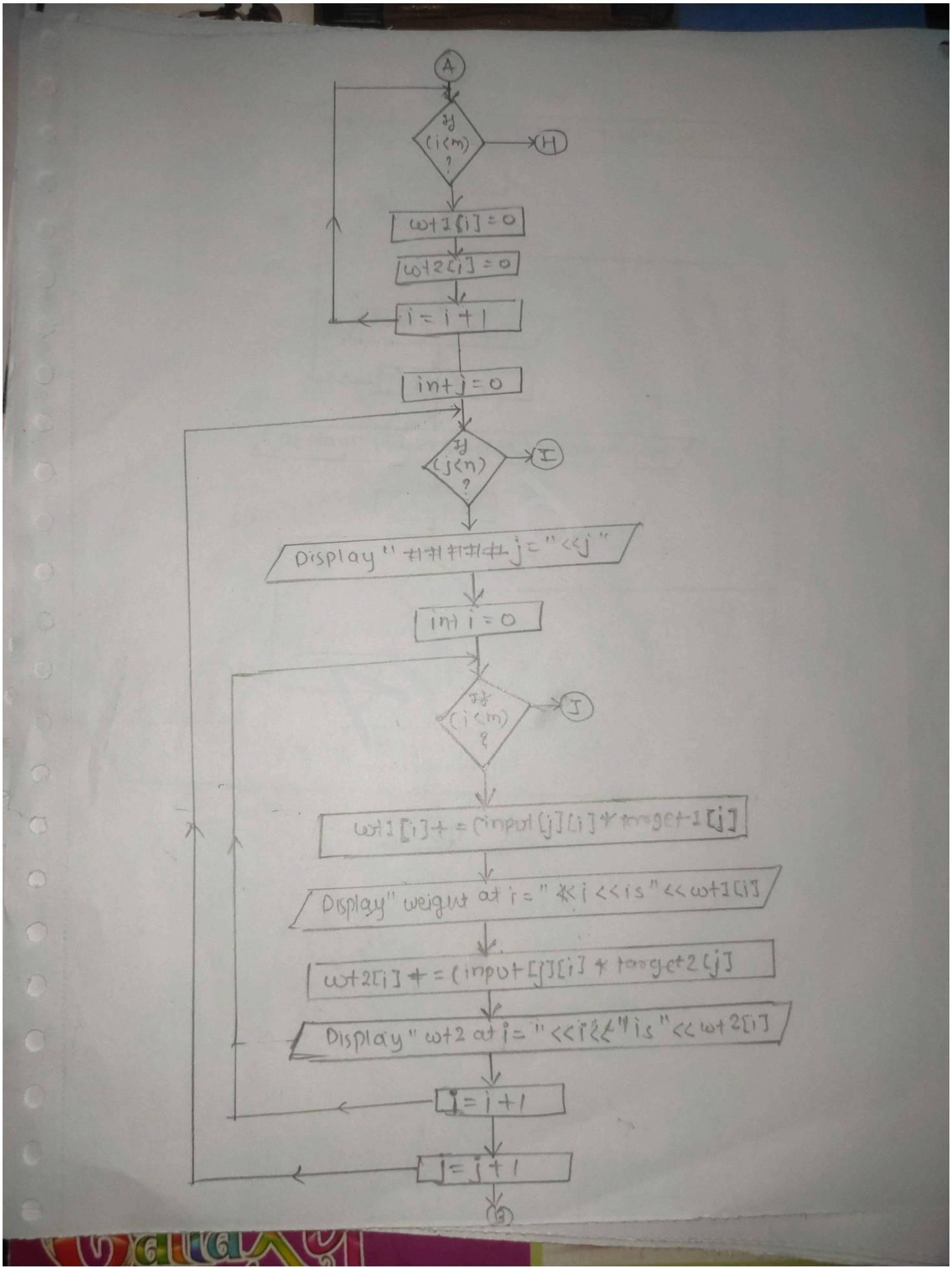
step33 :- i+1 = i

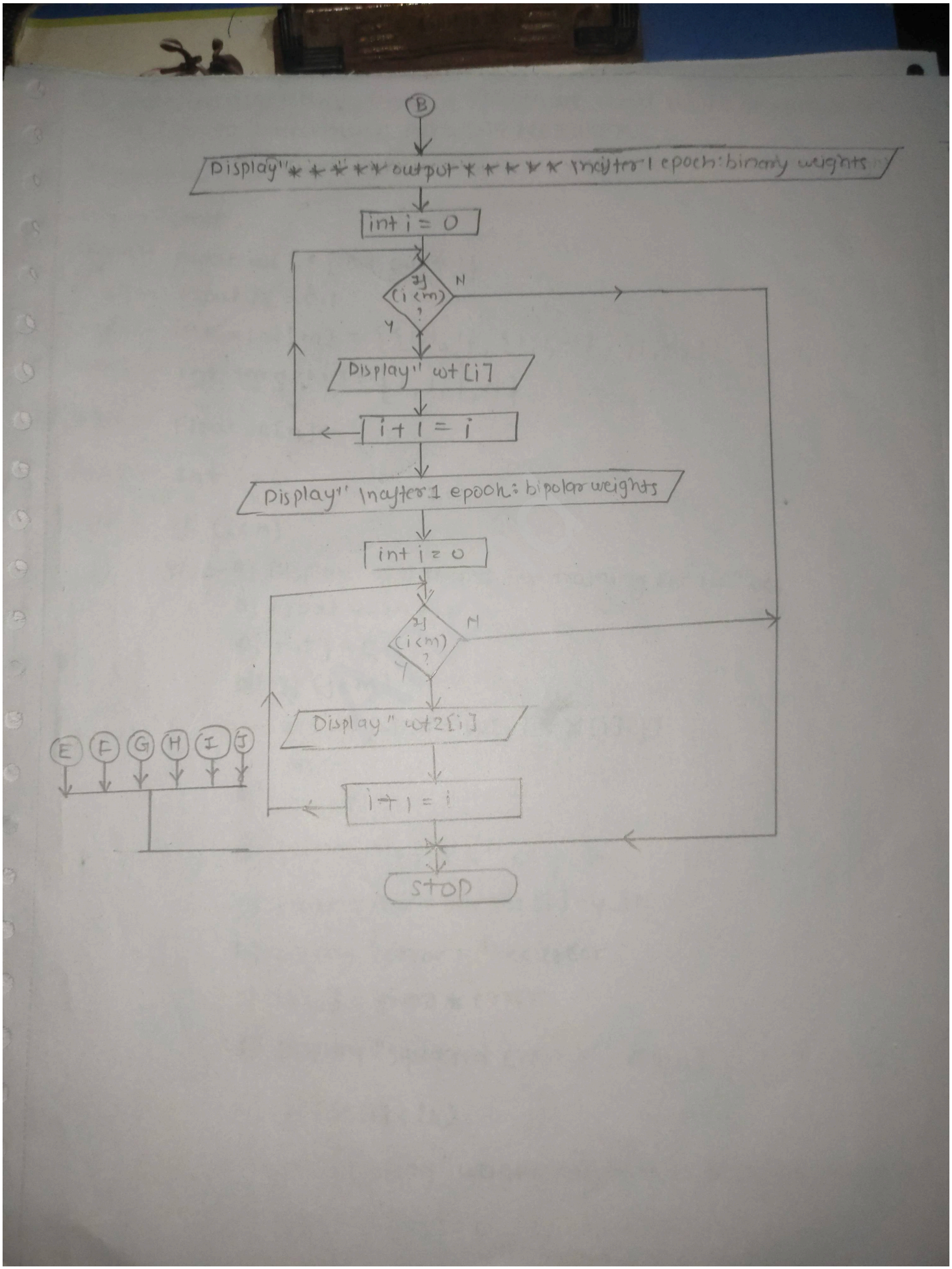
step34 :- stop

Flowchart


start
int m, n
Display "enter no. of features and no. of training datasets
Input m and n
int wt1[m], wt2[m]
int input[n][m]
Display "enter the input matrix row-wise"
int i=0
If (i<n) ?
N
E
Y
int j=0
If (j<m) ?
N
Y
input[i][j]
j=j+1
i=i+1
X


X
int target1[n], target2[n]
Display "enter the target in binary"
int i = 0
If (i<n) ?
N
F
Y
input target1[i]
i = i+1
Display "Enter the target in bipolar
int i = 0
If (i<n) ?
G
Input target2[i]
i = i+1
Int i = 0
A


A
if (i<m) ?
H
wt1[i] = 0
wt2[i] = 0
i = i + 1
int j = 0
if (j<n) ?
I
Display "######j=" <<j"
int i = 0
if (i<m) ?
J
wt1[i] += (input[j][i] * target1[j]
Display" weight at i=" <<i<<is" <<wt1[i]
wt2[i] += (input[j][i] * target2[j]
Display" wt2 at i=" <<i<<" is" <<wt2[i]
i = i + 1
j = j + 1
H


B
Display "*****output*****" "after 1 epoch: binary weights"
int i = 0
If (i < m) ?
N
Y
Display " wt[i]
i + 1 = i
Display " after 1 epoch: bipolar weights
int i = 0
If (i < m) ?
N
Y
Display " wt2[i]
i + 1 = i
E
F
G
H
I
J
STOP

