Name: Muhammad Asim Anis ID: FA19-BSSE-0032

Assignment #4

Topic: Asymtatic Notations: example # 1

i) Big O (Best)

$$0 \leq Fn \leq c\,(g(n))$$

$Fn = 3n + 2$

$gn = n$

$0 \leq 3n + 2 \leq c\,(n)$ $\quad c = 1$

$0 \leq 3(1) + 2 \leq 1(1)$ $\quad n = 1$

$0 \leq 5 \leq 1$

$0 \leq 3(2) + 2 \leq 2(2)$ $\quad c = 2$

$0 \leq 8 \leq 4$ $\quad n = 2$

$0 \leq 3(3) + 2 \leq 3(3)$ $\quad c = 3$

$0 \leq 11 \leq 9$ $\quad n = 3$

$0 \leq 3(4) + 2 \leq 4(4)$ $\quad \boxed{c = 4}$

$0 \leq 14 \leq 16$ $\quad n = 4$

equation satisfy of the best Case.

ii Big Ω (worest)

$0 \leq Fn \geq c\,(g(n))$

$fn = 3n+2$

$gn = n$

$Fn. \geq c\,(g(n))$

$3n+2 \geq c\,(n)$ $\quad c=1,\ n=1$

$3(1)+2 \geq c(1)\,(1)$

$5 \geq 1$

equation satisfy of the worest case.

iii) Big Θ (Avevage)

$0 \leq c_1(g(n)) \leq fn \leq c_2(g(n))$

$fn = 3n+2$

$gn = n$

$c_1(n) \leq 3n+2 \leq c_2(n)$ $\quad c_1=1,\ c_2=1,\ n=1$

$1(1) \leq 3(1)+2 \leq 1(1)$

$1 \leq 5 \leq 1$

$4(4) \leq$

$3(3) \leq 3(3)+2 \leq 3(3)$ $\quad C_1 = 3,\ C_2 = 3,\ n = 3$

$9 \leq 11 \leq 9$

$3(4) \leq 3(4)+2 \leq 4(4)$ $\quad C_1=3,\ C_2=4,\ n=4$

$12 \leq 14 \leq 20$

$12 \leq 14 \leq 20$

equation satisfy of the Average Case.

example # 2:

i) Big O (Best)

$$0 \leq Fn \leq C(g(n))$$

$Fn = n^2 + 1$

$gn = n$

$0 \leq n^2+1 \leq C(n)$ $\quad C=1$

$0 \leq (1)^2+1 \leq 1(1)$ $\quad n=1$

$0 \leq 2 \leq 1$

$0 \leq (2)^2+1 \leq 2(2)$ $\quad C=2$

$0 \leq 5 \leq 4$ $\quad n=2$

$0 \leq (3)^2+1 \leq 3(3)$ $\quad \boxed{C=3}$

$0 \leq 7 \leq 9$ $\quad n=3$

equation satisfy of the Best case.

ii) Big Ω (worst)

$0 \leq fn \geq c\,(gn))$

$fn = n^2 + 1$

$gn = n$

$fn \geq c\,(g(n))$ $\quad c = 1$, $n = 1$

$n^2 + 1 \geq c\,(n)$

$(1)^2 + 1 \geq 1\,(1)$

$2 \geq 1$ ✓

equation satisfy of the worst case.

iii) Big Θ (Average)

$$0 \leq C_1(g(n)) \leq Fn \leq C_2(g(n))$$

$Fn = n^2+1$

$g(n) = n$

$C_1(n) \leq n^2+1 \leq C_2(n)$ $\quad C_1 = 1$

$1(1) \leq (1)^2+1 \leq 1(1)$ $\quad C_2 = 1$

$1 \leq 2 \leq 1$ $\quad n = 1$

$1(2) \leq (2)^2+1 \leq 2(2)$ $\quad C_1 = 1$

$2 \leq 5 \leq 4$ $\quad C_2 = 2$

$n = 2$

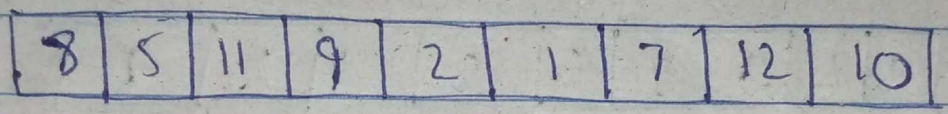
$1(3) \leq (3)^2+1 \leq 3(3)$ $\quad C_1 = 1$

$3 \leq 7 \leq 9$ $\quad C_2 = 3$

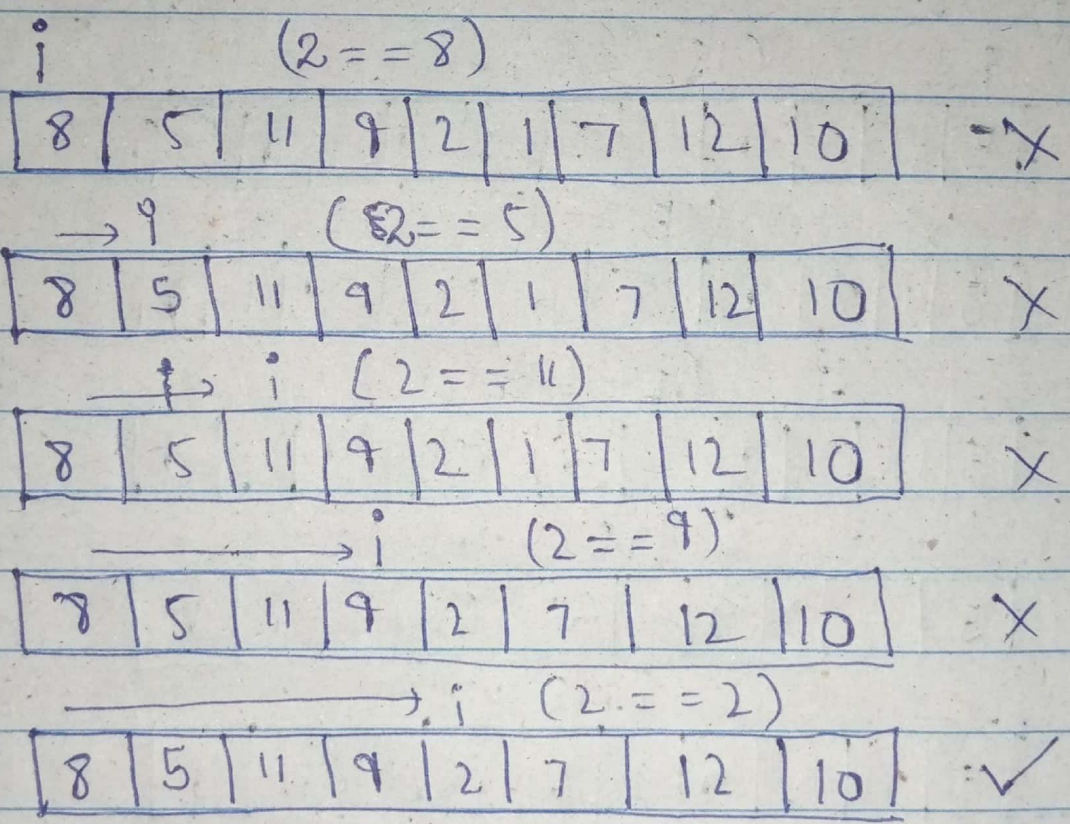
$n = 3$

equation satisfy of the Average case.

Topic: Linear Search example # 1



Key = 2



Number found = 2

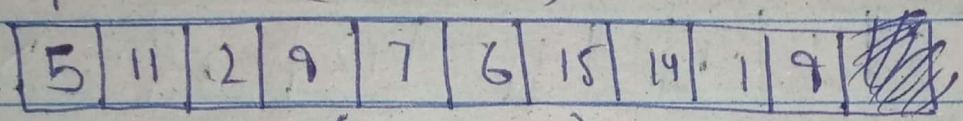
Complete the iteration and end. or else break the flow.

Example # 2

5	11	2	9	7	6	15	14	1	9

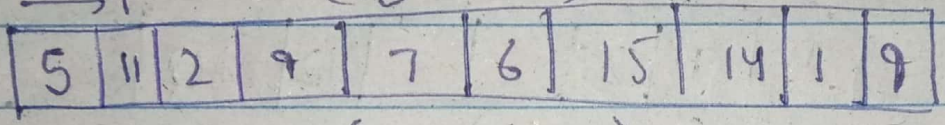
Key = 15

i (15 == 5)



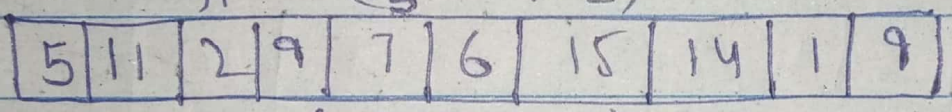
X

→i (15 == 11)



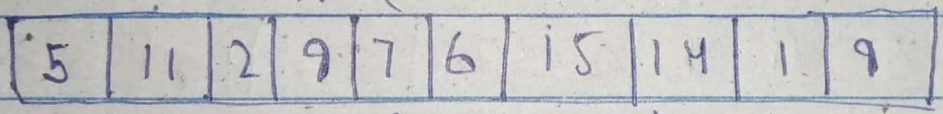
X

→i (15 == 2)



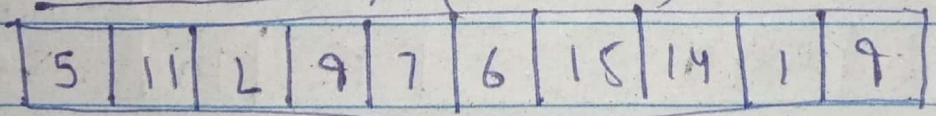
X

→i (15 == 9)



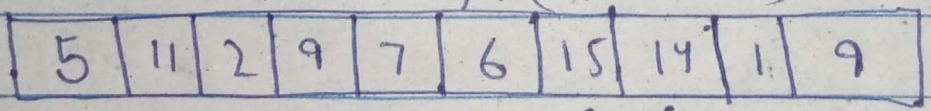
X

→i (15 == 7)



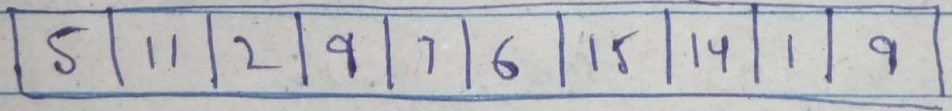
X

→i (15 == 6)



X

→i (15 == 15)



✓

Number found = 15

Now break the flow of loop :

Topic: Binary Search example: #1

5	11	12	14	16	20	24	30	50

Key = 14

low mid high

5	11	12	14	16	20	24	30	50

✓(14 < 16) (14 > 16) X (14 == 16) X

low mid high

5	11	12	14

✓(14 > 11) (14 < 11) X (14 == 11)

low/mid high

12	14

✓(14 > 12) X(14 < 12) X (14 == 12)

low/mid | 14 | high

X(14 > 14) ✓(14 == mid = 14) X(14 < 14)

Number found = 14

while condition false. Loop end.

example # 2

1	5	7	9	10	12	16	18	21	24

Key = 16

low mid high

1	5	7	9	10	12	16	18	21	24

✓ (16 > 10) X (16 < 10) (16 == 10)

low mid high

12	16	18	21	24

↳ (16 < 18)

low/mid high

12	16

↳ (16 > 12)

low/mid | 16 | high

↳ (16 == 16)

Number found = 16

while condition false loop end.

Topic: Bubble Sort

example # 1

5	2	1	4	7	6	8	9	11	10	14

Iteration 1:

5	2	1	4	7	6	8	9	11	10	14

i J

if (arr[i] > arr[j])

(5 > 2) Swap

2	5	1	4	7	6	8	9	11	10	14

i J (2 > 1) swap

1	5	2	4	7	6	8	9	11	10	14

i J (1 > 4) false

1	5	2	4	7	6	8	9	11	10	14

i J (1 > 7)

1	5	2	4	7	6	8	9	11	10	14

i J (1 > 6)

1	5	2	4	7	6	8	9	11	10	14

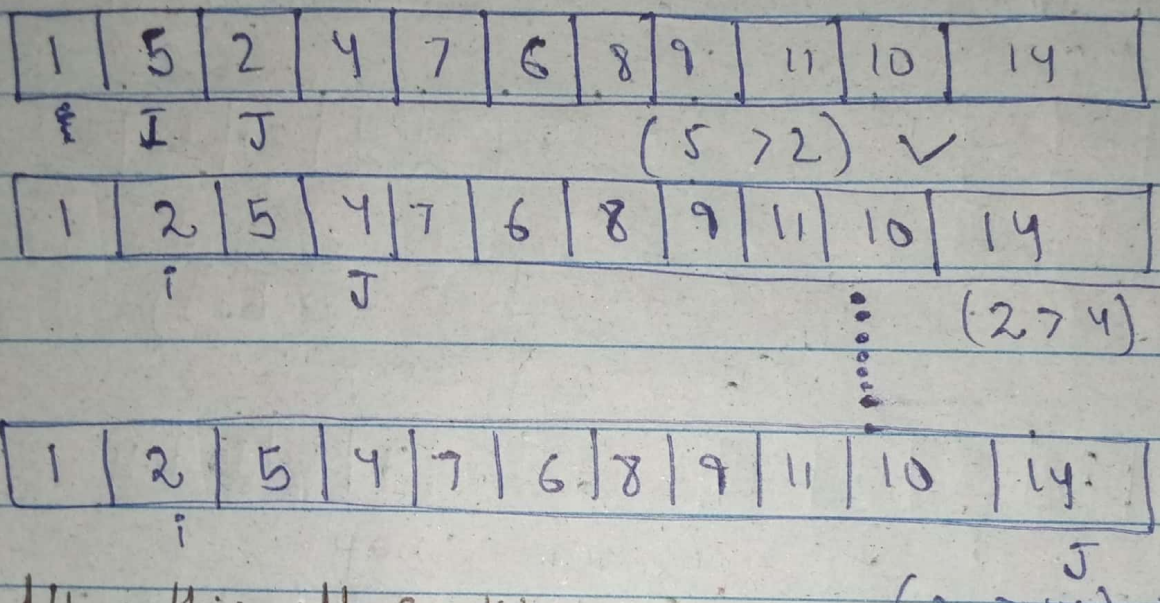
i J (1 > 8) (1 > 9) (1 > 11) (1 > 10) false

1	5	2	4	7	6	8	9	11	10	14

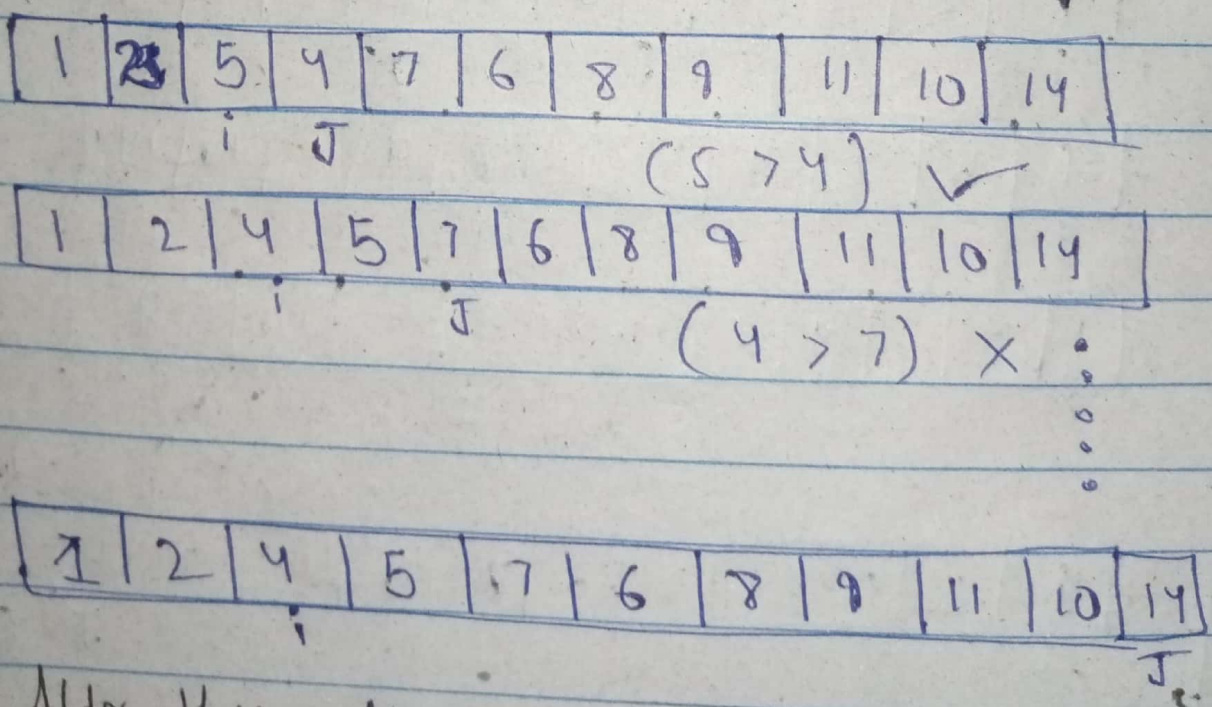
i J (1 > 14) false.

After this all conditions are false:

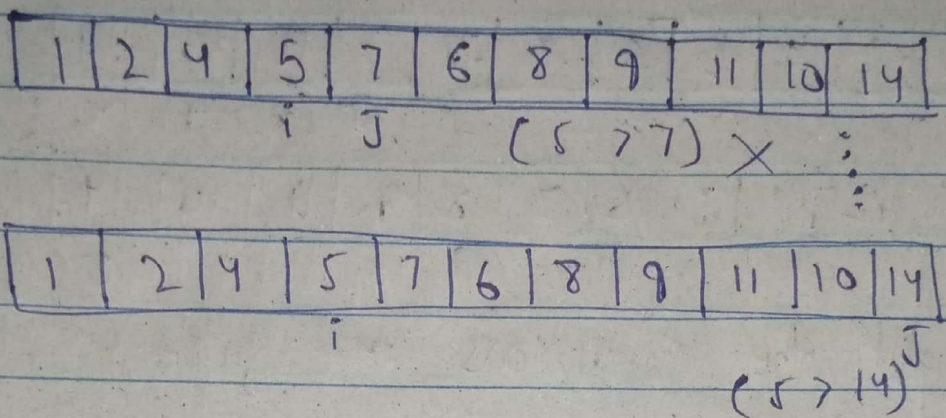
Iteration : 2 :



Iteration : 3

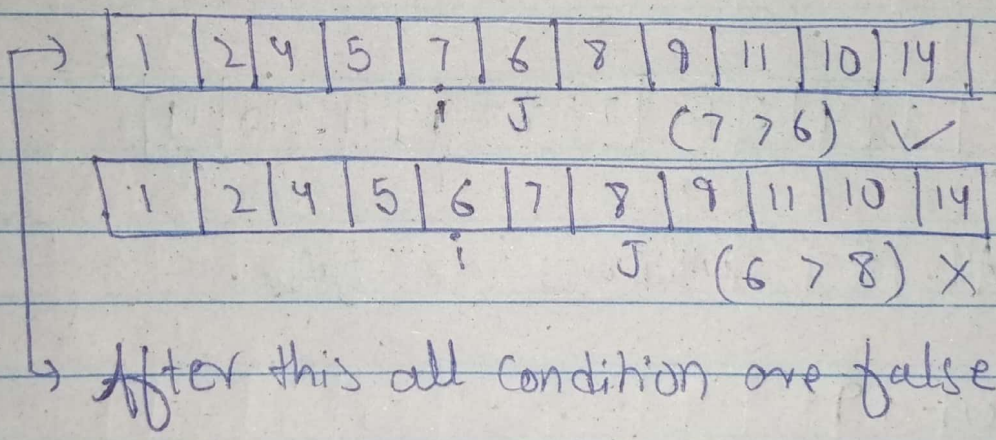


Iteration : 4



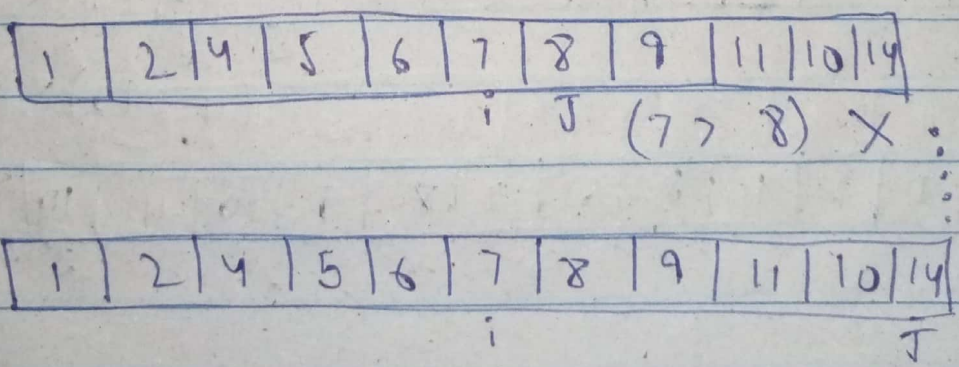
All the condition are false

Iteration : 5



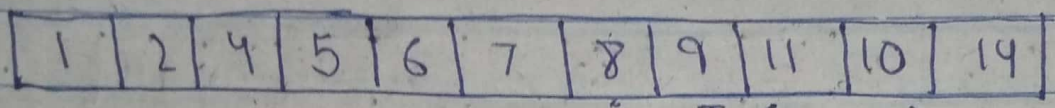
After this all condition are false.

Iteration : 6

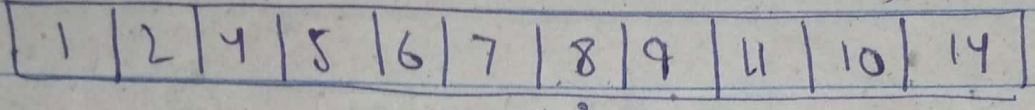


All conditions are false.