6] Write an algorithm, Draw a flowchart and write a program in cpp to implement ADALINE NETWORK

Algorithm

step1 :- start

step2 :- Float w[] = {0.1, 0.1, 0.1}

step3 :- Float lr = 0.1

step4 :- int x[n][m] = {{1, 1, 1}, {1, 1, -1}, {1, -1, 1}, {1, -1, -1}}

step5 :- int target[] = {1, 1, 1, -1}

step6 :- Float se[n]

step7 :- Int i = 0

step8 :- If (i<n)

yes :- a] Display "#### for training set i = " <<i

b] Float y_in = 0

c] int j = 0

d] If (j<m)

yes :- y_in += w[j] * x[i][j]

NO :-

e] j = j + 1

f] Display " y_in = " << y_in

g] Float error = target[i] - y_in

h] Display " error = " << error

i] se[i] = error * error

j] Display "squared error = " <<se[i]

k] If (se[i] < lr)

yes :- Display "weights adjusted.. training stopped".

NO :- i] int j = 0

ii] If (j<m)

yes :- a] w[j]+= (lr * error * x[i][j]

b] Display " for j = " << j << " weight is " << w[j]

No :-

iii] j = j+1

step9 :- i = i+1

step10 :- Float mse = 0

step11 :- Int k = 0

step12 :- If (k<n)

yes :- mse+ = se[k]

No :-

step13 :- k = k+1

step14 :- mse /= n

step15 :- Display " ***** OUTPUT ***** rmse = "
<< sqrt(mse) << " weights : "

step16 :- Int j = 0

step17 :- If (j<m)