Iteration : 7 :

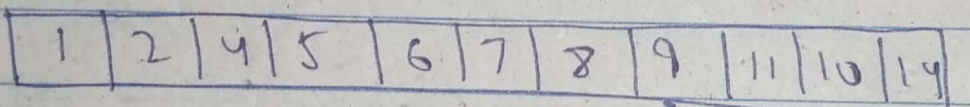


(8 > 9) ✗

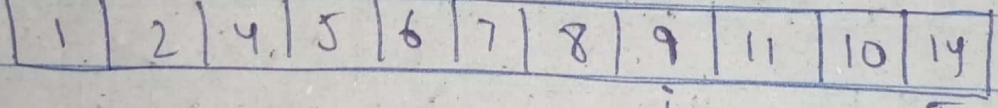


All the conditions are false.

Iteration : 8

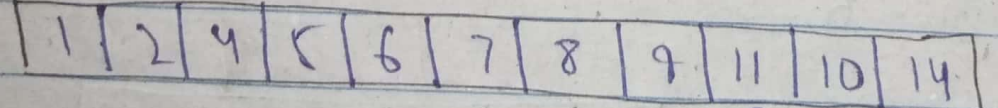


(9 > 11) ✗

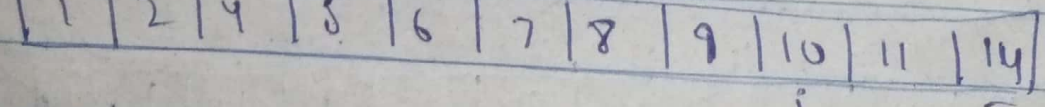


All the conditions are false.

Iteration : 9

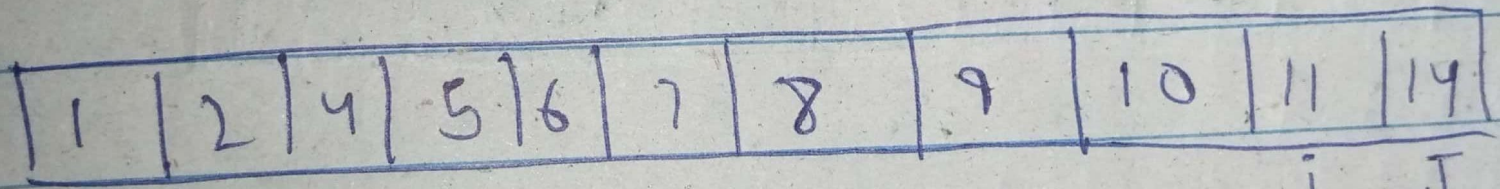


(11 > 10) ✓



(10 > 14) ✗

Iteration : 10



(11 > 14) X

* Sorted Array:

1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 14

example # 2:

6	2	3	9	7	12	10	8	1	4

Iteration : 1.

6	2	3	9	7	12	10	8	1	4

i J (6 > 2) ✓

2	6	3	9	7	12	10	8	1	4

i J (2 > 3) ✗

2	6	3	9	7	12	10	8	1	4

i J (2 > 9) ✗

2	6	3	9	7	12	10	8	1	4

i J (2 > 7) ✗

2	6	3	9	7	12	10	8	1	4

i J (2 > 12) ✗

2	6	3	9	7	12	10	8	1	4

i J (2 > 10) ✗

2	6	3	9	7	12	10	8	1	4

i J (2 > 8)

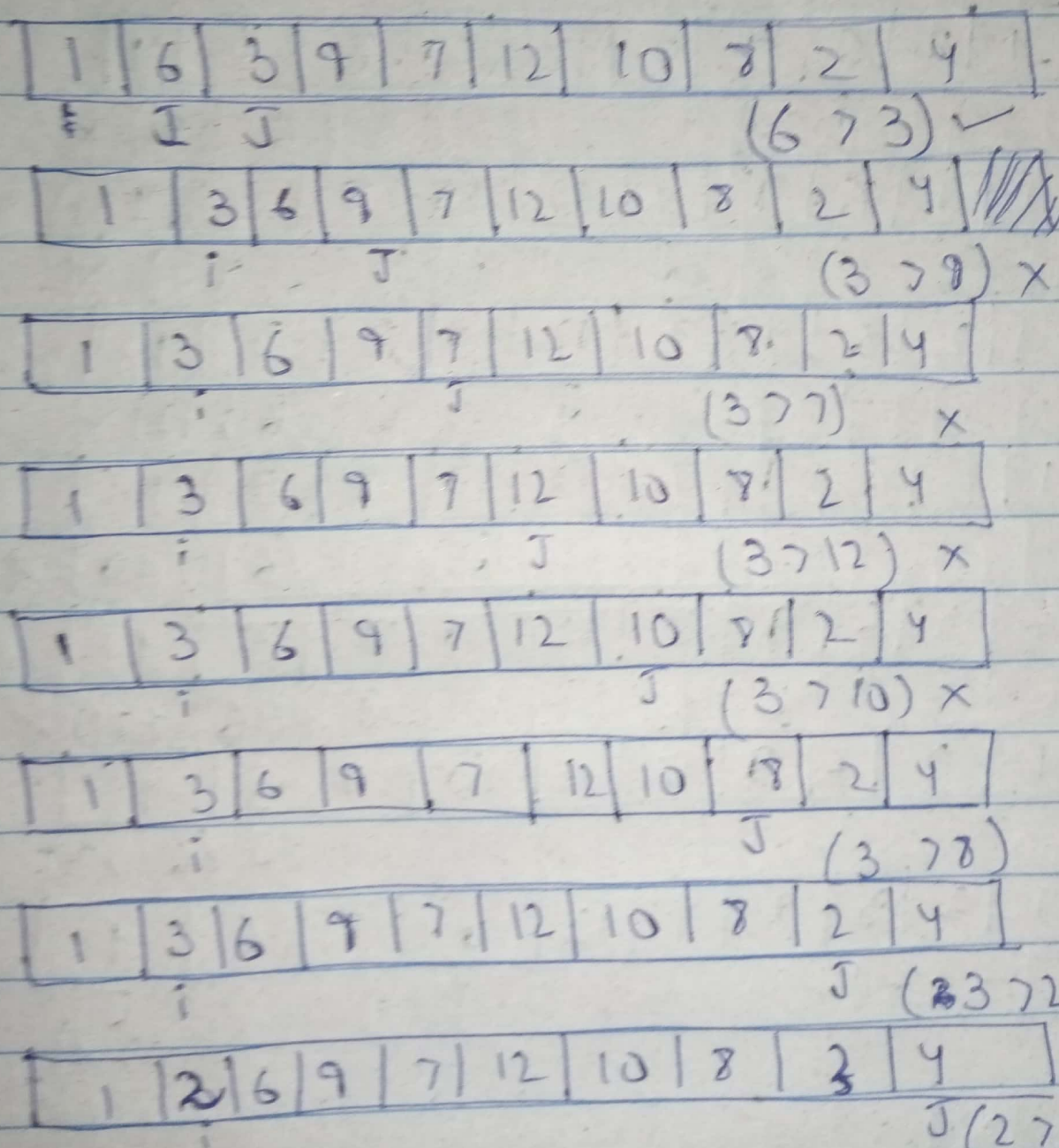
2	6	3	9	7	12	10	8	1	4

i J (2 > 1) ✓

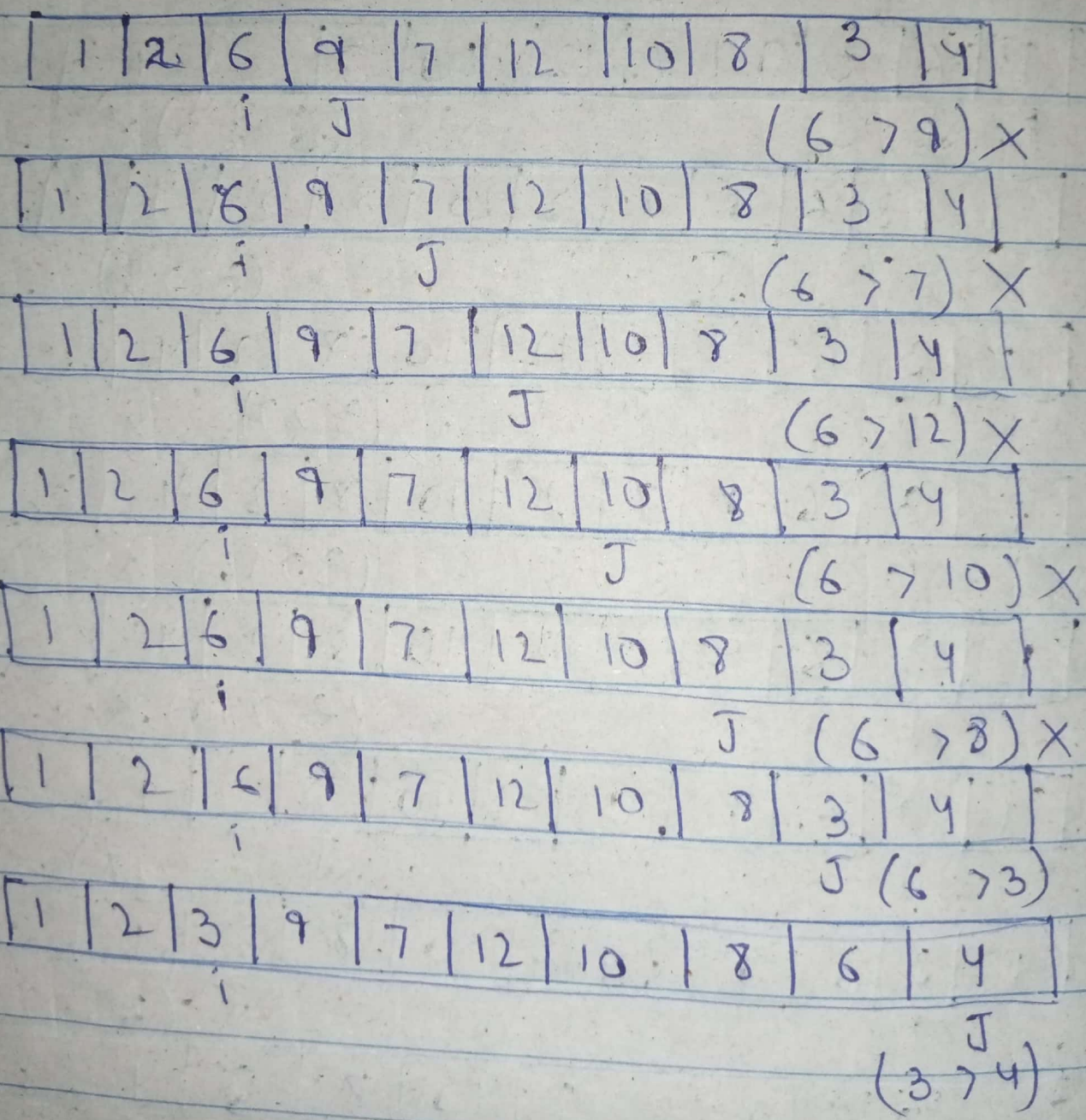
1	6	3	9	7	12	10	8	2	4

i J (1 > 4) ✗

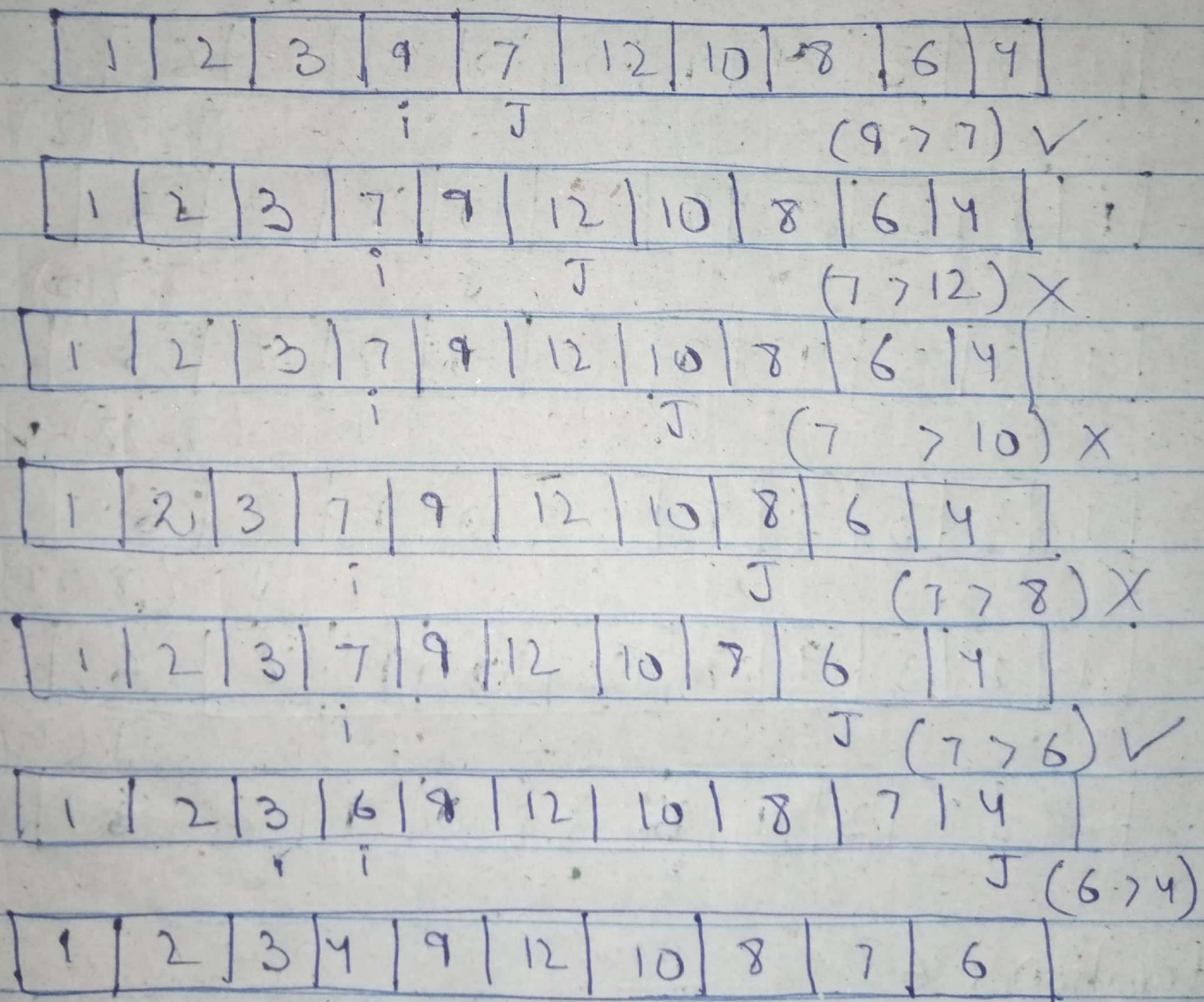
Iteration # 2:



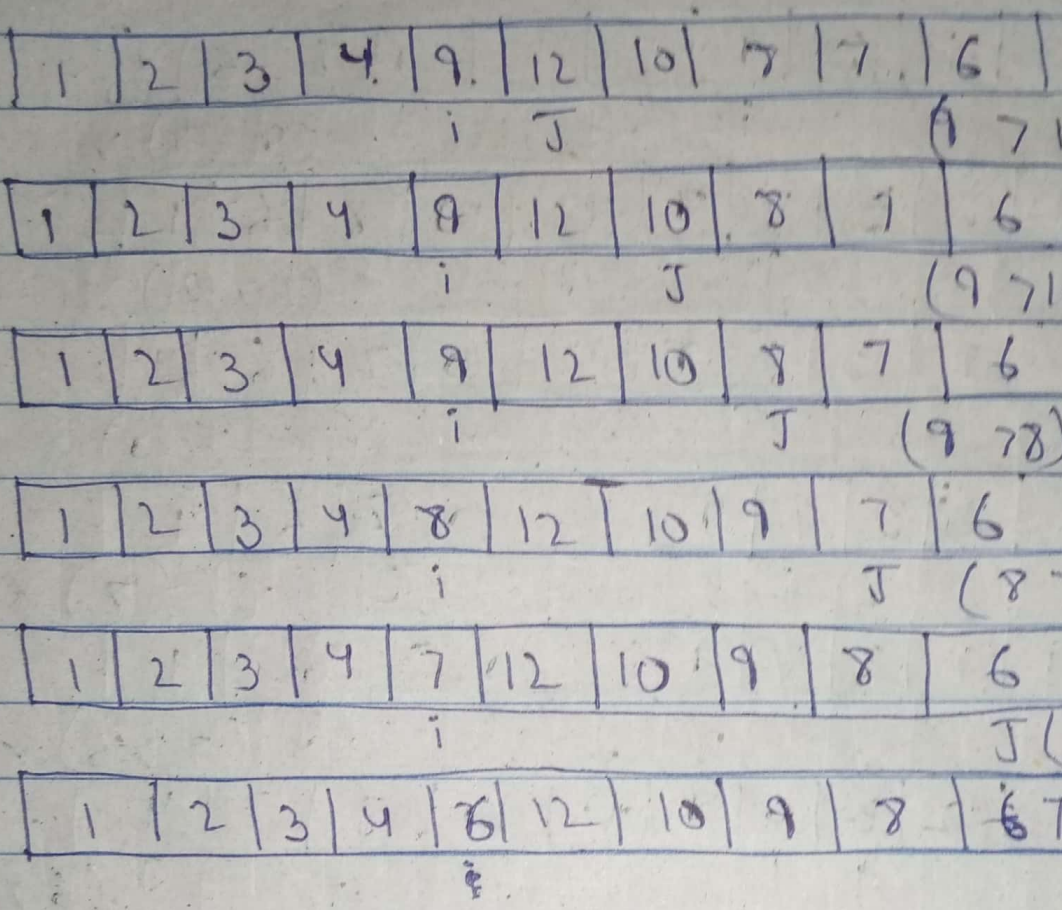
Iteration : 3



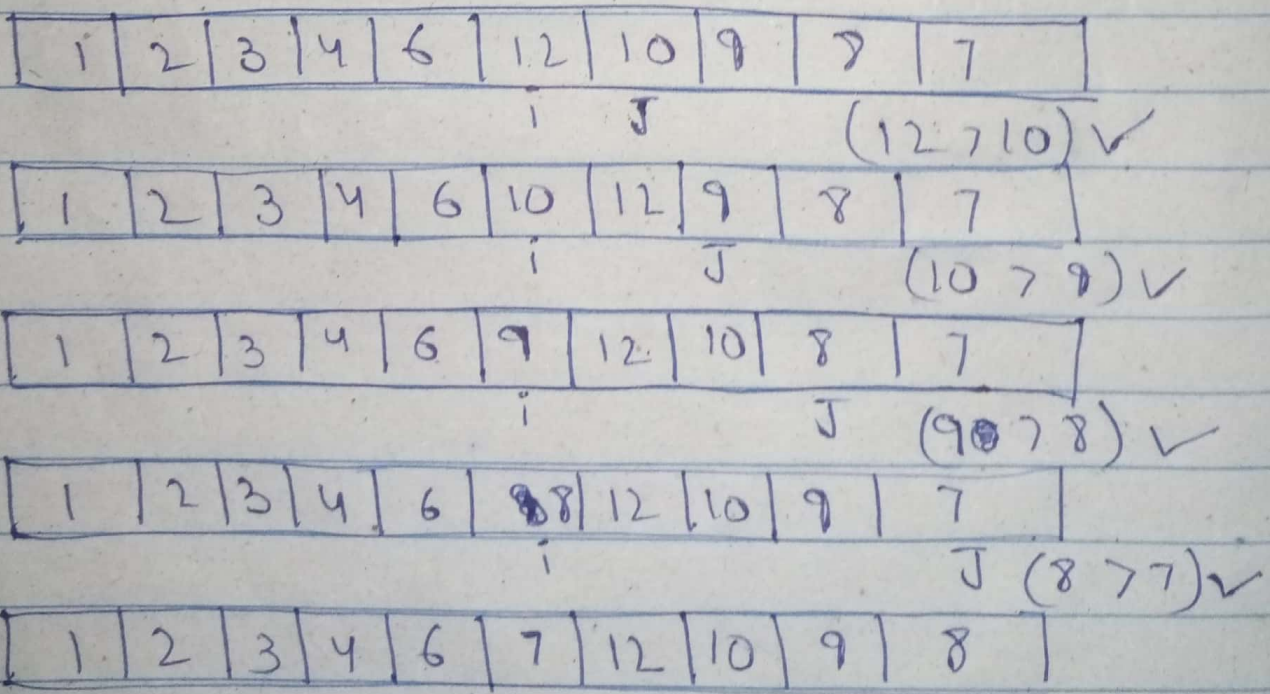
Iteration : 4



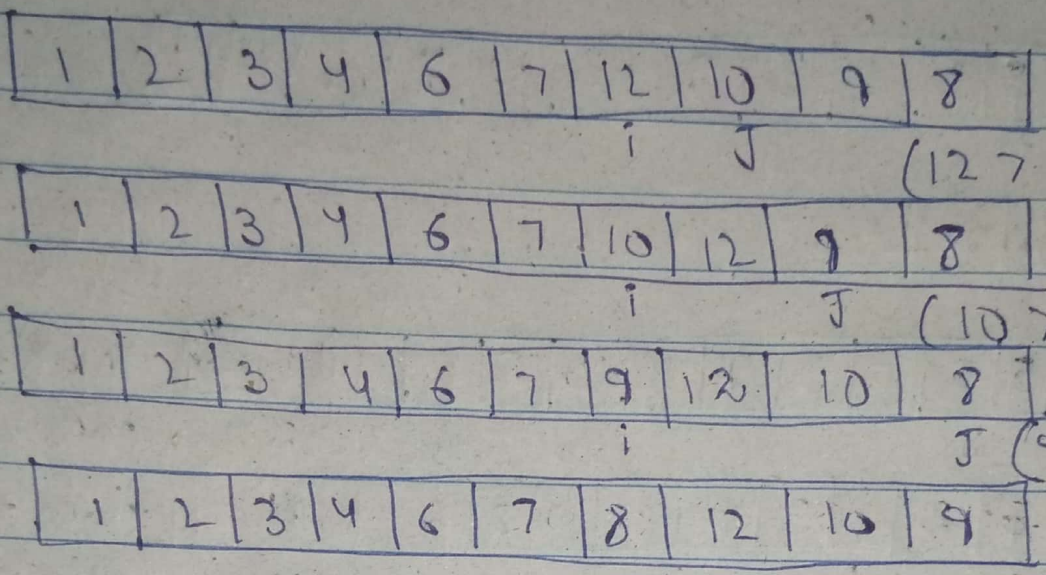
Iteration : 5



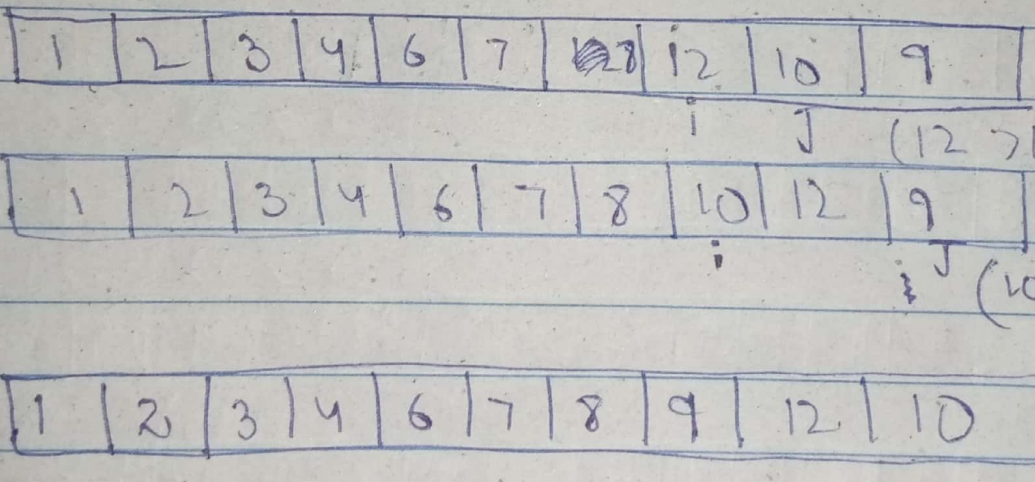
Iteration : 6



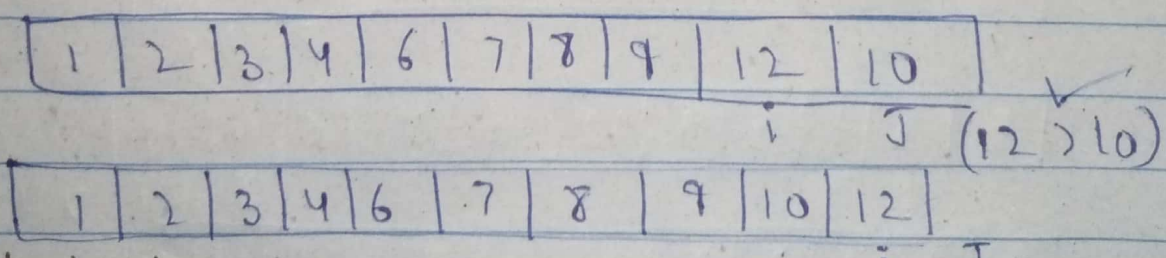
Iteration : 7



Iteration : 8



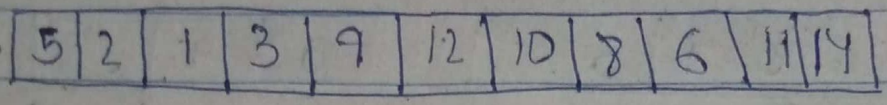
Iteration : 9



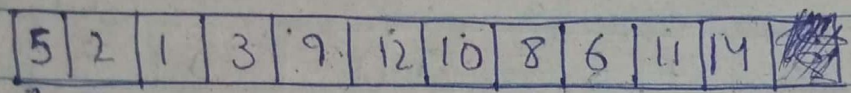
Sorted Array:

1, 2, 3, 4, 6, 7, 8, 9, 10, 12

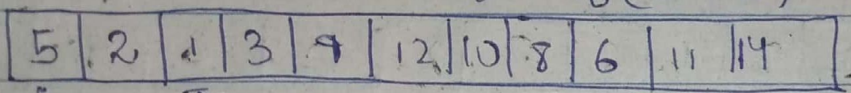
Topic: Selection Sort example # 1



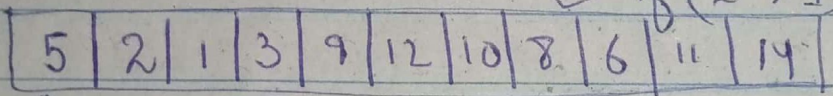
Iteration :- 1



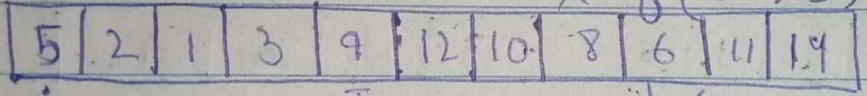
if (5 > 2) (index > J)



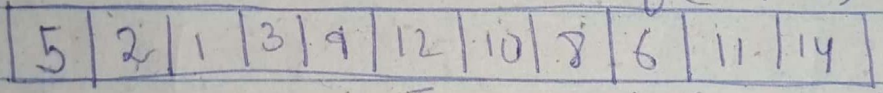
if (2 > 1) update



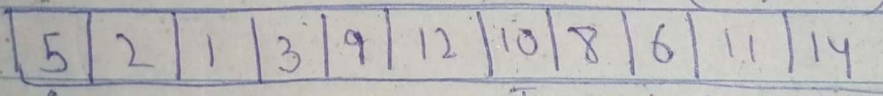
× if (1 > 3)



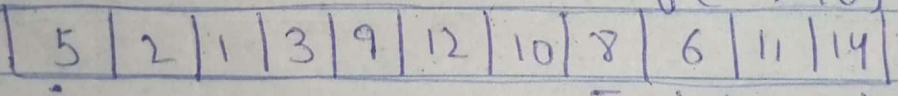
× if (1 > 9)



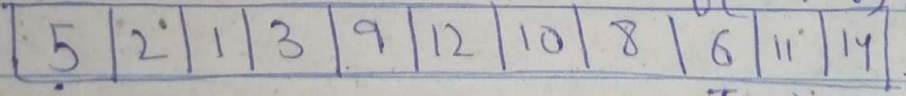
× if (1 > 12)



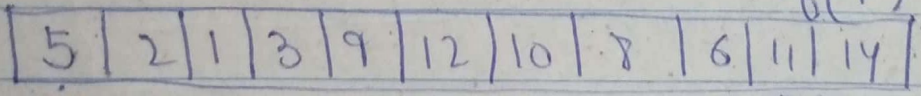
× if (1 > 10)



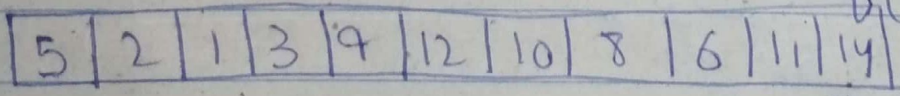
if (1 > 8) ×



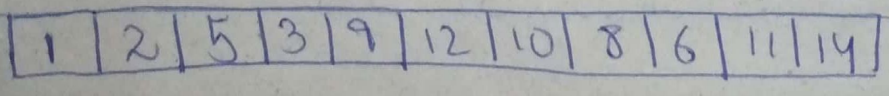
if (1 > 6) ×



if (1 > 11) ×

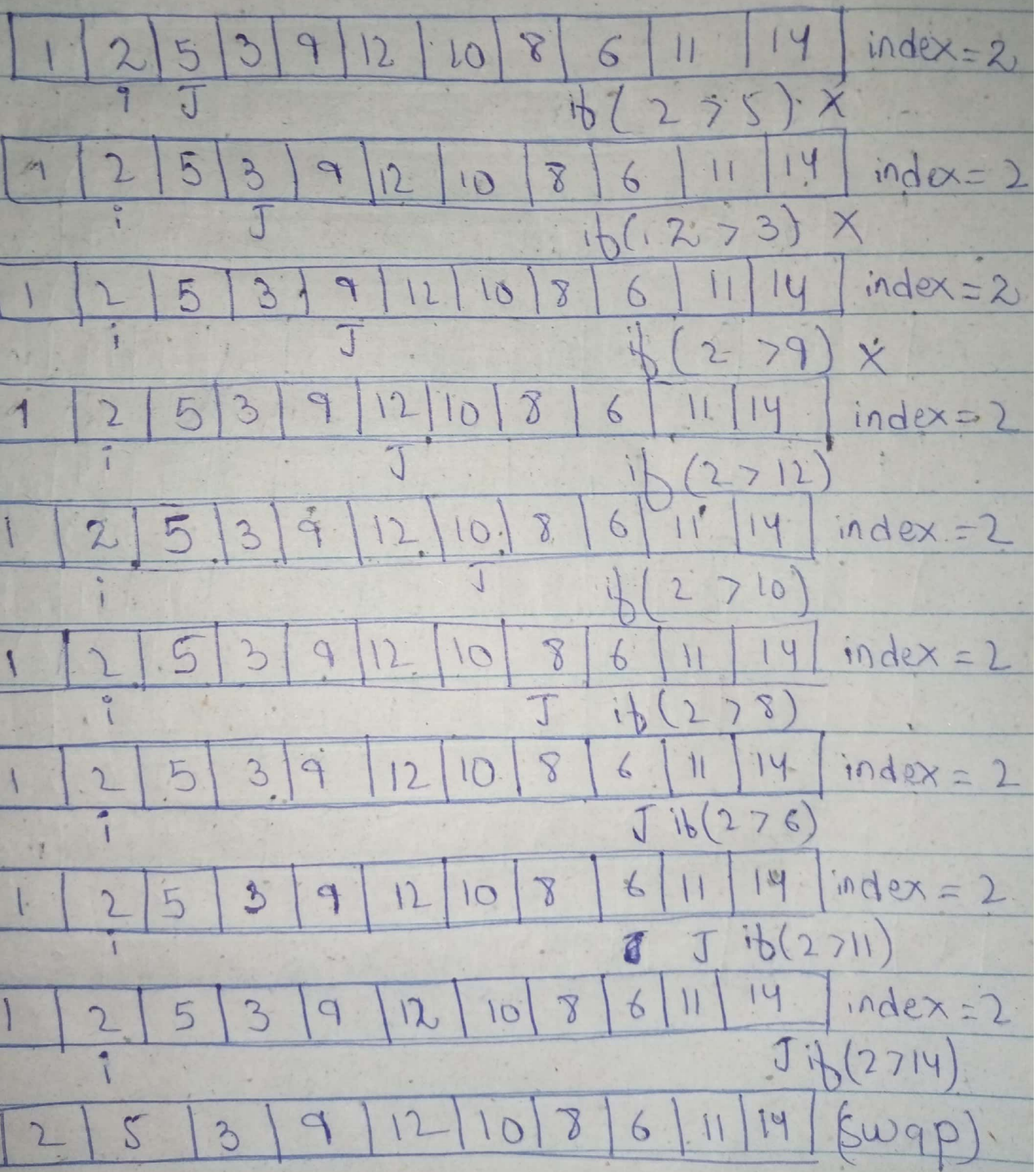


if (1 > 14) ×

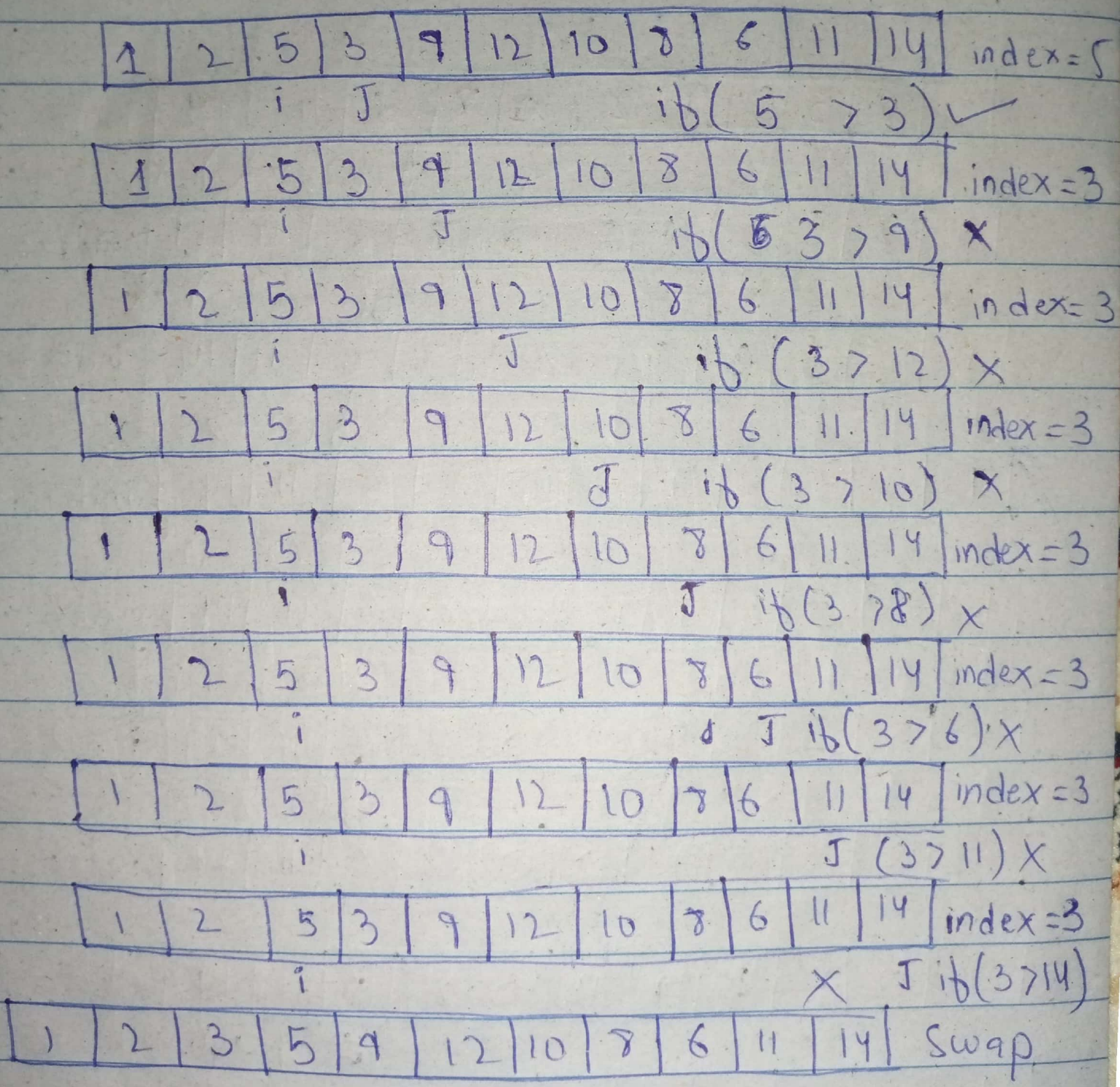


(Swap)