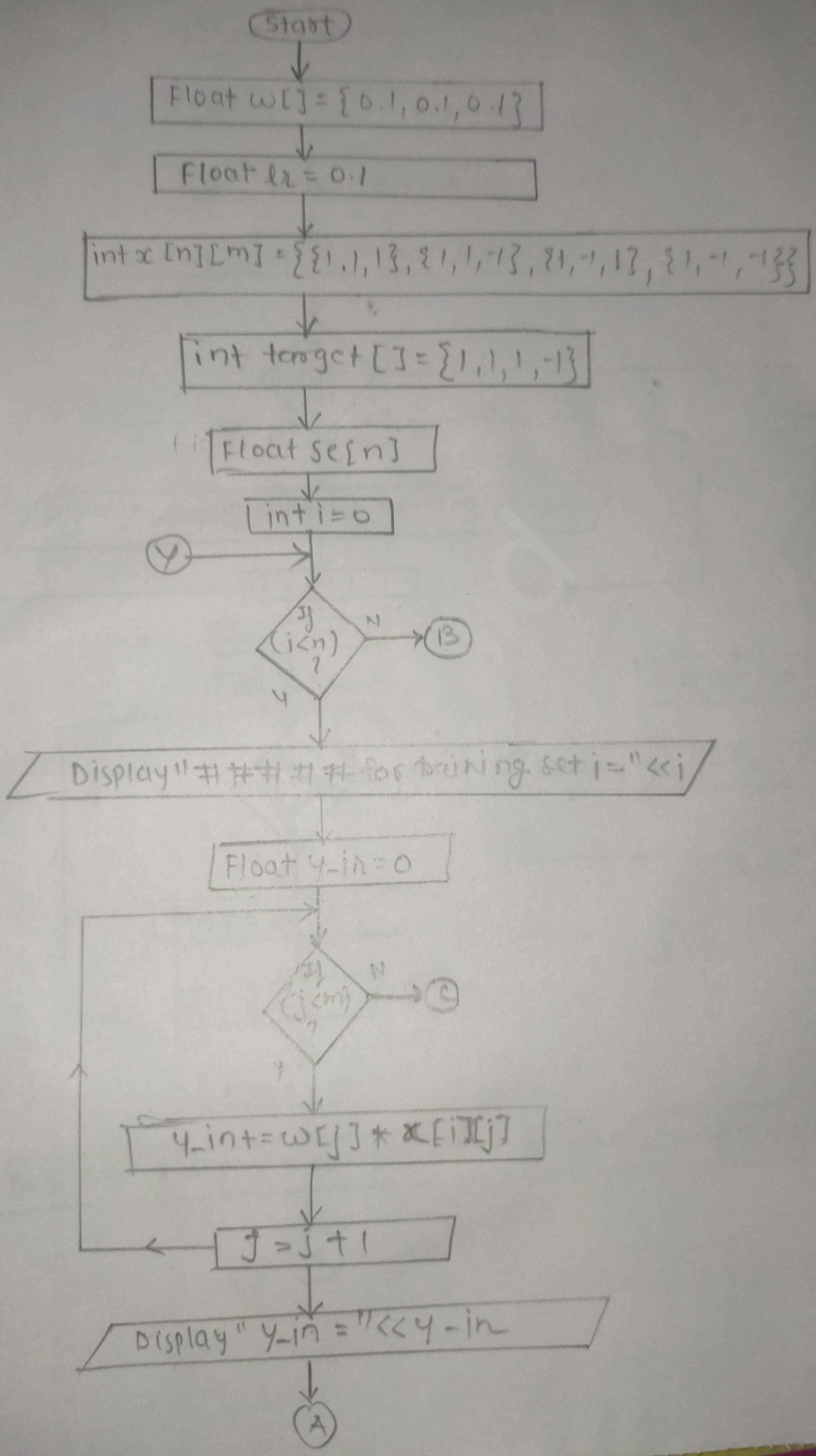
yes :- Display "w[j]"

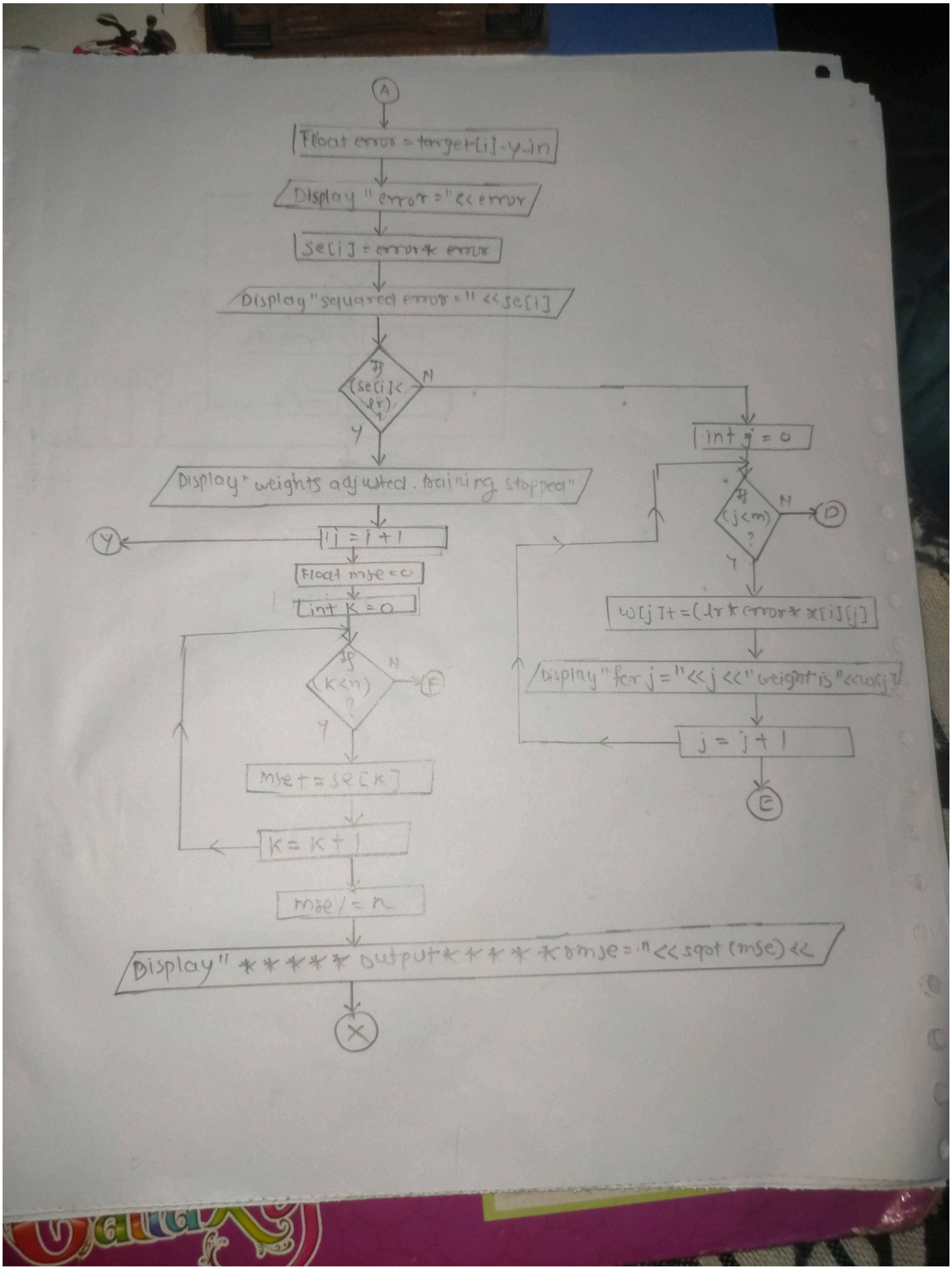
No :-

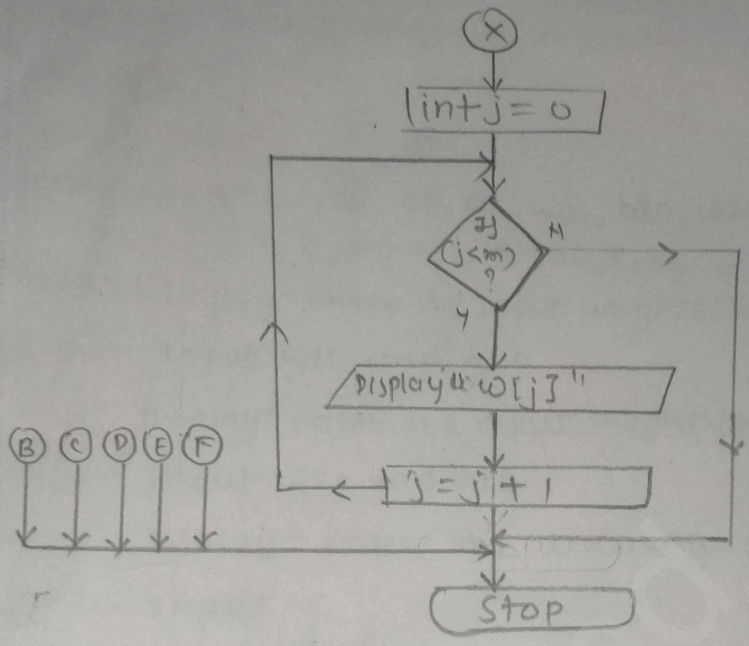
step18 :- j = j+1

step19 :- stop

Flowchart


Start
Float w[] = {0.1, 0.1, 0.1}
Float lr = 0.1
int x [n][m] = {{1,1,1}, {1,1,-1}, {1,-1,1}, {1,-1,-1}}
int target[] = {1,1,1,-1}
Float Se[n]
int i=0
Y
If (i<n) ?
N
13
Y
Display "#### for training set i=" <<i
Float y_in=0
If (j<m) ?
N
C
Y
y_in+=w[j] * x[i][j]
j=j+1
Display "y_in =" <<y_in
A


A
Float error = target[i] - y_in
Display "error =" << error
se[i] = error * error
Display "squared error =" << se[i]
If (se[i] < tr)
N
Y
Display "weights adjusted. training stopped"
i = i + 1
Y
Float mse = 0
int K = 0
If (K < n) ?
N
F
Y
mse += se[K]
K = K + 1
mse /= n
Display "***** output ***** mse =" << sqrt(mse) <<
X
int j = 0
If (j < m) ?
N
D
Y
w[j] += (lr * error * x[i][j]
display "for j =" << j << "weight is" << w[j]
j = j + 1
E


X
int j = 0
(j < m) ?
N
Y
Display " w[j] "
j = j + 1
B
C
D
E
F
Stop

7] Write an algorithm, Draw a flowchart and write a program in cpp for Error Back propagation Algorithm (Ebpa) learning

Algorithm

Step1 :- start

Step2 :- Float i, c, s1, n1, n2, w10, b10, w20, b20, w11, b11, w21, b21, p, t, a0 = 1, a1, a2, e, s2

Step3 :- Display " enter the input weights / base of second n/w = "

Step4 :- Input w10 and b10

Step5 :- Display " enter the input weights / base of second n/w = "

Step6 :- Input w20 and b20

Step7 :- Display " enter the learning coefficient of n/w c = "

Step8 :- Input c

Step9 :- n1 = w10 * p + b10

Step10 :- a1 = tanh(n1)

Step11 :- n2 = w20 * a1 + b20

Step12 :- a2 = tanh(n2)

Step13 :- e = (t - a2)

Step14 :- s2 = -2 * (1 - a2 * a2) * e

Step15 :- s1 = (1 - a1 * a1) * w20 * s2

Step16 :- w21 = w20 - (c * s2 * a1)