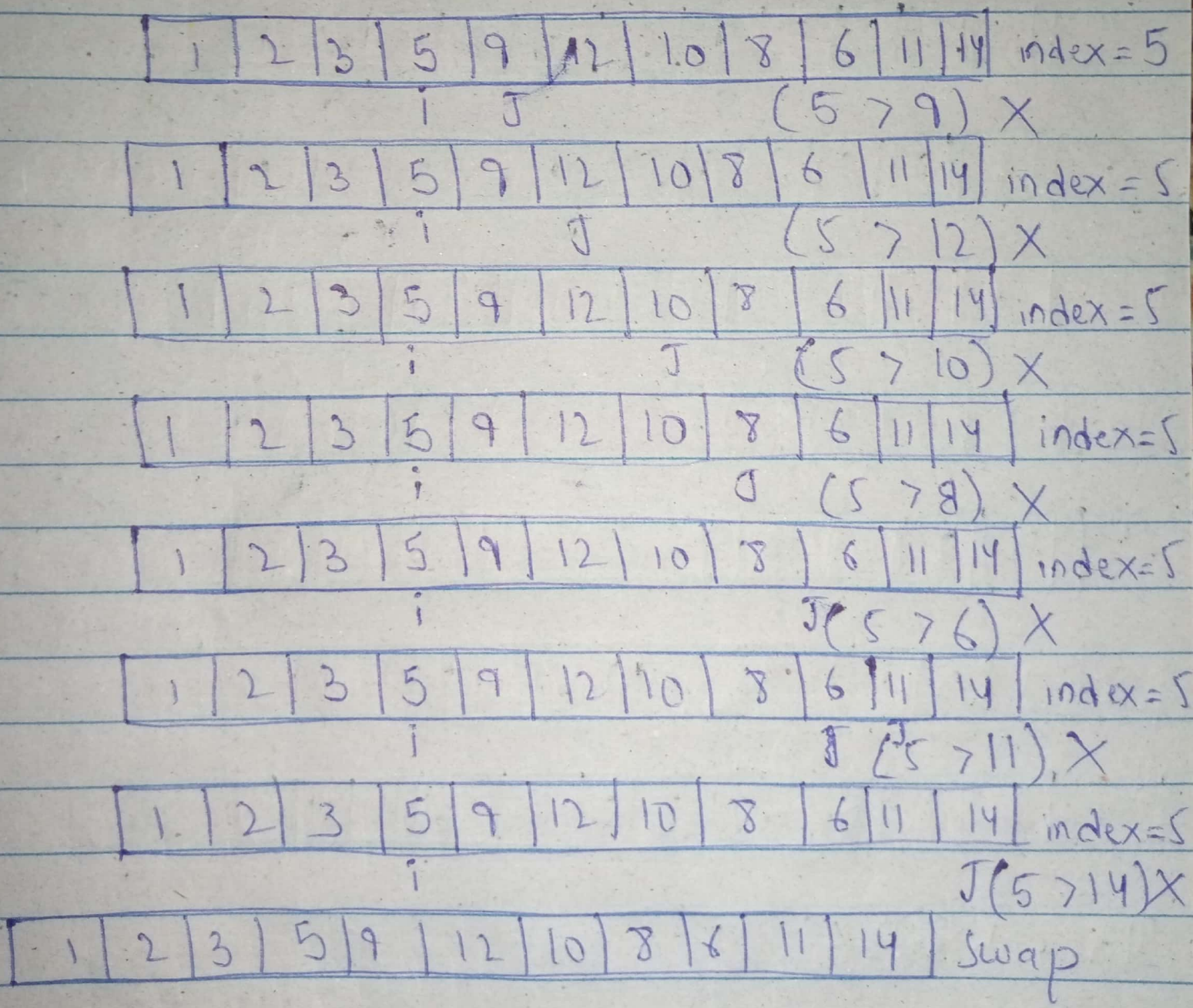
Iteration : 2



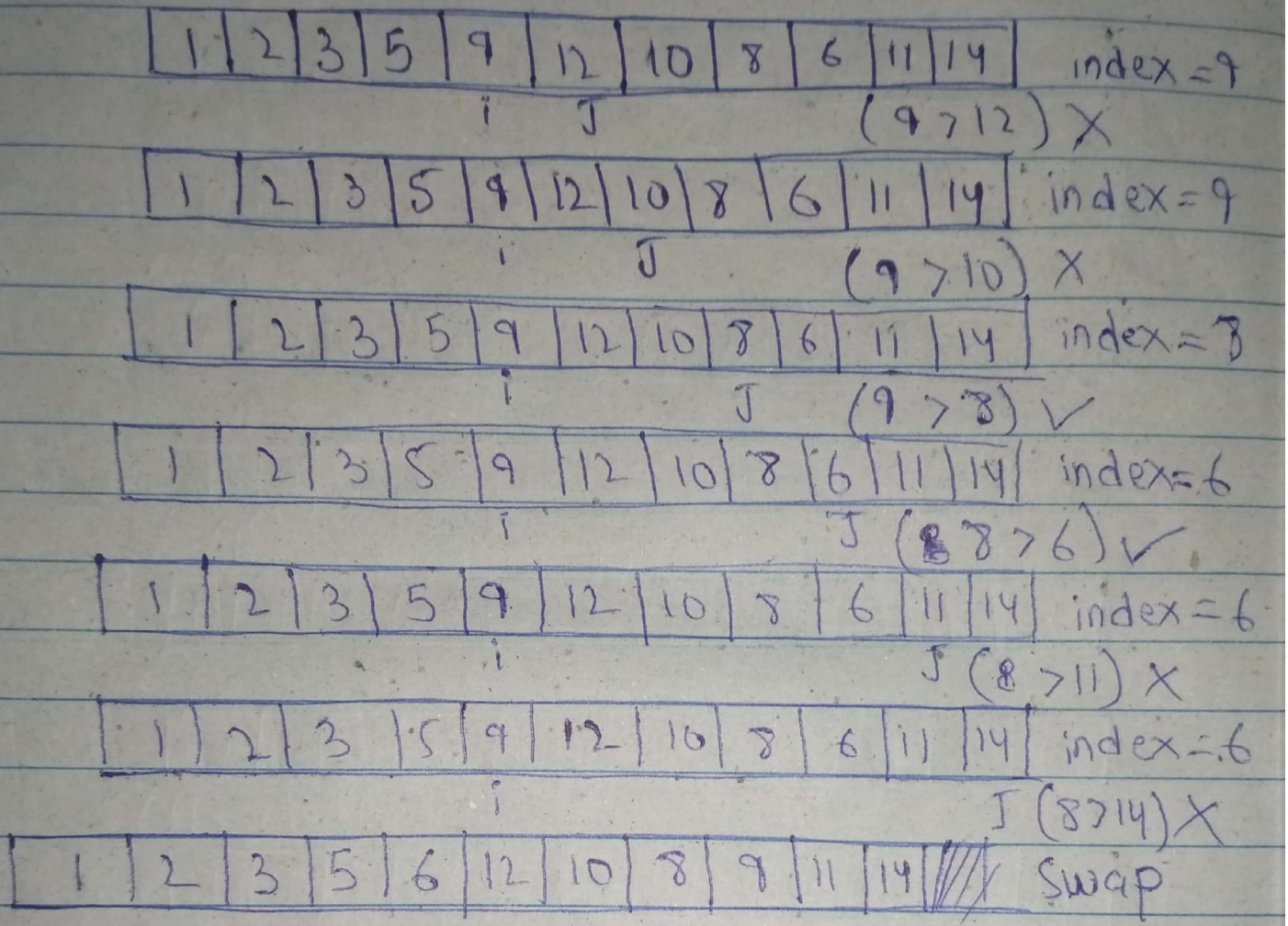
Iteration : 3



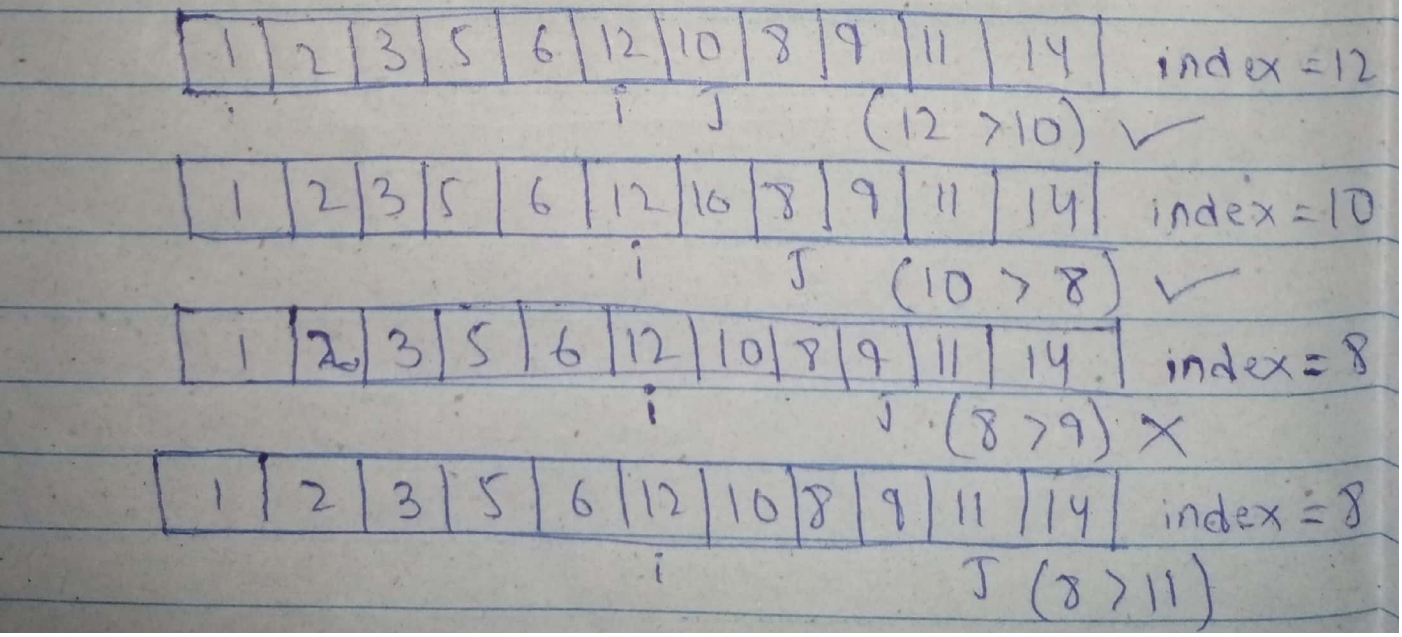
Iteration : 4

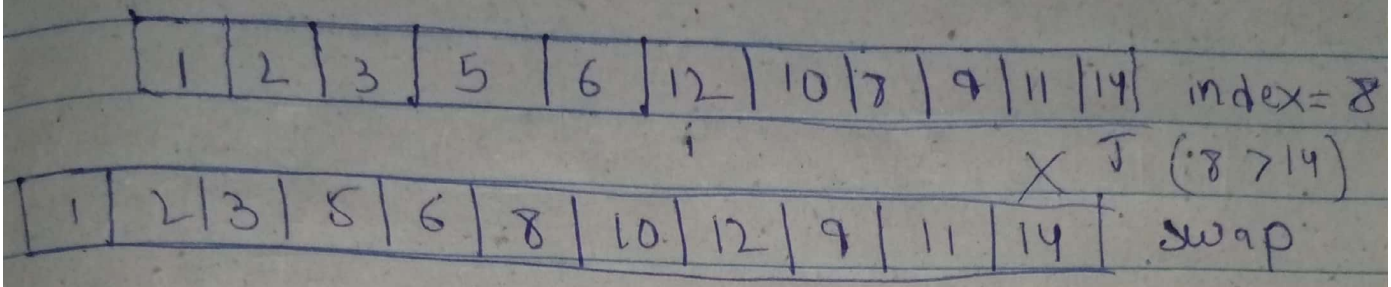


Iteration : 5

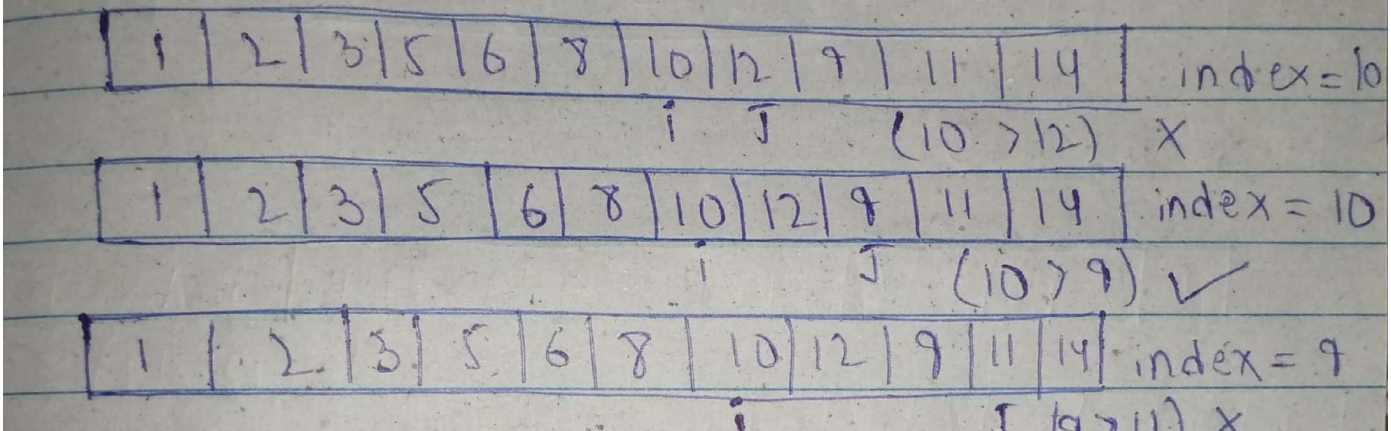


Iteration : 6

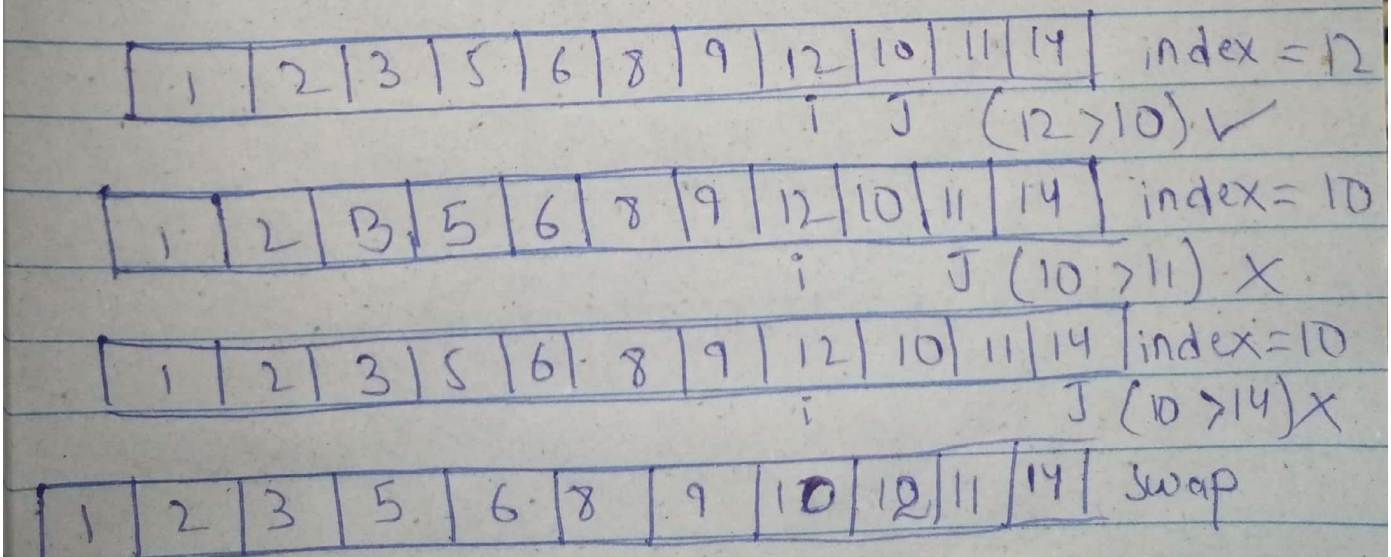




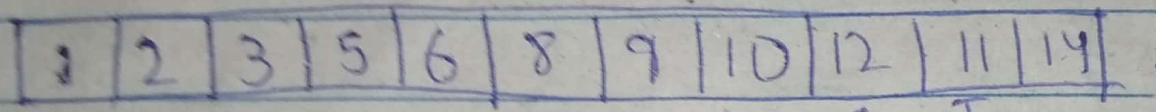
Iteration : 7

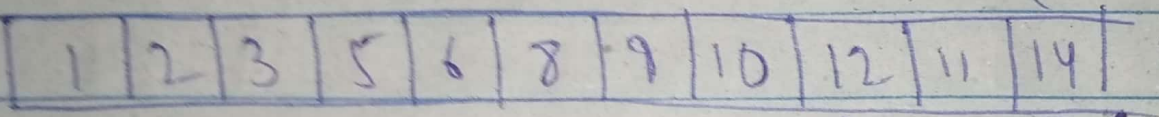


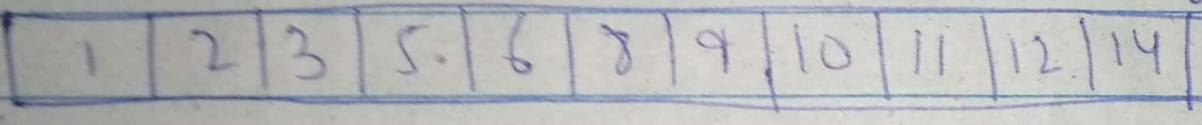
Iteration : 8



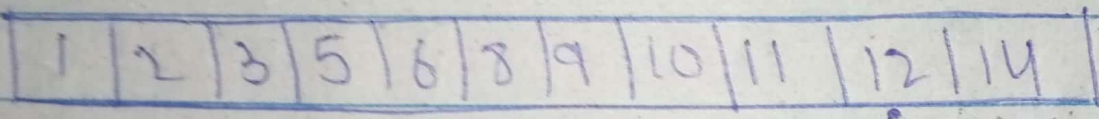
Iteration : 9.

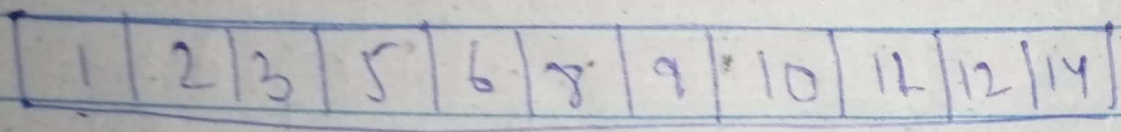






Iteration : 10





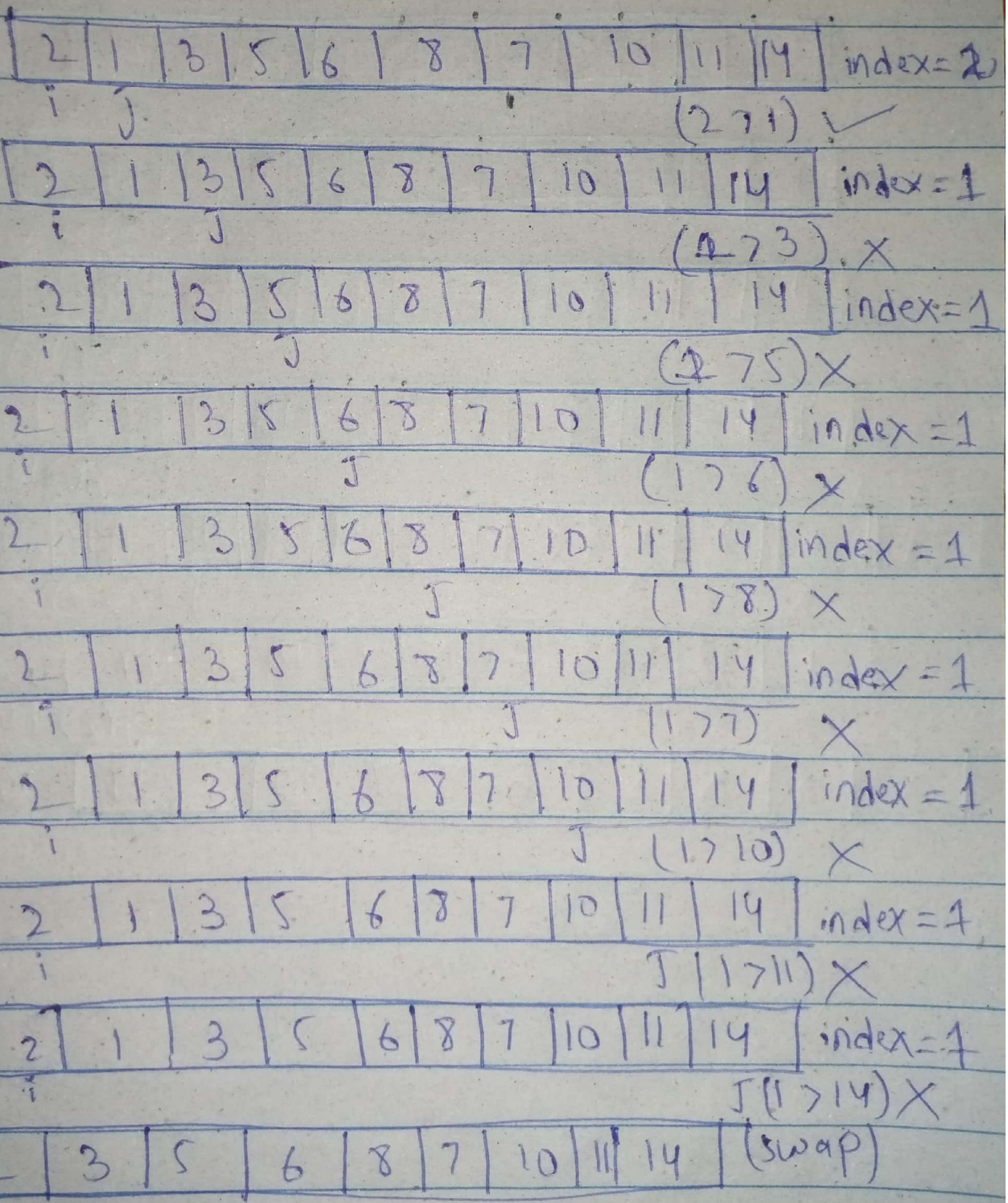
Sorted Array :

1, 2, 3, 5, 6, 8, 9, 10, 11, 12, 14

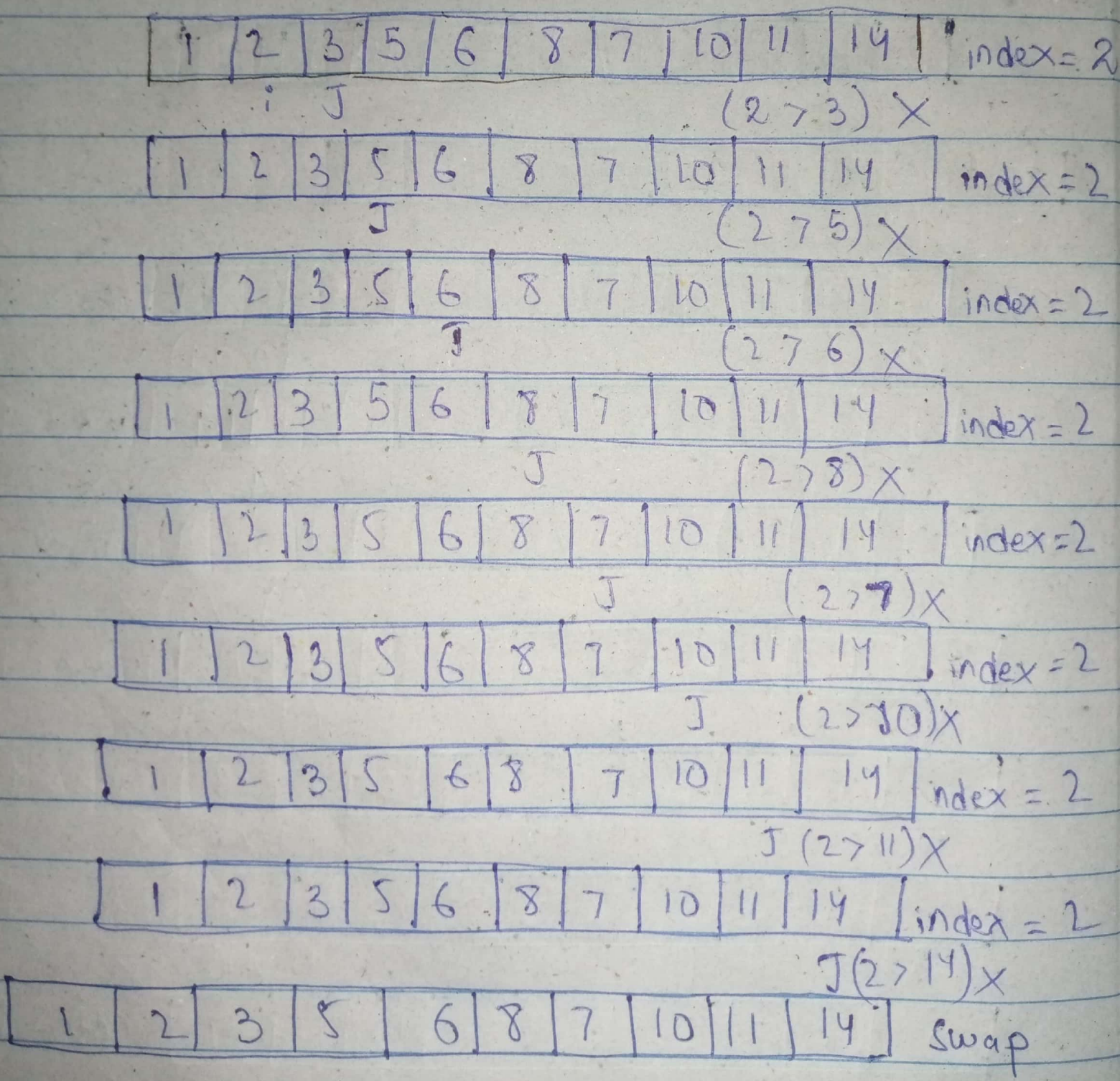
example # 2:

2	1	3	5	6	8	7	10	11	14

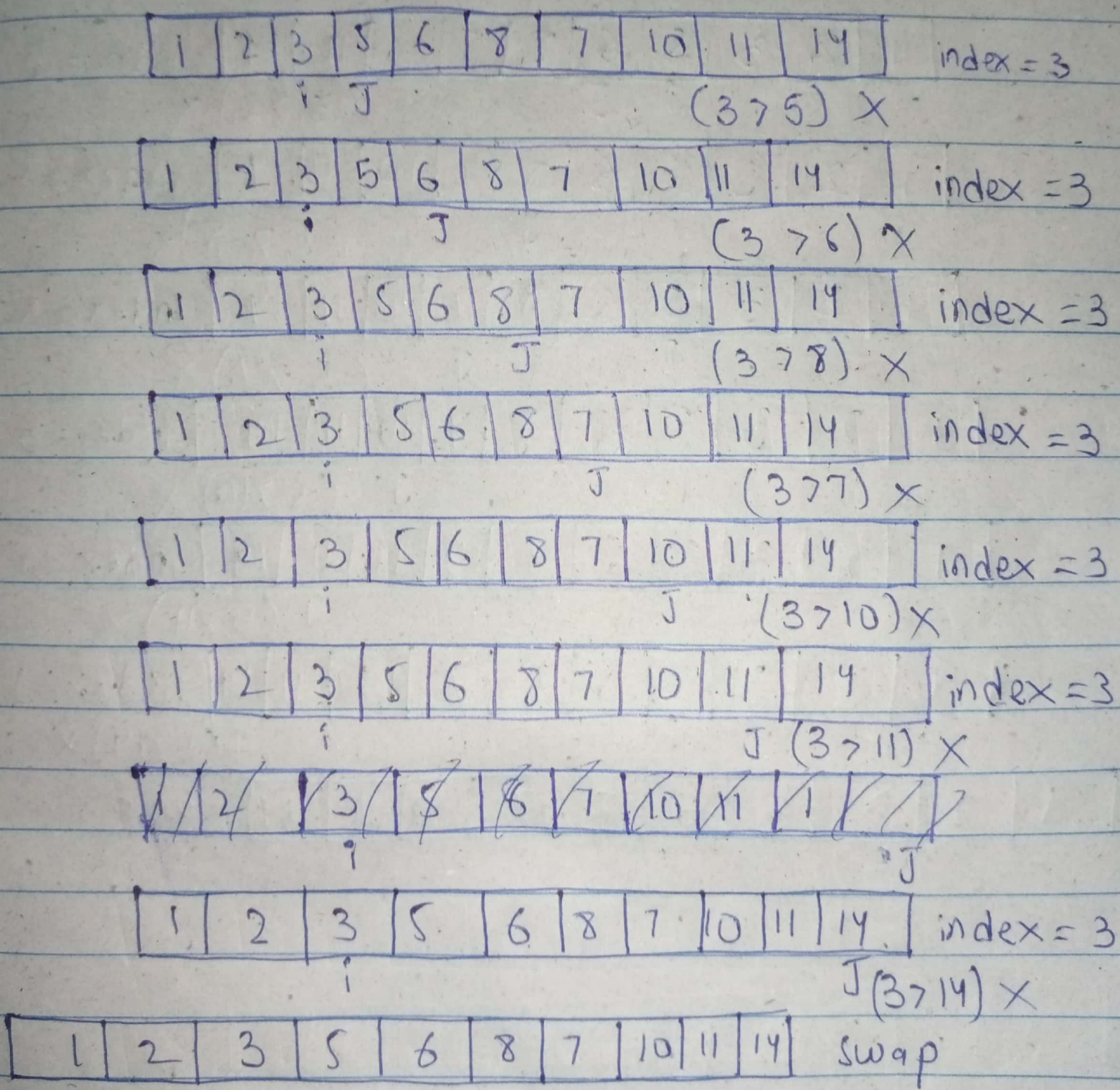
Iteration : 1



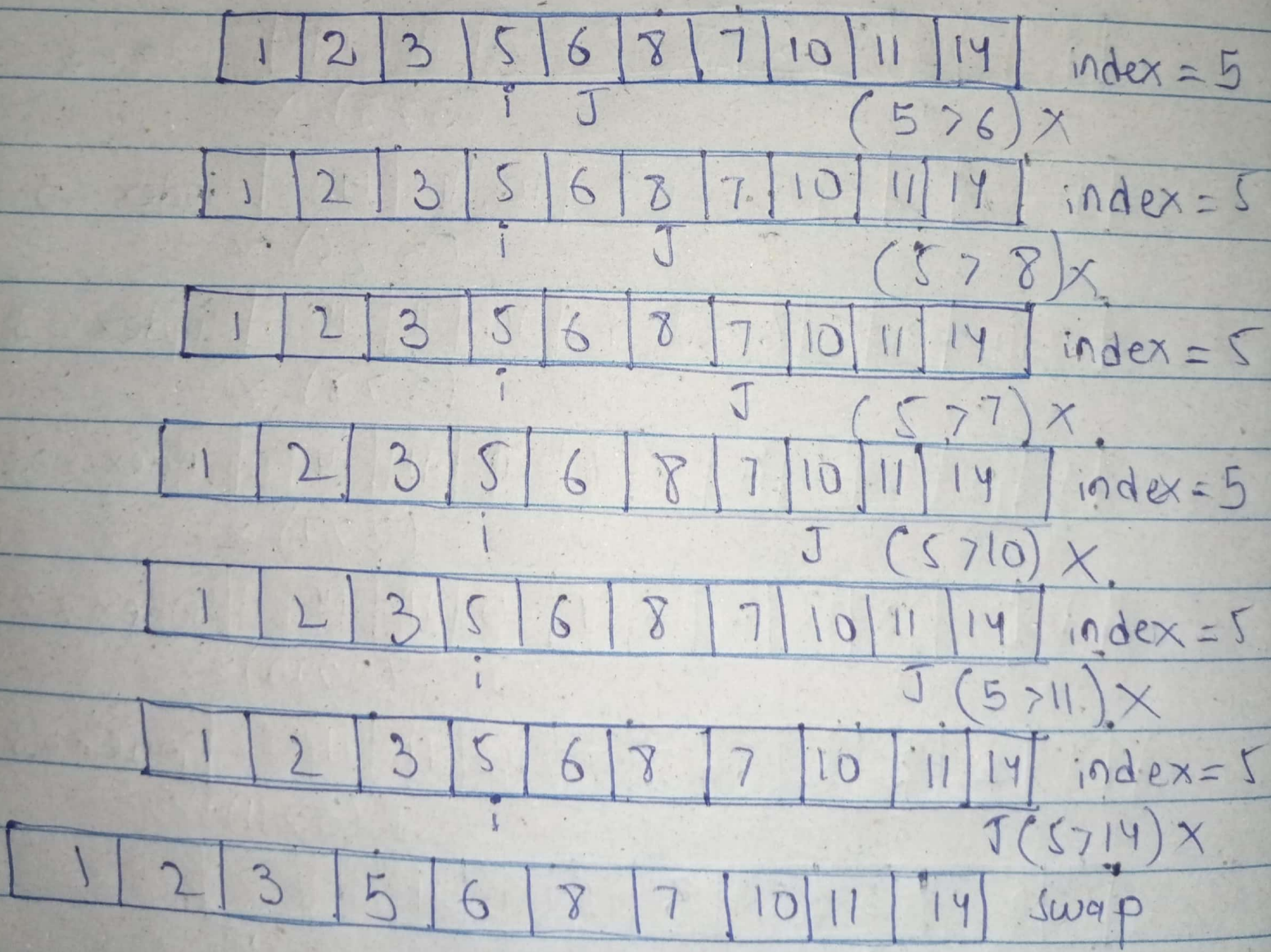
Iteration : 2 :



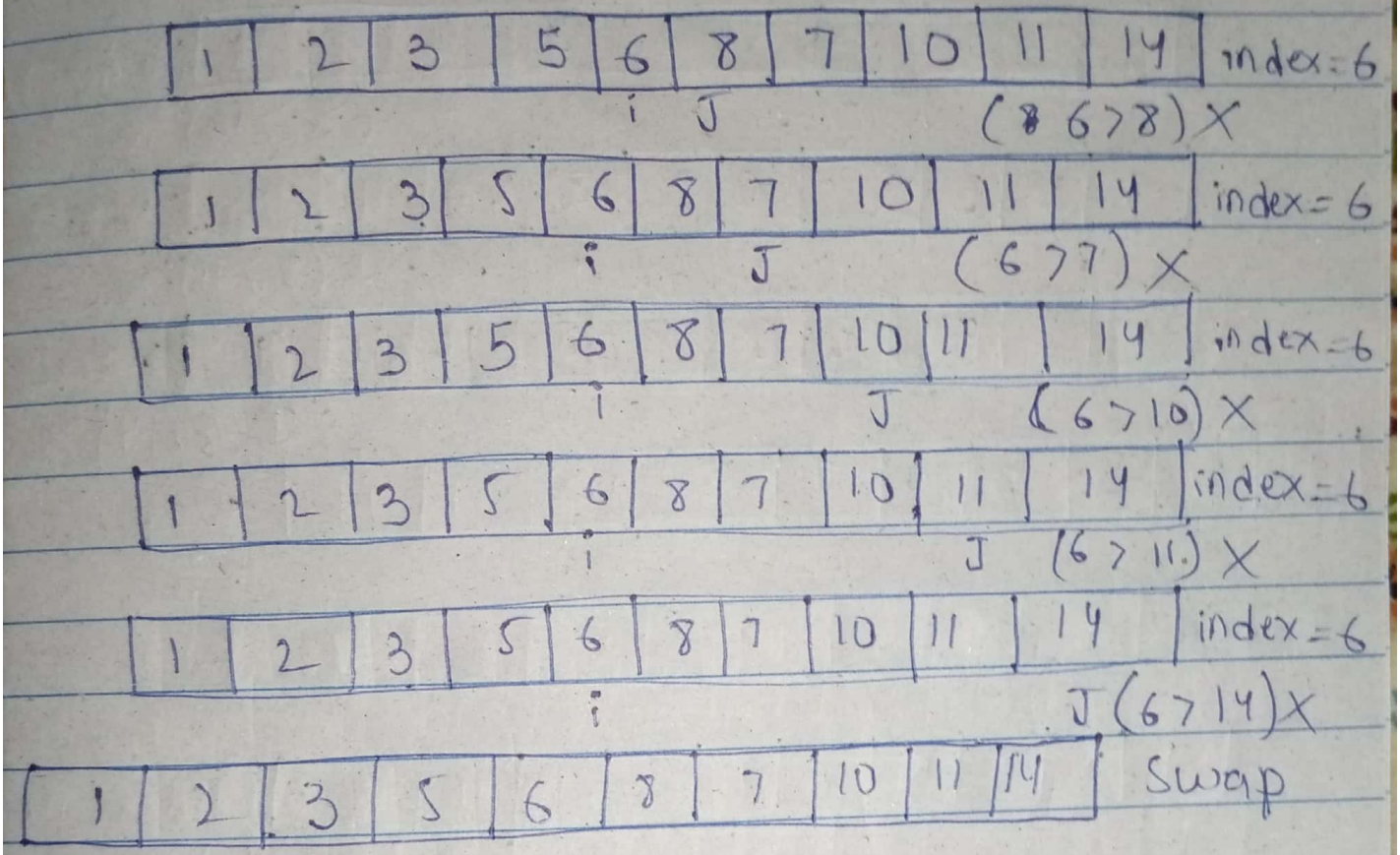
Iteration : 3



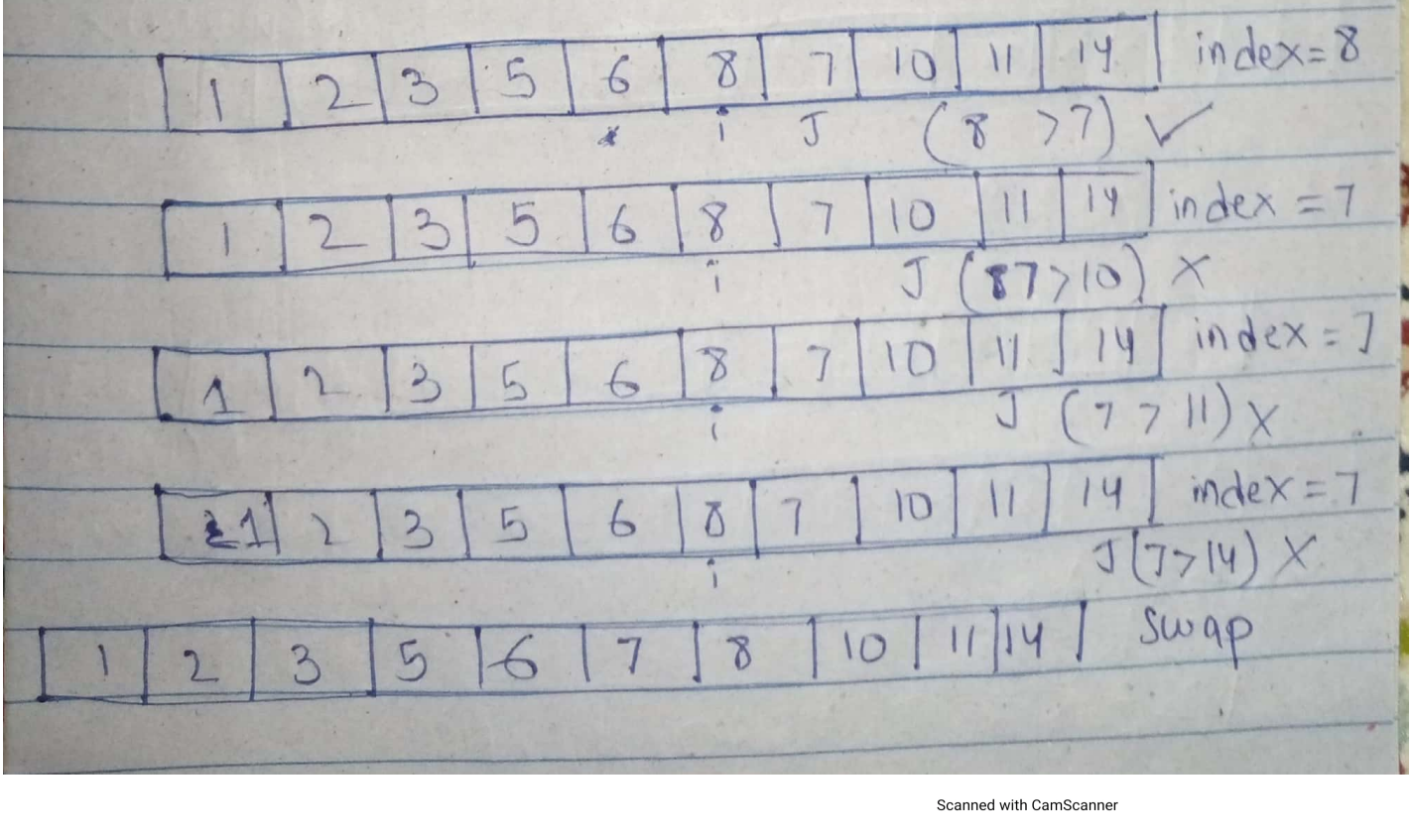
Iteration : 4



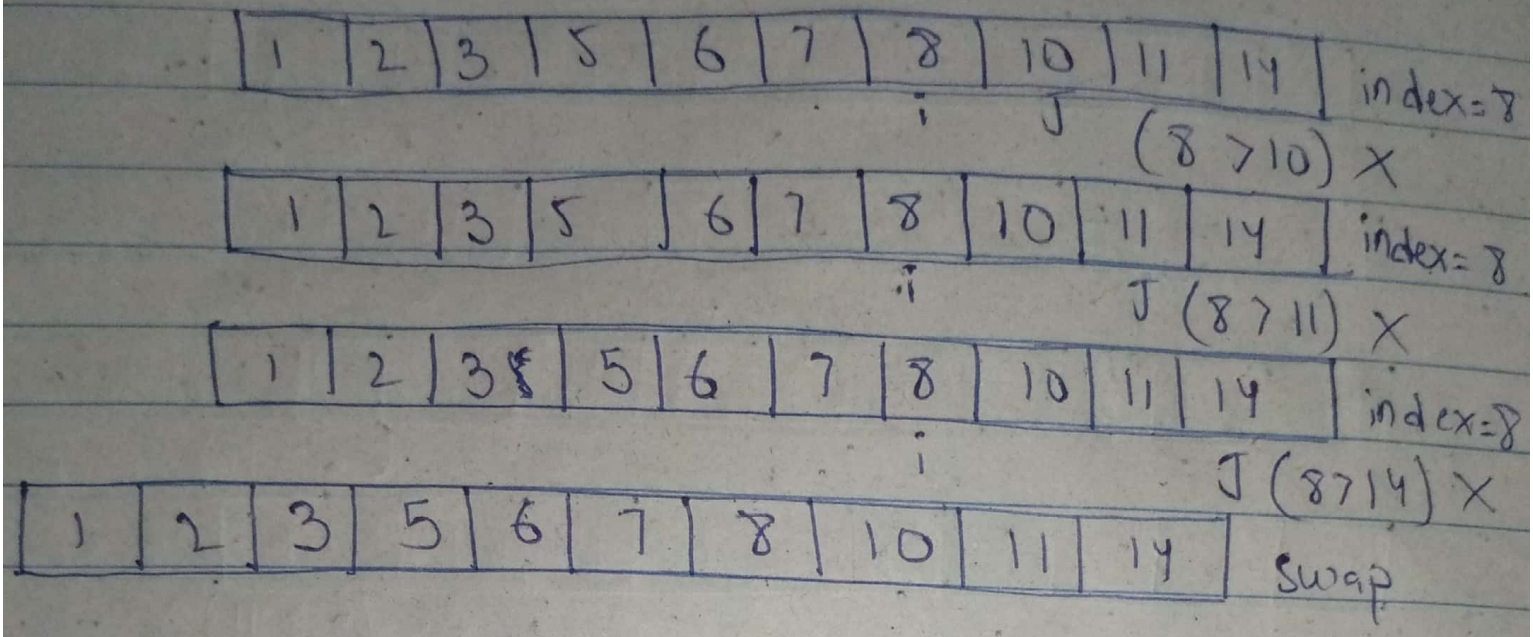
Iteration : 5



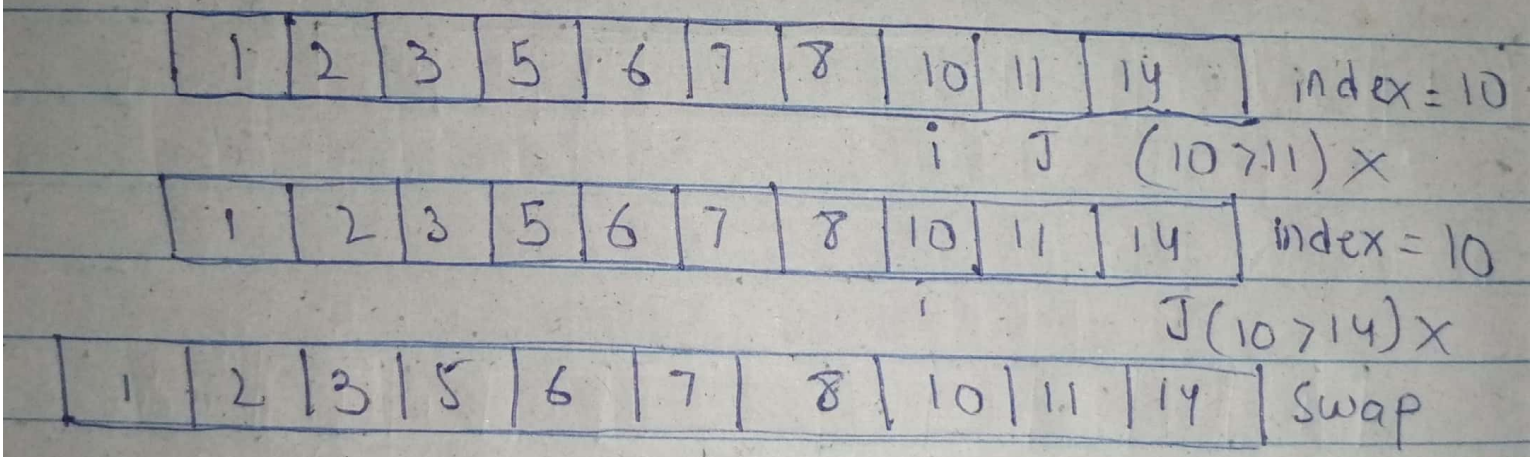
Iteration : 6



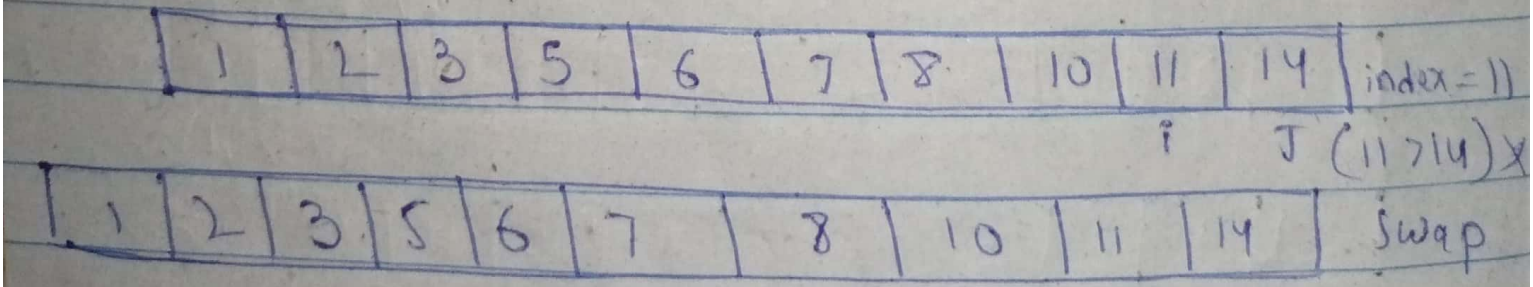
Iteration : 7



Iteration : 8



Iteration : 9



Sorted Array:

1, 2, 3, 5, 6, 7, 8, 10, 11, 14

TOPIC : Insertion Sort example: # 1

2	1	3	5	4	11	10	6	7

Iteration : 1

2	1	3	5	4	11	10	6	7

J i J=0 Key=1

(J>=0 && arr[j] > key)