Step17 :- w11 = w10 - (c * s1 * a0)

Step18 :- b21 = b20 - (c * s2)

Step19 :- b11 = b10 - (c * s1)

Step20 :- Display " the uploaded weight of first n/w w11 = " << w11

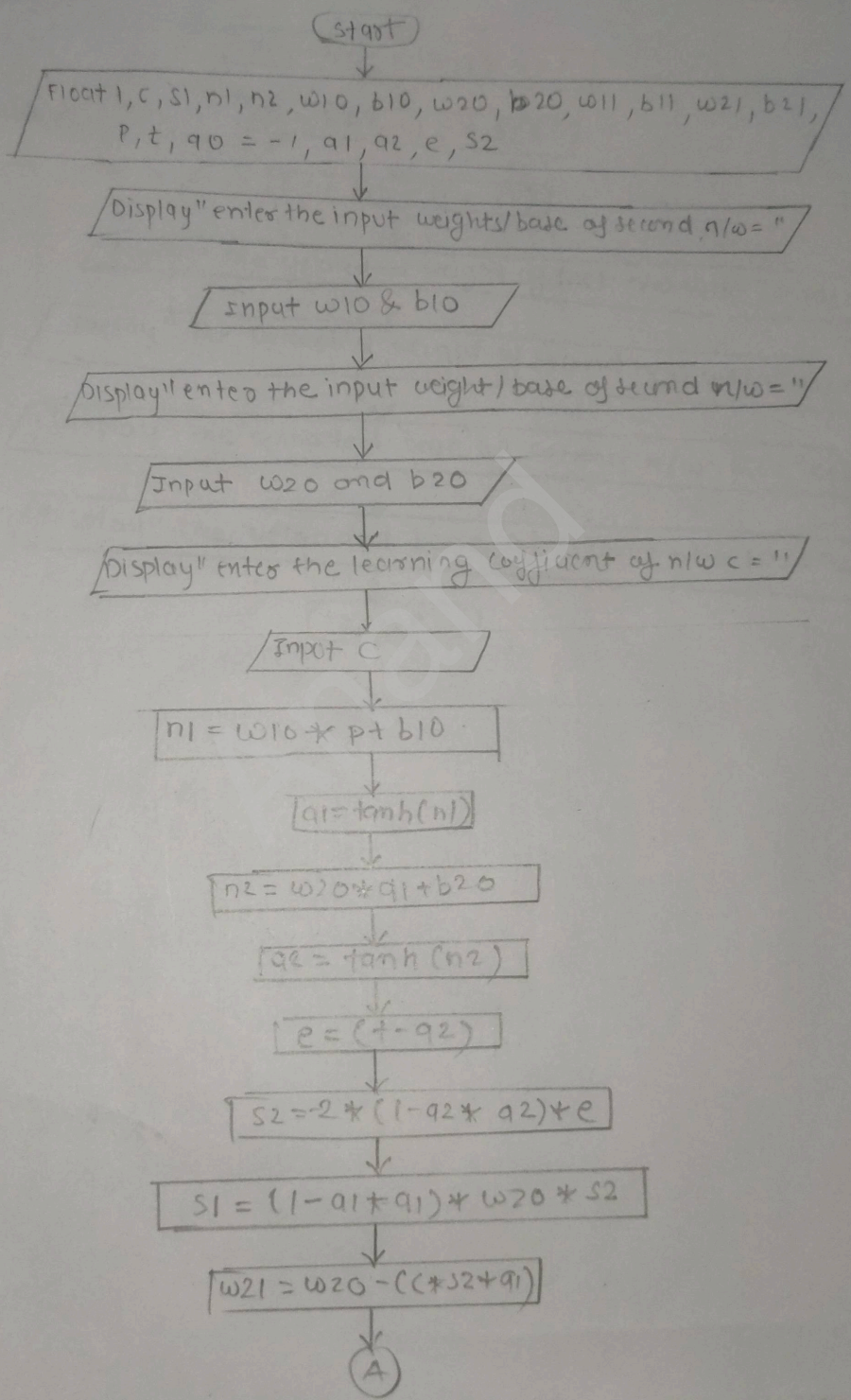
step21 :- Display " the uploaded weight of second n/w w21=" <<w21

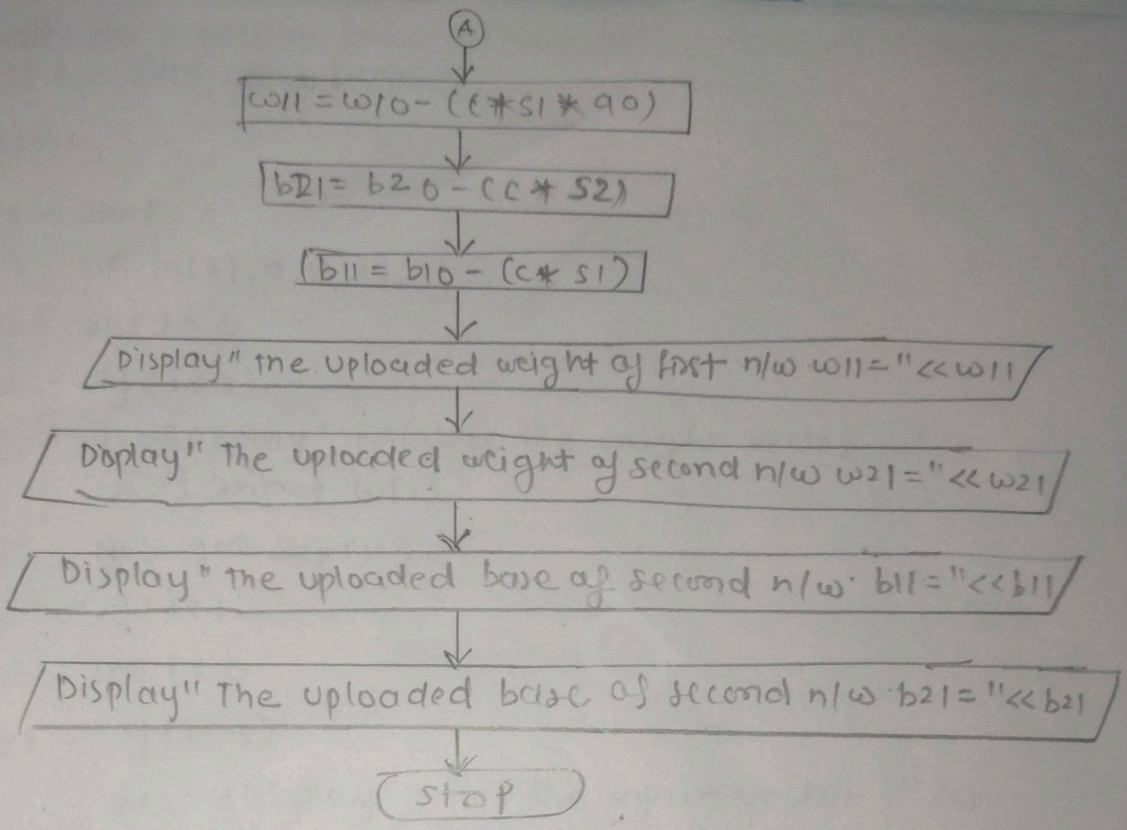
step22 :- Display" the uploaded base of second n/w b11=" <<b11

step23 :- Display" the uploaded base of second n/w b21=" <<b21

step24 :- STOP

Flowchart


start
Float I, C, S1, n1, n2, w10, b10, w20, b20, w11, b11, w21, b21, P, t, q0 = -1, q1, q2, e, S2
Display "enter the input weights/base of second n/w="
Input w10 & b10
Display "enter the input weight/base of second n/w="
Input w20 and b20
Display "enter the learning coefficient of n/w c="
Input C
n1 = w10 * p + b10
q1 = tanh(n1)
n2 = w20 * q1 + b20
q2 = tanh(n2)
e = (t - q2)
S2 = -2 * (1 - q2 * q2) * e
S1 = (1 - q1 * q1) * w20 * S2
w21 = w20 - (C * S2 * q1)
A


A
w11 = w10 - (c * s1 * a0)
b21 = b20 - (c * s2)
b11 = b10 - (c * s1)
Display " the uploaded weight of first n/w w11 = " << w11
Display " The uploaded weight of second n/w w21 = " << w21
Display " the uploaded base of second n/w b11 = " << b11
Display " The uploaded base of second n/w b21 = " << b21
stop

2] Write an algorithm, Draw a Flowchart and write a program in cpp for SVM classification.

Algorithm

Step1 :- start

Step2 :- int in[3], d, w[3], q=0

Step3 :- Int i = 0

Step4 :- If (i<3)

Yes :- i] Display "initialize the weight vector w" <<i

ii] Input w[i]

No :- goto step (12)

Step5 :- i = i+1

Step6 :- int i = 0

Step7 :- If (i<3)

yes :- i] Display "enter the input vector i" <<i

ii] Input in[i]

No :- goto step(12)

Step8 :- Display "enter the desired output

Step9 :- Input d

Step10 :- int ans = 1

Step11 :- while (ans==1)

Yes :- i] Int i = 0

ii] If (i<3).

Yes :- a] q = q + w[i] * in[i]

No :- goto step(12)

iii] i = i+1

iv] Display "desired output is" <<q

v] Display "actual output is " <<q

vi] int e

vii] e = d - q

viii] Display " error is" << e

viiii] Display " press 1 to adjust weight else 0"

ix] input ans

x] If (e<0)

yes :- a] int i = 0

b] If (i<3)

yes :- w[i] = w[i] - 1

no :- goto step(12)

c] i = i + 1

No :- If (e>0)

yes :- a] int i = 0

b] If (i<3)