(✓ && 12 > 1) ✓

2	2	3	5	4	11	10	6	7

J=-1

(J>=0 && arr[j] > (key)) X

1	2	3	5	4	11	10	6	7

arr[J+1] = key

Iteration : 2

1	2	3	5	4	11	10	6	7

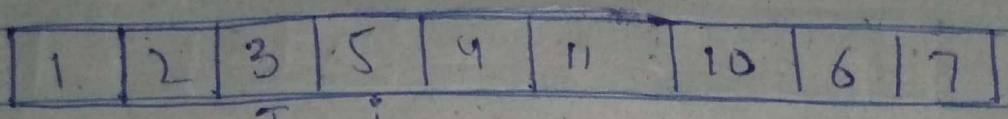
J i J= 1 Key=3

(1>=0 && 2 > 3) X

1	2	3	5	4	11	10	6	7

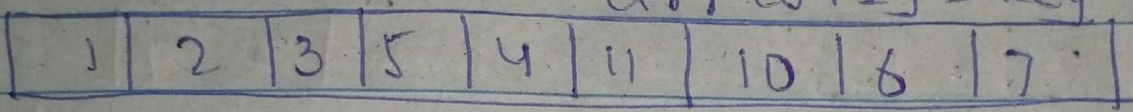
arr[J+1] = key

Iteration : 3

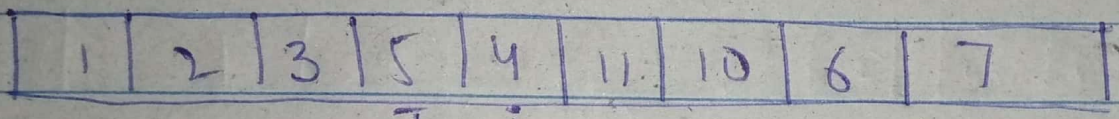


(2>=0 && 3>5) X

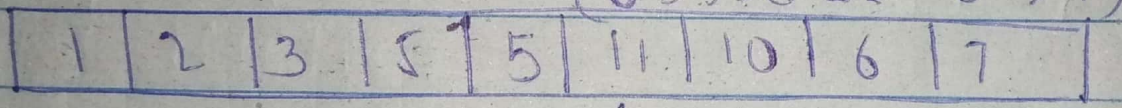
arr[J+1] = key



Iteration : 4

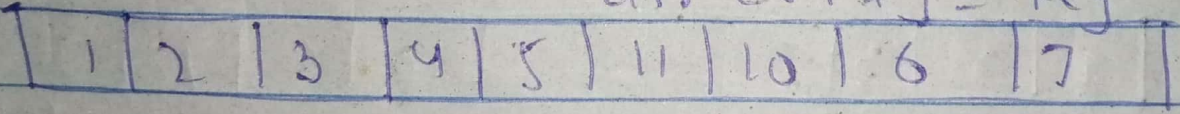


(3>=0 && 5>4) ✓

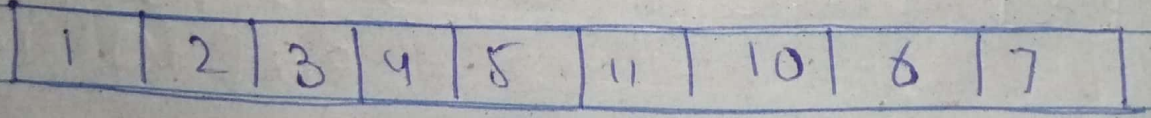


(2>=0 && 3>5) X

arr[J+1] = key

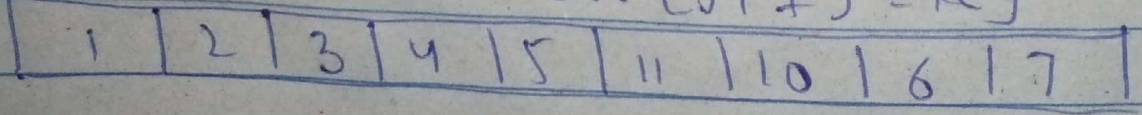


Iteration : 5

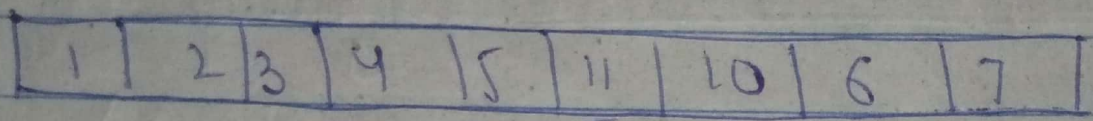


(4>=0 && 5>11) X

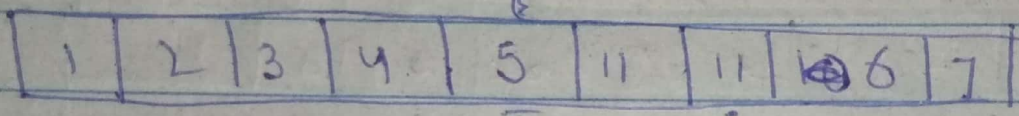
arr[J+1] = key



Iteration : 6

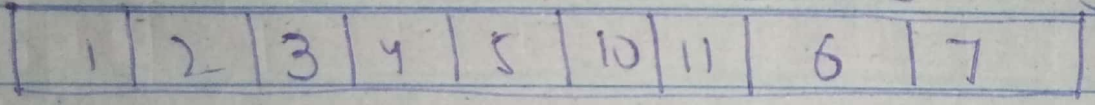


(5 >= 0 && 11 > 10) ✓

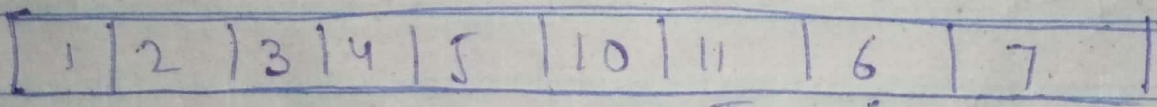


(4 >= 0 && 5 > 10) ✗

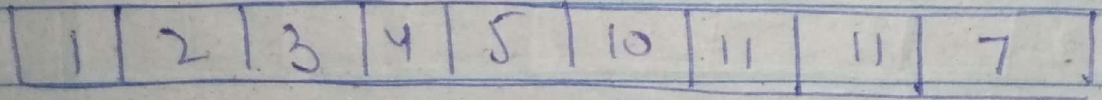
arr[4+1] = key



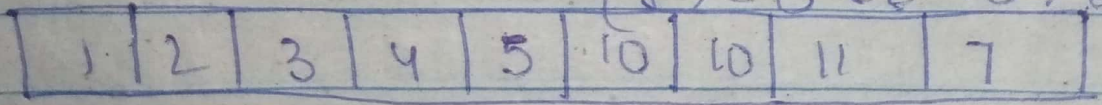
Iteration : 7



(6 >= 0 && 11 > 6) ✓

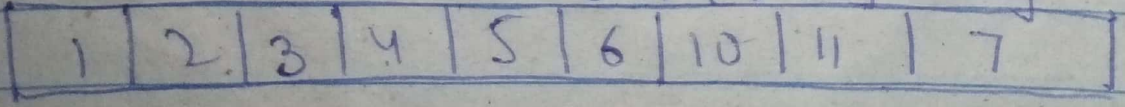


(5 >= 0 && 10 > 6) ✓

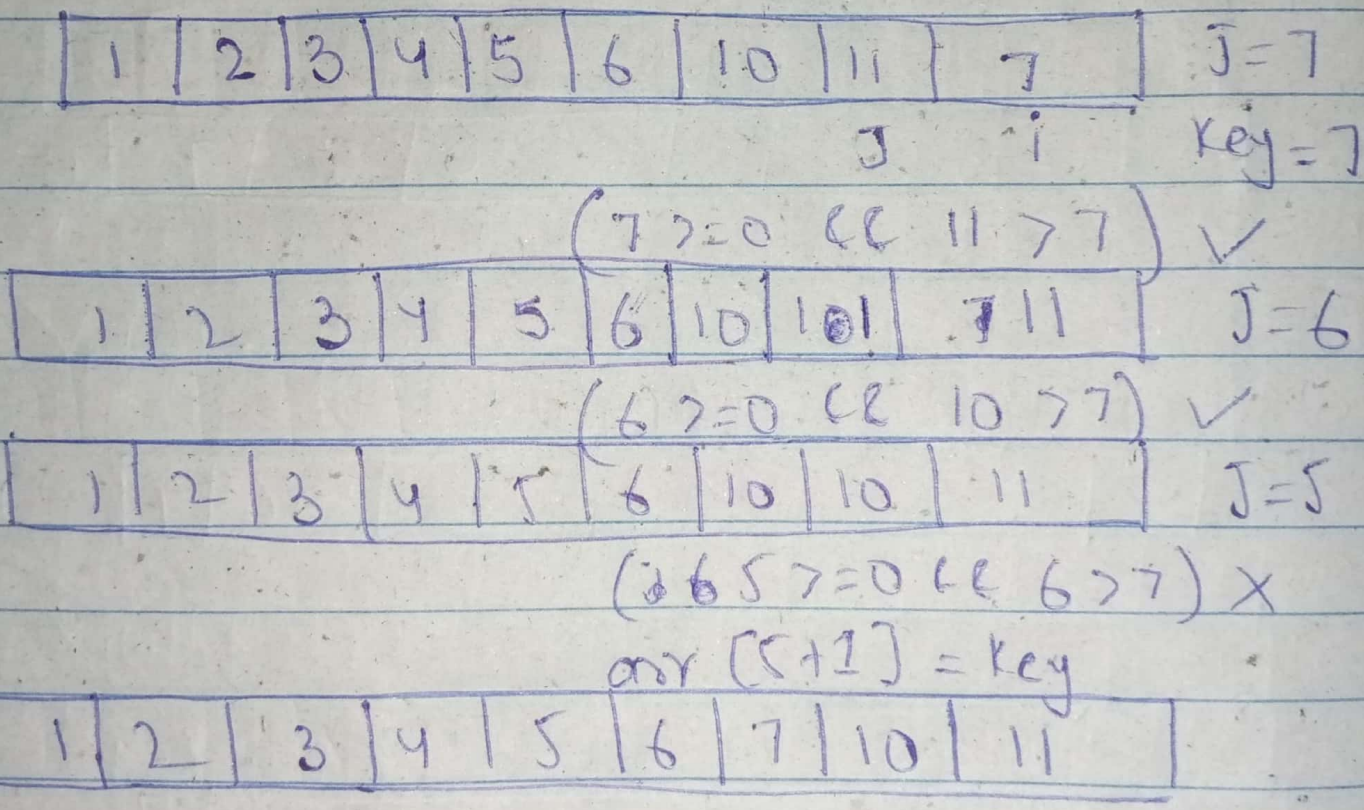


(4 >= 0 && 5 > 6) ✗

arr[J+1] = key



Iteration : 8



Sorted Array :

1, 2, 3, 4, 5, 6, 7, 10, 11

example # 2:

3	1	2	5	8	11	7	12	6	10

Iteration : 1

3	1	2	5	8	11	7	12	6	10

J i

J = 0

key = 1

(0 >= 0 && 3 > 1) ✓

3	3	2	5	8	11	7	12	6	10

J i

J = -1

(-1 >= 0 && 4) ✗

1	3	2	5	8	11	7	12	6	10

arr[-1+1] = key

Iteration : 2

1	3	2	5	8	11	7	12	6	10

J i

J = 1

key = 2

(1 >= 0 && 3 > 2) ✓

1	3	3	5	8	11	7	12	6	10

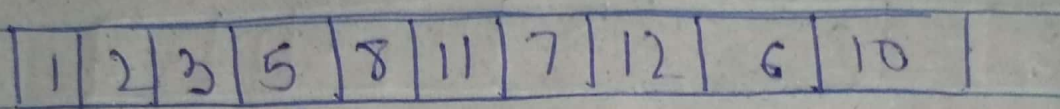
J = 0

(0 >= 0 && 1 > 2) ✗

arr[0+1] = key

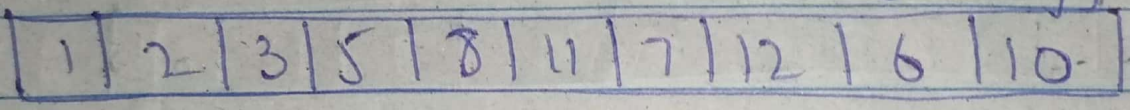
1	2	3	5	8	11	7	12	6	10

Iteration : 3

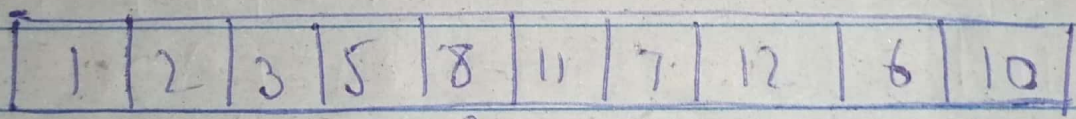


(2 >= 0 && 3 > 5) ×

arr [J+1] = key

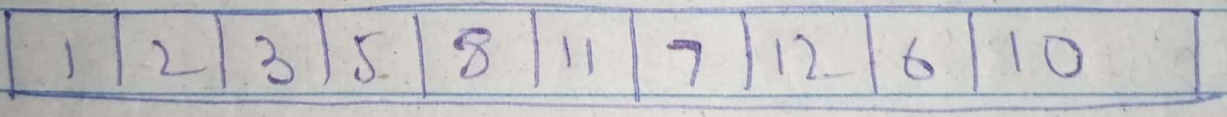


Iteration : 4

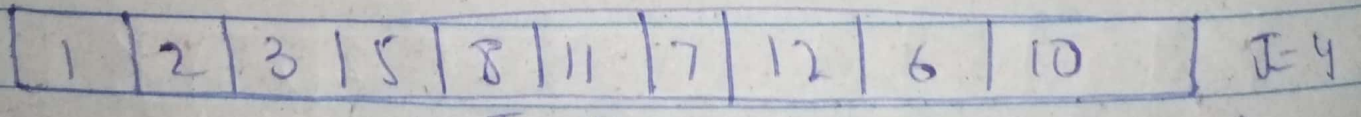


(3 >= 0 && 5 > 8) ×

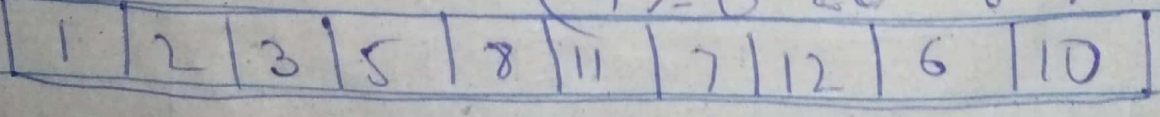
arr [3+1] = 8



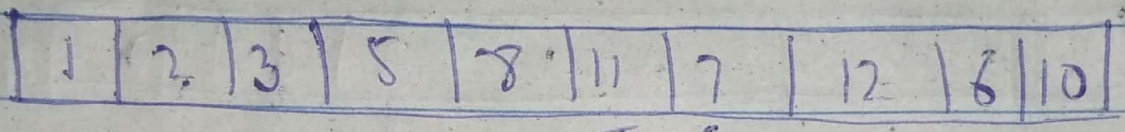
Iteration : 5



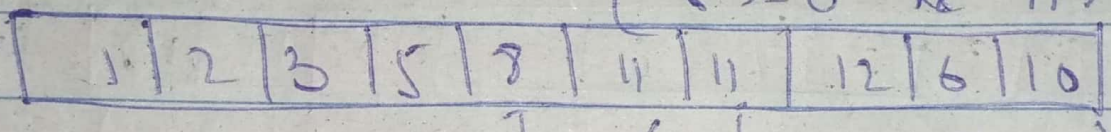
(4 >= 0 && 8 > 11) ×



Iteration : 6



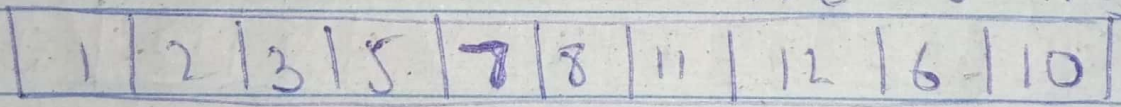
(5>=0 && 11>7) ✓



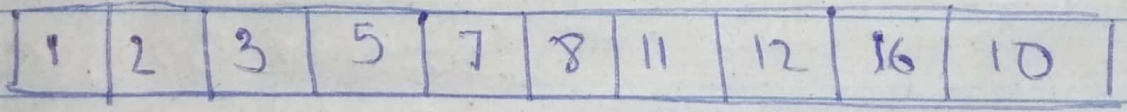
(4>=0 && 8>7) ✓

(3>=0 && 5>7) ✗

arr[3+1] = key

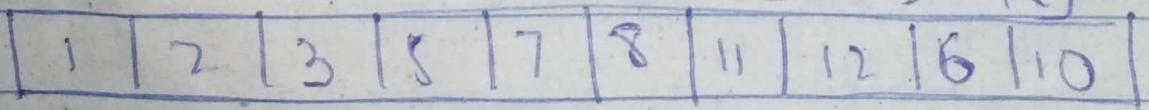


Iteration : 7

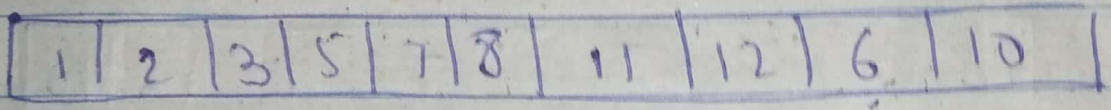


(6>=0 && 11>12) ✗

arr[J+1] = key

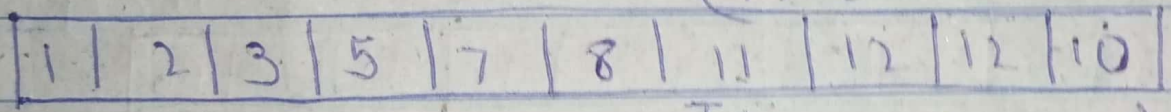


Iteration : 8



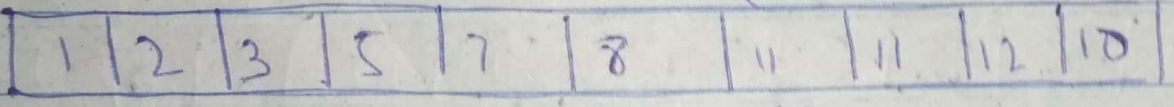
J = 7
key = 6

(7 >= 0 && 12 > 6) ✓



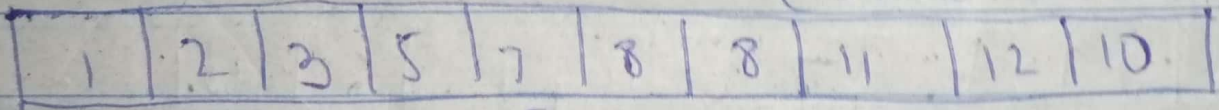
J = 6

(6 >= 0 && 11 > 6) ✓



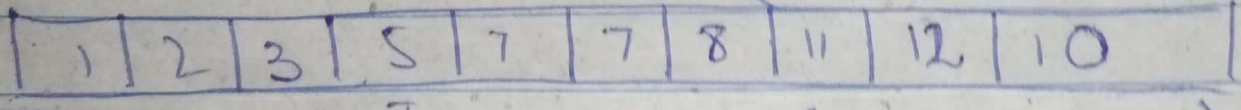
J = 5

(5 >= 0 && 8 > 6) ✓



J = 4

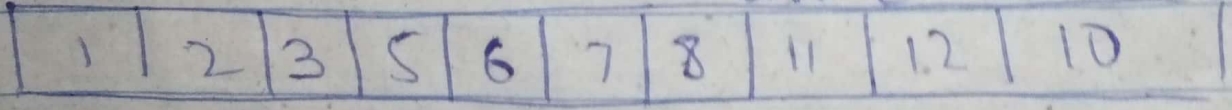
(4 >= 0 && 7 > 6) ✓



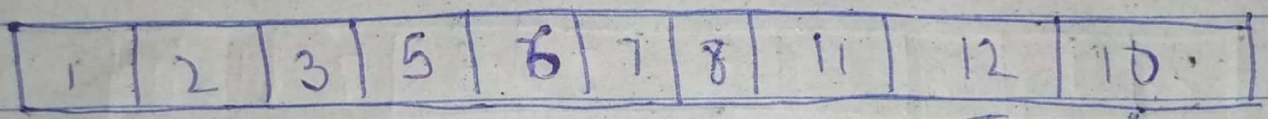
J = 3

(3 >= 0 && 5 > 6) ✗

arr [3+1] = key



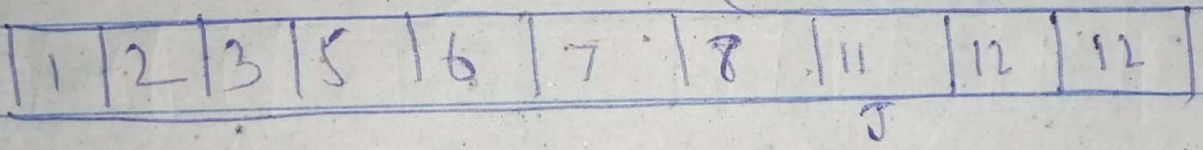
Iteration : 9



J = 8

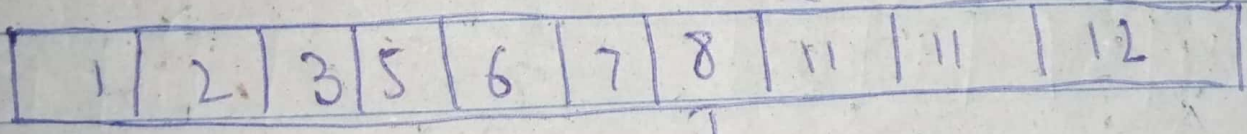
Key = 10

(8 >= 0 && 12 > 10) ✓



J = 7

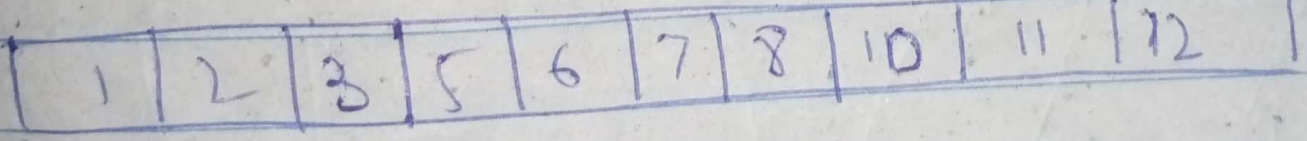
(7 >= 0 && 11 > 10) ✓



J = 6

(6 >= 0 && 8 > 10) X

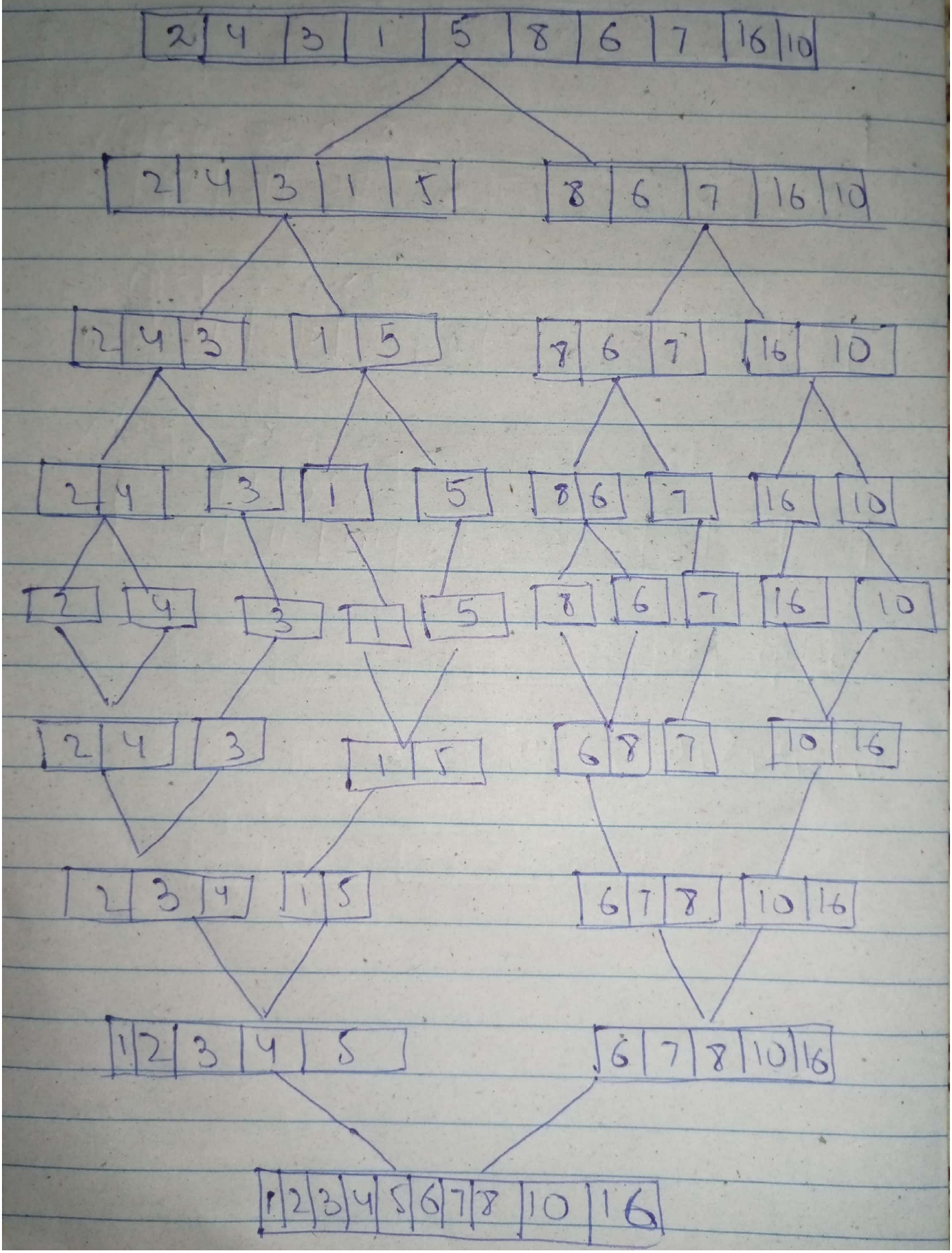
arr[6+1] = key



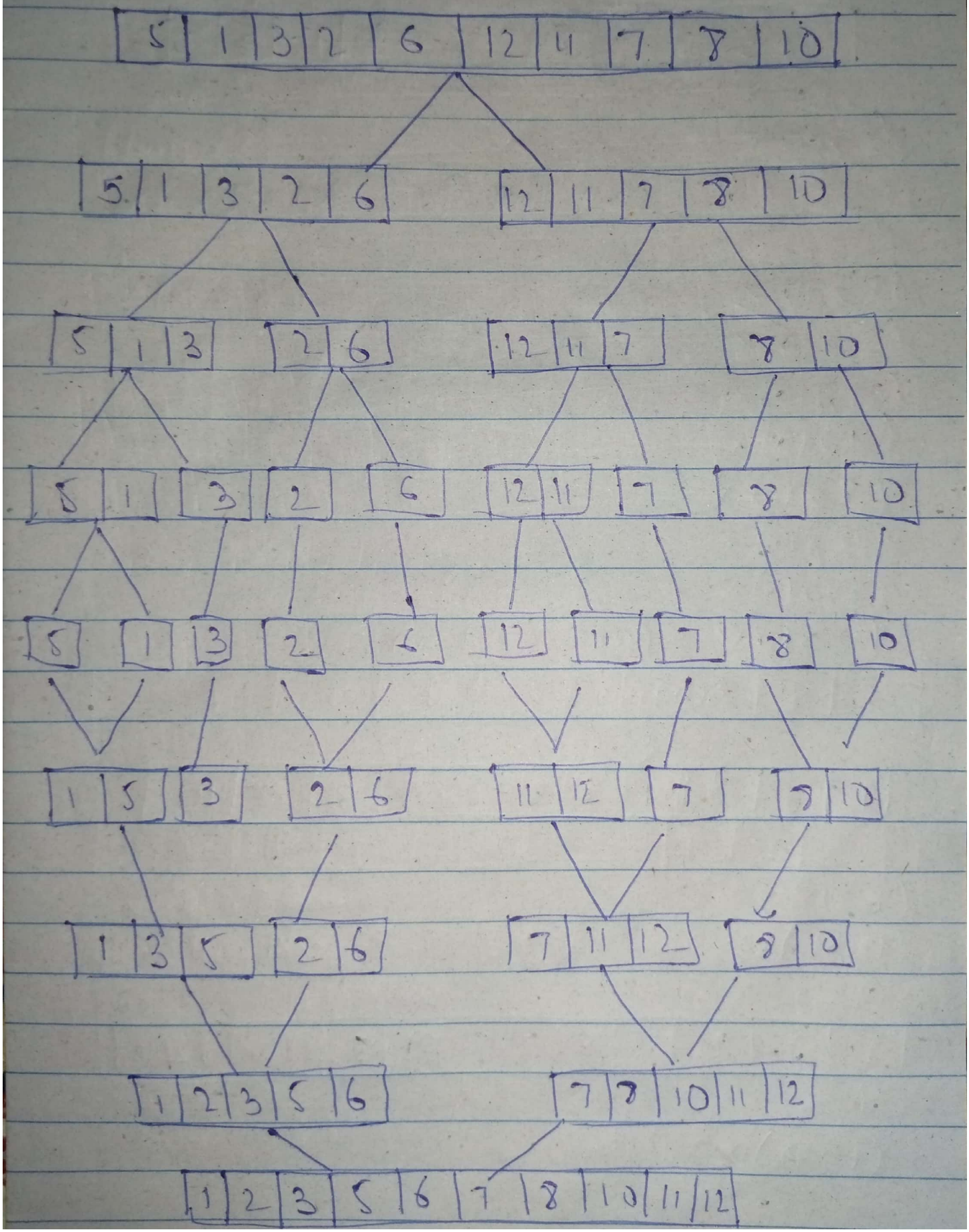
Sorted Array :

1, 2, 3, 5, 6, 7, 8, 10, 11, 12

Topic: Merge Sort: example # 1



example # 2



Topic: Quicksort example #1

0	1	2	3	4	5	6	7	8	9	10
1	3	2	4	8	7	5	9	12	6	10

i → i → i → i → i → i → i → i J Pivot

(i > pivot) (J < pivot)

if I stop then the J run after it.

(1 > 10) x (6 < 10) ✓

(3 > 10) x

(2 > 10) x

(4 > 10) x

(8 > 10) x

(7 > 10) x

(5 > 10) x

(12 > 10 ✓ now J will run

swap i with J

1	3	2	4	8	7	5	9	6	12	10

J i → J i Pivot

(6 > 10) x (6 < 10) ✓

(12 > 10) ✓ (i > J) Pivot with will swap with i

1	3	2	4	8	7	5	9	6	10	12

i J Pivot

(1 > 12) x

1	3	2	4	8	7	5	9	6	10	12

i → i → i → i → i → i → i → i → iJ → i Pivot

(3 > 12) X

(2 > 12) X

(4 > 12) X

(8 > 12) X

(5 > 12) X

(9 > 12) X

(6 > 12) X

(10 > 12) X

(12 > 12) X

i > J now Pivot swap with

Sorted

1	3	2	4	8	7	5	9	6	10	12

i → i → i → i → i → i → i → i → iJ → i Pivot

(1 > 10) X

(3 > 10) X

(2 > 10) X

(4 > 10) X

(8 > 10) X

(7 > 10) X

(5 > 10) X

(9 > 10) X

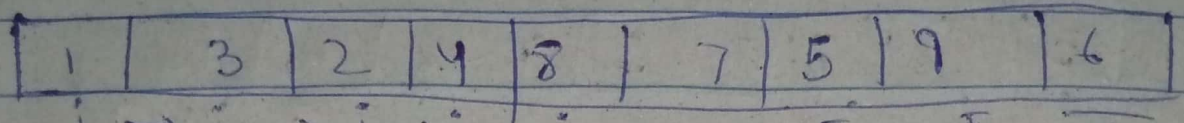
(6 > 10) X

i > J Pivot will swap

Sorted

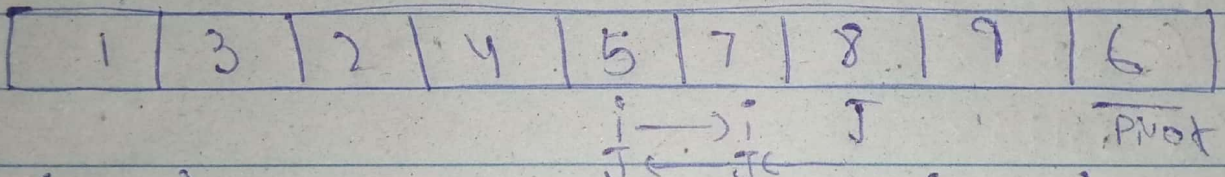
1	3	2	4	8	7	5	9	6	10	12

J Pivot



(1 > 8) X
(3 > 6) X
(2 > 6) X
(4 > 6) X
(8 > 6) ✓ Now J

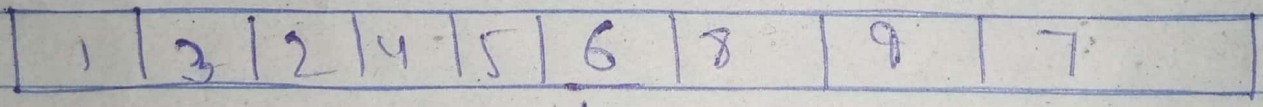
(9 < 6) X
(5 < 6) ✓



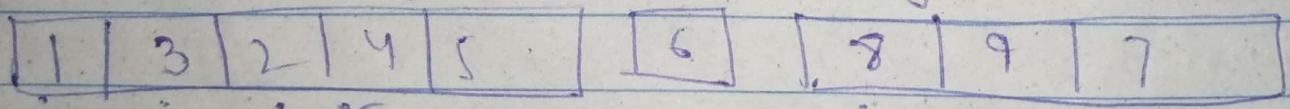
(5 > 6) X
(7 > 6) ✓

(7 < 6) X
(5 ≯ 6) ✓

(i > J) ✓ Pivot swap



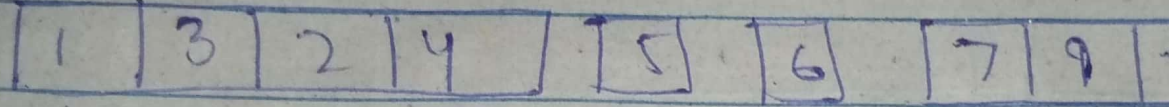
left small value Righ big values



(1 > 5) X
(3 > 5) X
(2 > 5) X
(4 > 5) X
(5 > 5 X (i > J) Pivot swap

(8 > 8) X
(9 > 7)
(8 > 7) ✓
(i ≯ J)

(9 < 7) X
(8 < 7) X
Pivot swap



i → i → i J i pivot

(1 > 4) x

(3 > 4) x

(2 > 4) x

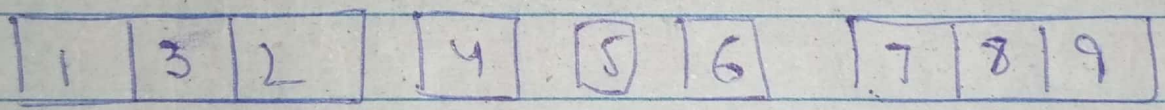
(4 > 4) x (i > J) swap

J i → iU pivot

(7 > 8) x (9 < 8) x

(9 > 8) ✓ (7 < 8) ✓

(i > J) pivot swap



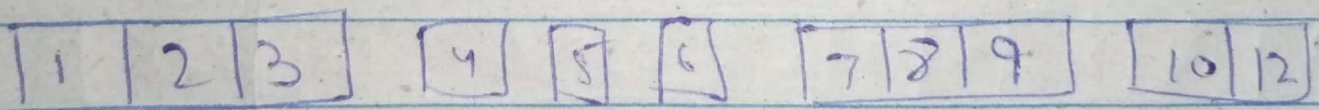
j i → iJ pivot

(1 > 2) x (3 < 2)

(3 > 2) ✓ (1 < 2) ✓

(i > J) swap

sorted



sorted array:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12

example # 2

0	1	2	3	4	5	6	7	8	9
9	8	2	1	7	5	10	6	20	12

i → i → i → i → i → i → i → i, J Pivot

(9 > 12) ×
(8 > 12) ×
(2 > 12) ×
(1 > 12) ×
(7 > 12) ×
(5 > 12) ×
(10 > 12) ×
(6 > 12) ×
(20 > 12) ✓

(20 < 12) ×
(6 < 12) ✓

(i > J) swap pivot

Sorted

9	8	2	1	7	5	10	6	12	20

i → i → i → i → i → i → i → i J i → i Pivot

(9 > 12) ×
(8 > 12) ×
(2 > 12) ×
(1 > 12) ×
(7 > 12) ×
(5 > 12) ×
(10 > 12) ×
(6 > 12) ×
(12 > 12) ×

(i > J) swap pivot

Sorted.

9	8	2	1	7	5	10	6	12	20

i → J ← J Pivot

(9 > 6) ✓ swap (10 < 6) ×

(5 < 6) ✓

5	8	2	1	7	9	10	6	12	20

i → i J ← J ← J Pivot

(5 > 6) × swap (9 < 6) ×

(8 > 6) ✓ (7 < 6) ×

(1 < 6) ✓

5	1	2	8	7	9	10	6

i → j → iJ Pivot

j ←

(1 > 6) × (8 < 6) ×

(2 > 6) × (2 < 6) ✓

(8 > 6) ✓ (i > j) Pivot swap.

5	1	2	6	7	9	10	8	12	20

left small values Right big values

5	1	2	6	7	9	10	8	12	20

i → i → iJ i Pivot Pivot

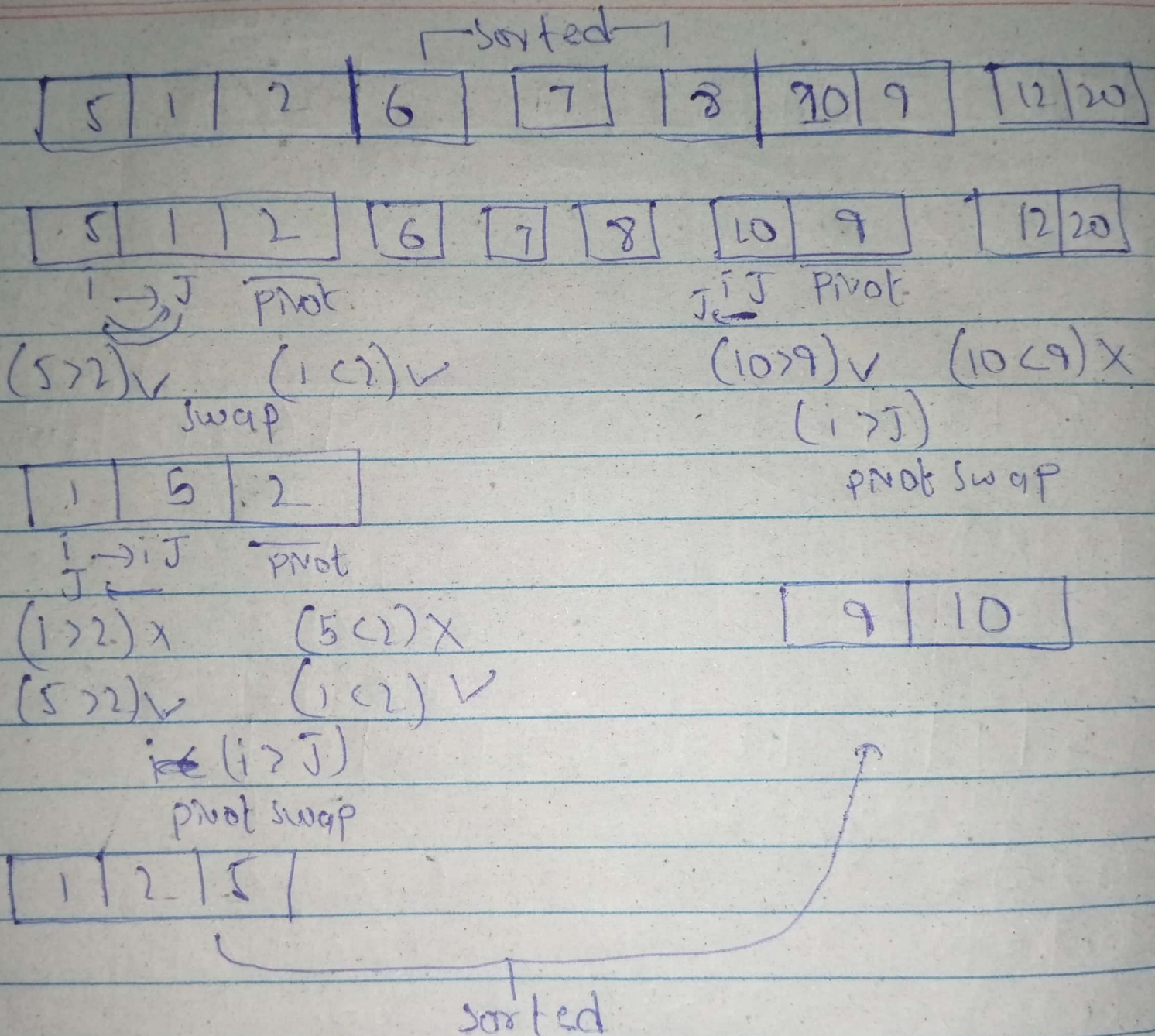
J ← J ← J

(5 > 6) ×

(1 > 6) × (i > J) swap pivot (9 > 6) ✓ (10 < 8) ×

(2 > 6) × (i > J) pivot swap (9 < 8) ×

(6 > 6) ×

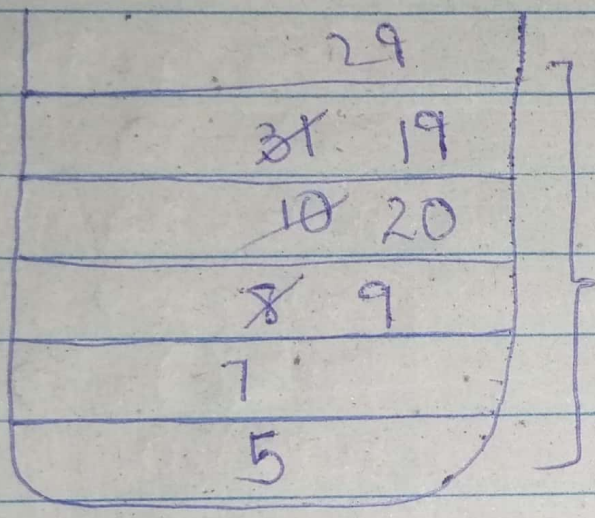


Sorted Array.

1, 2, 5, 6, 7, 8, 9, 10, 12, 20

TOPIC : Stack

example # 1.



POP : 31, 10, 8

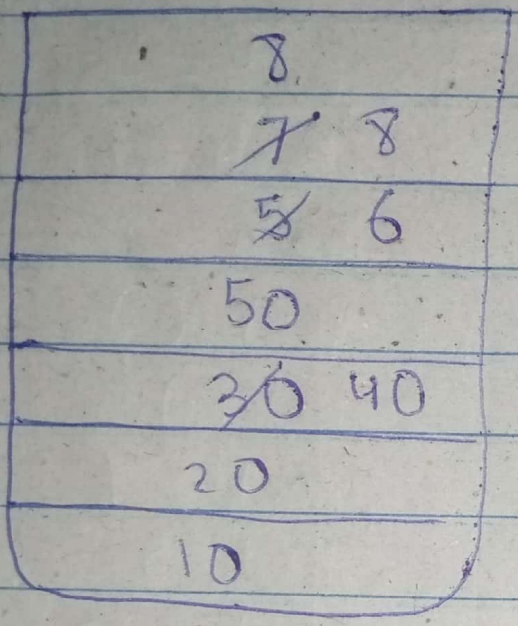
Push : 5, 7, 8, 9, 10, 20, 31, 19

29.

Peek :

Push = 5
Push = 7
Push = 8
Peek () = 8
POP ()
Peek = 7
Push = 9
Push = 10
Peek = 10
POP()
Push = 20
Push = 31
POP()
Peek = 20
Push = 19
Push = 29
isempty = false
is full = True

example #2



isempty = True
Push = 10
Push = 20
Push = 30
Peek() = 30
POP()30
Push = 40
isfull = false
Push = 50
Push = 5
POP()
Push = 6
Push = 7
Peek() = 7
isempty = True false
POP()
Push = 8
Push = 9
is full = True

Push = 10, 20, 30, 40, 50, 5, 6, 7, 8, 8
POP: 30, 5, 7,

Topic : Polish Notations: example # 1

Infix to PostFix:

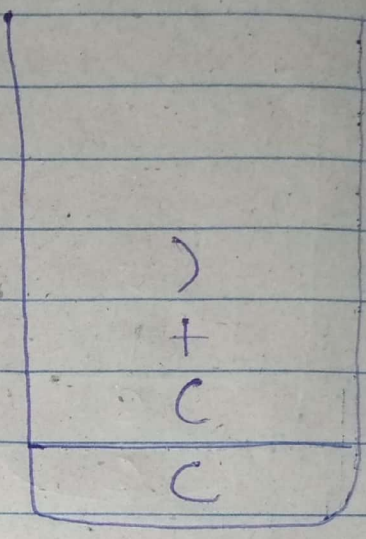
((A+B)÷(C+D))

Push = C, C

if Any Alphabat come dont push.