yes :- w[i] = w[i] + 1

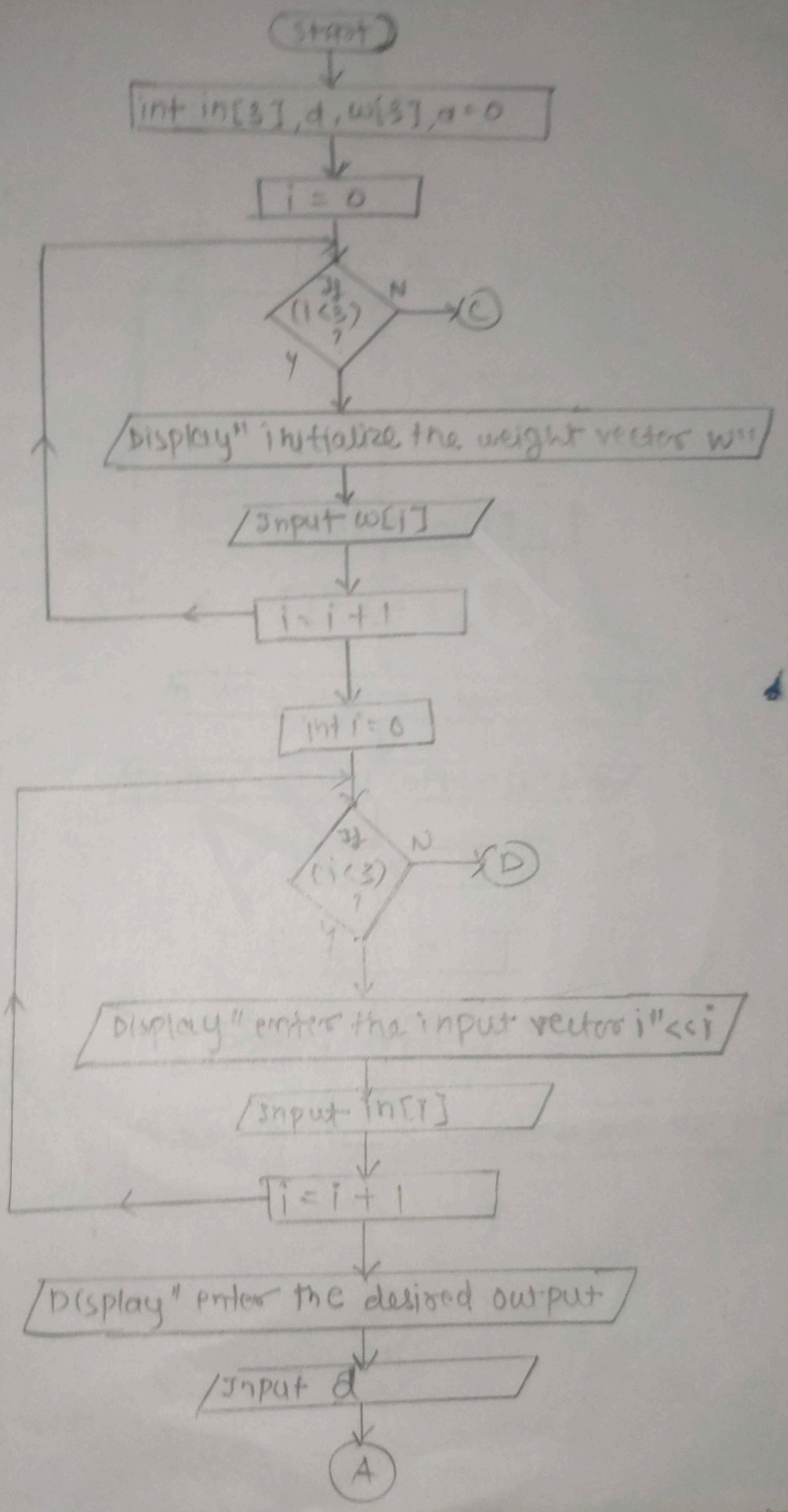
No :- goto step(12)

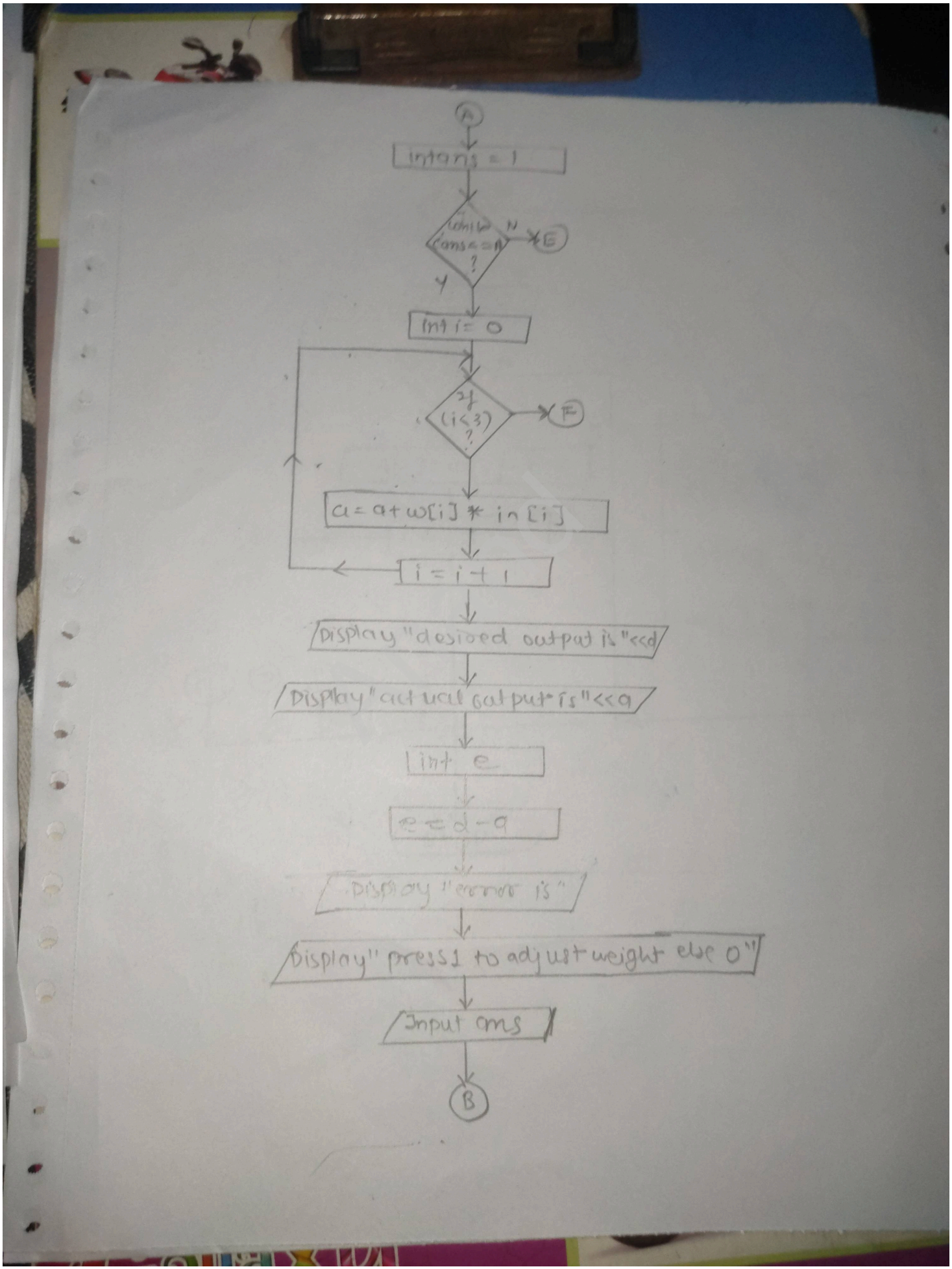
c] i = i + 1

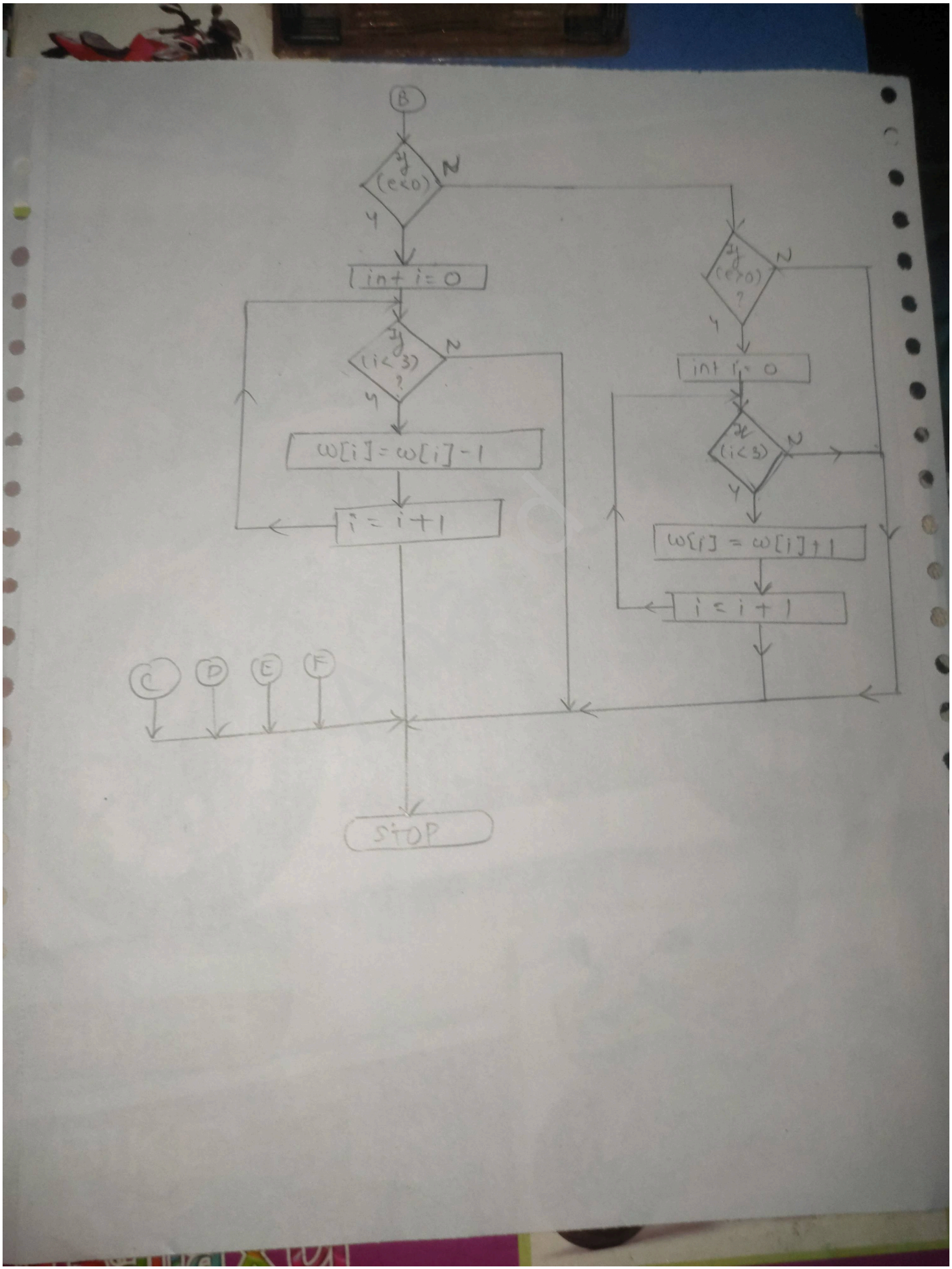
No :- goto step(12)

Step12 :- stop

Flowchart


start
int in[3], d, w[3], a=0
i = 0
if (i<3) ?
N
C
Y
Display" initialize the weight vector w"
Input w[i]
i = i + 1
int i = 0
if (i<3) ?
N
D
Y
Display" enter the input vector i"<<i
Input in[i]
i = i + 1
Display" enter the desired output
Input d
A


A
int ans = 1
while ans <= n ?
N
E
Y
int i = 0
if (i<3) ?
F
a = a + w[i] * in[i]
i = i + 1
Display "desired output is" << d
Display "actual output is" << a
int e
e = d - a
Display "error is"
Display "press 1 to adjust weight else 0"
Input ans
B


B
(e<0)
Y
N
int i=0
(i<3)
Y
N
w[i] = w[i] - 1
i = i+1
(e>0)
Y
N
int i = 0
(i<3)
Y
N
w[i] = w[i] + 1
i = i + 1
C
D
E
F
STOP