Resut

A B

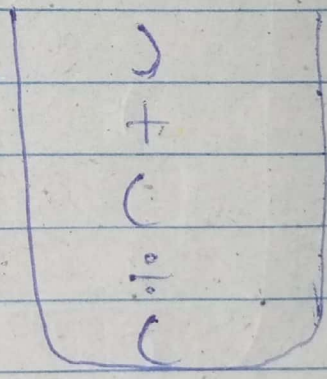


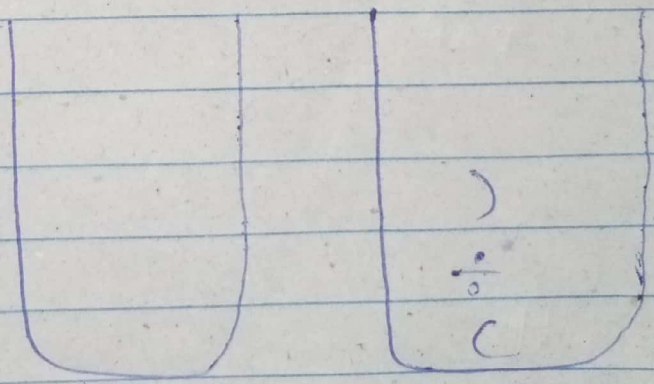
if Close bracket came pop unless open bracket came.

if any operator came pop that → and write in resut

Result

AB+CD+÷





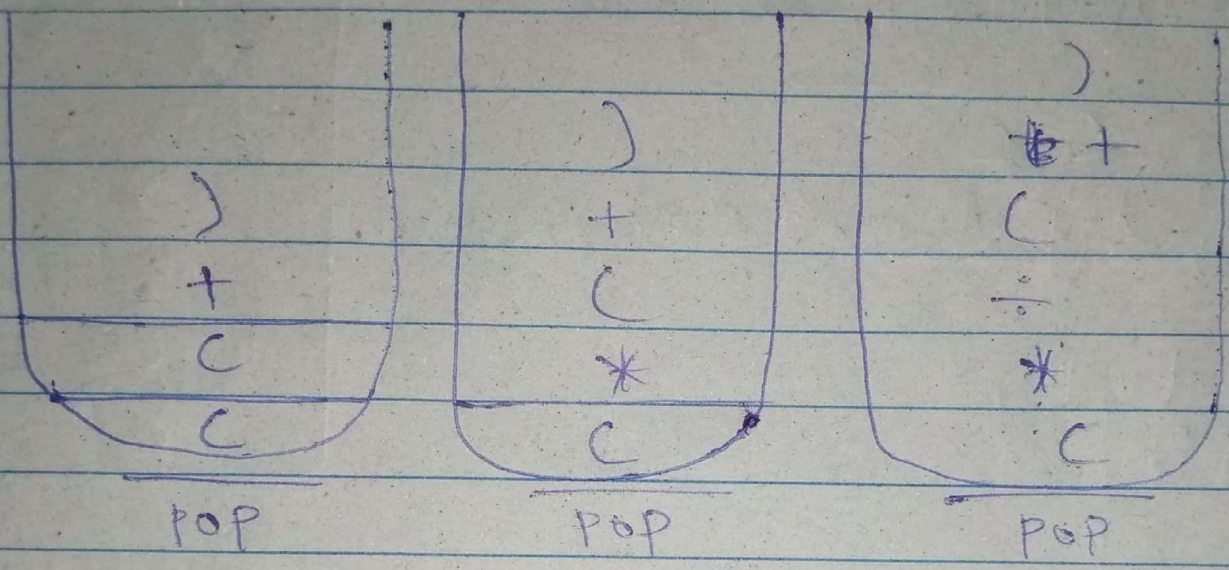
Stack is empty Now.

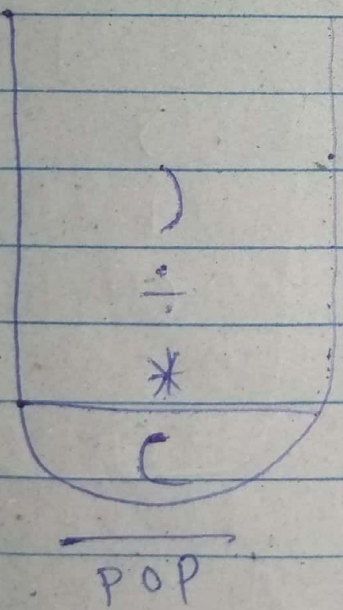
(Pop) till open bracket

example #2

Infix to postfix:

((A+B)*(C+D)÷(E+F))





Result

A B + C D + E F + ÷ *

Infix to Prefix

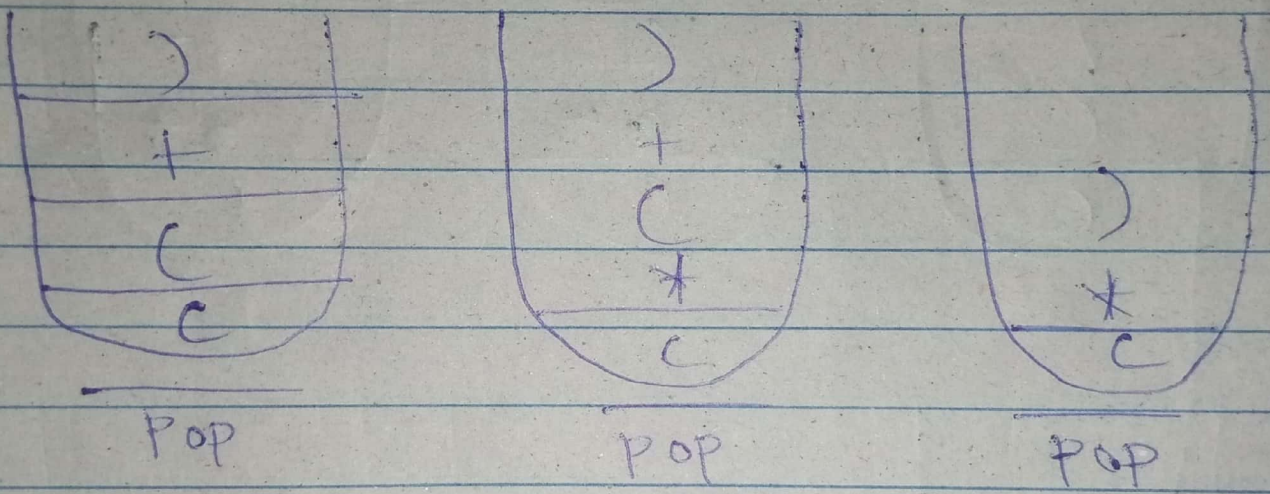
example # 1

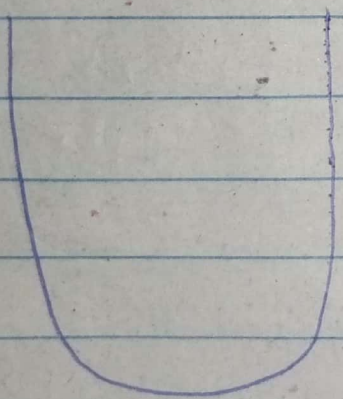
infix = ((A+B)*(C+D))

According to the Rule first reverse the whole equation.

((D+C)*(B+A))

Now do the postfix thing here.





→ Stack empty.

Result

DC+BA+* Now reverse the output

prefix :

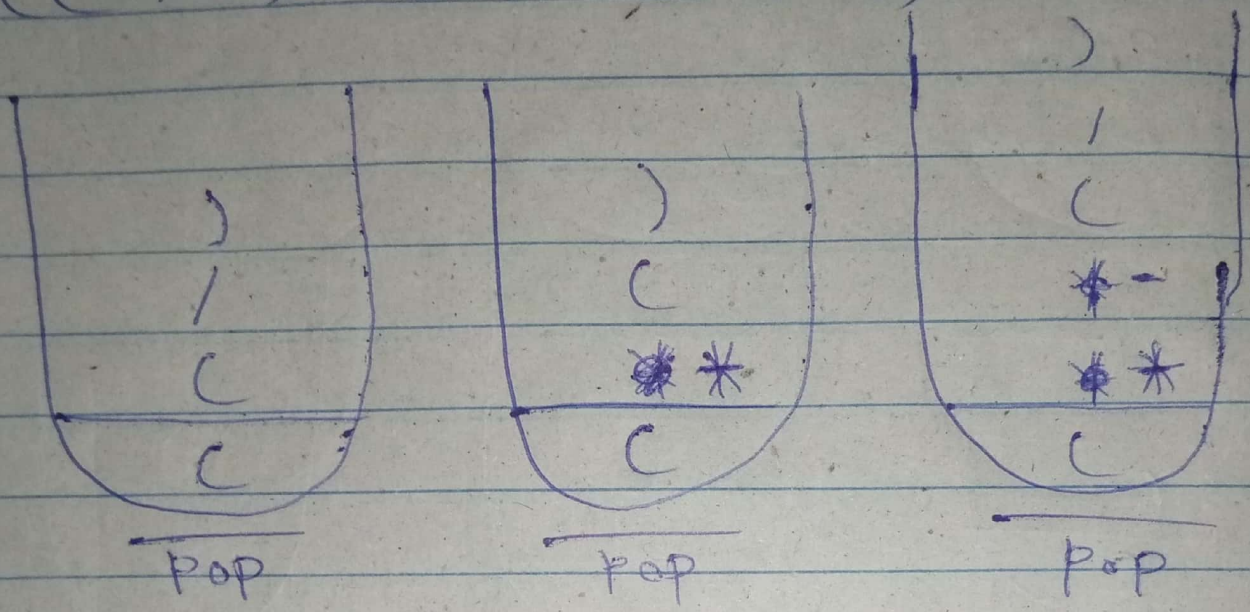
*+AB+CD

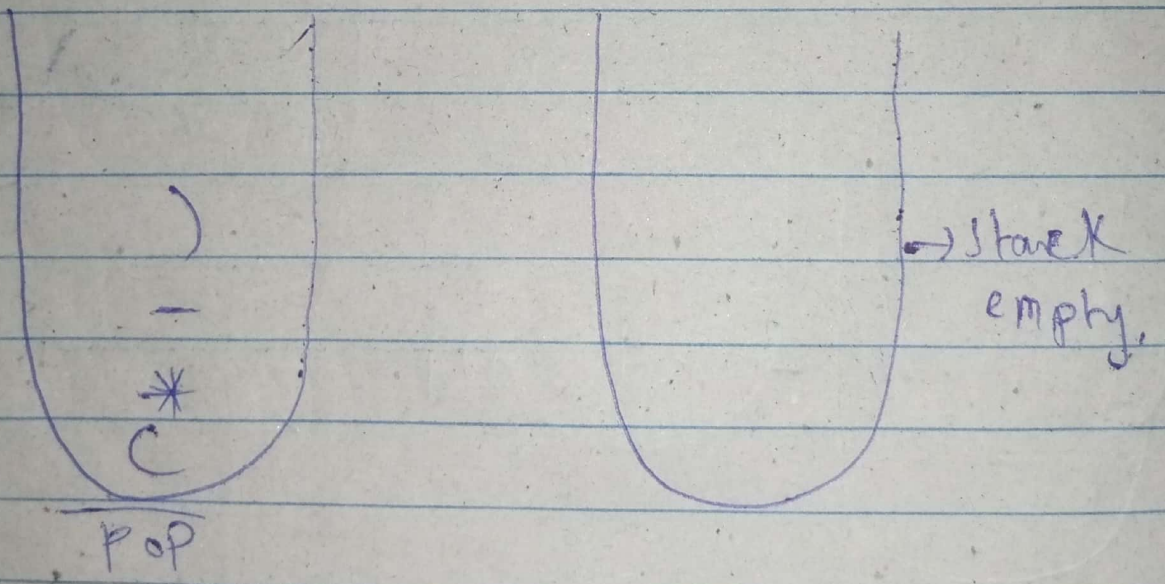
example # 2

Infix to prefix

Infix = ((a+b) - (c) * (d / e))

((e / d) * (c) - (b + a))





Result

ed / c b a / - *

Prefix

* - / a b c / d e

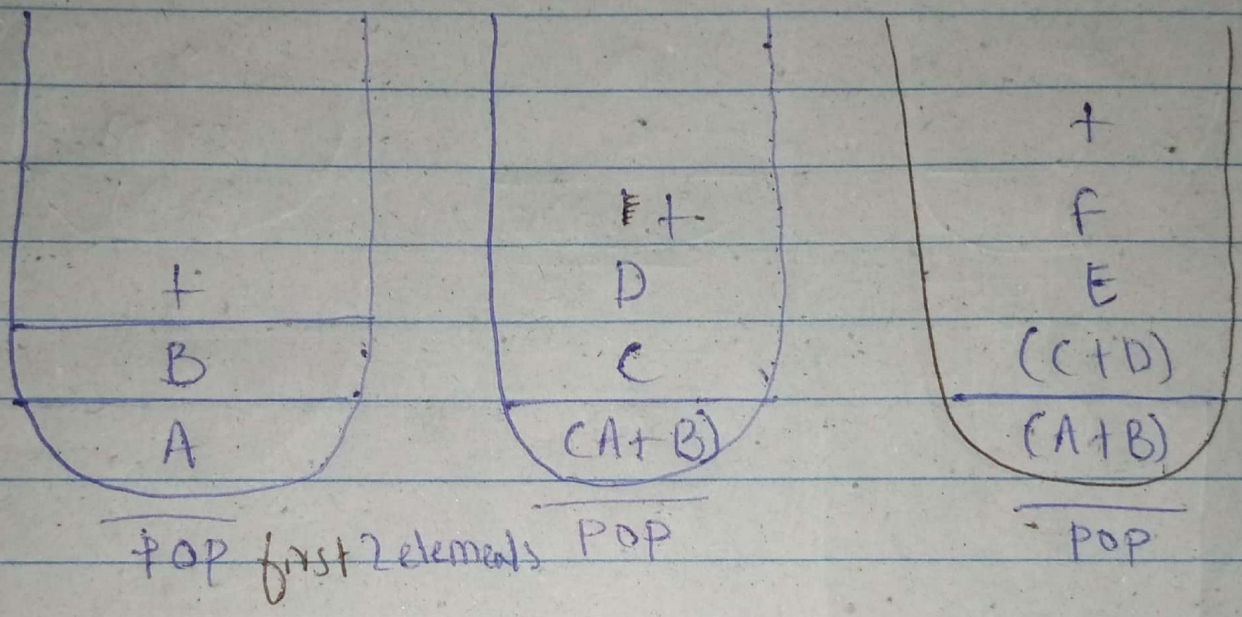
Topic: Postfix to Infix

example# 2

Postfix: ed/c/de/-*

Postfix

AB+CD+EF+÷*



pop first 2 elements

pop

pop

pop

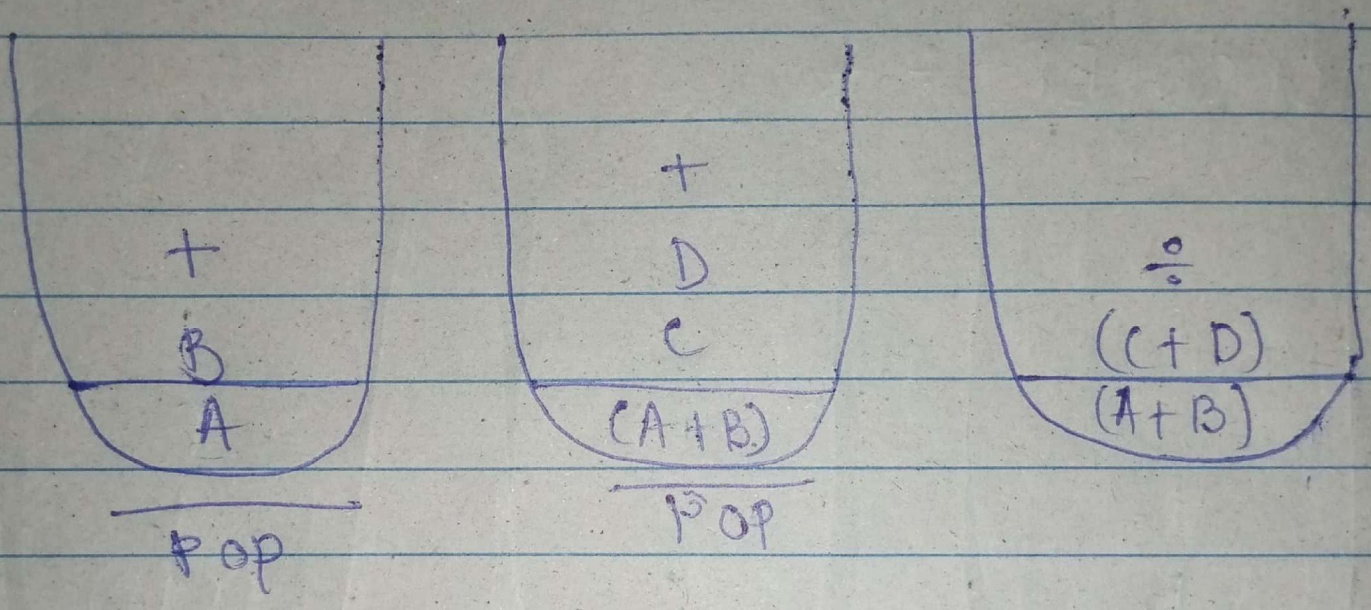
pop

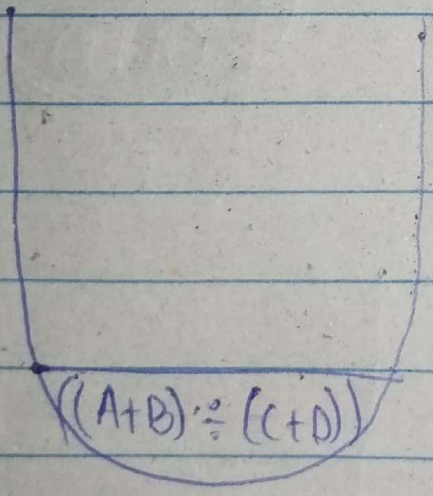
Result

((A+B)*((C+D)÷(E+F)))

example # 2

Postfix : AB+CD+÷



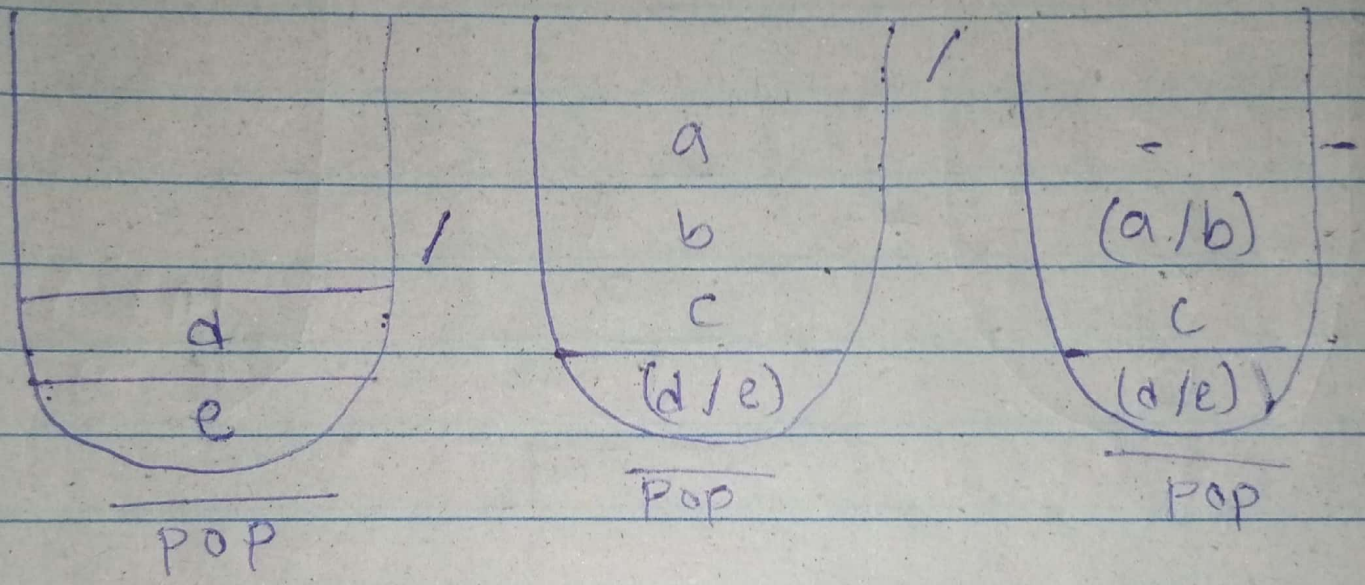


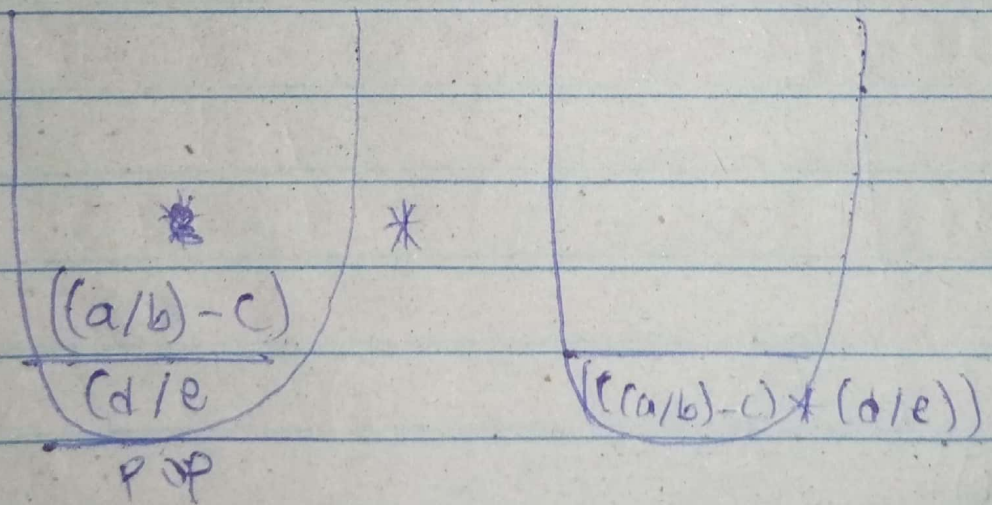
Result:

((A+B)÷(C+D))

TOPIC : Prefix to infix example # 1

Prefix : ~~* - / a b c / d e~~
* - / a b c / d e

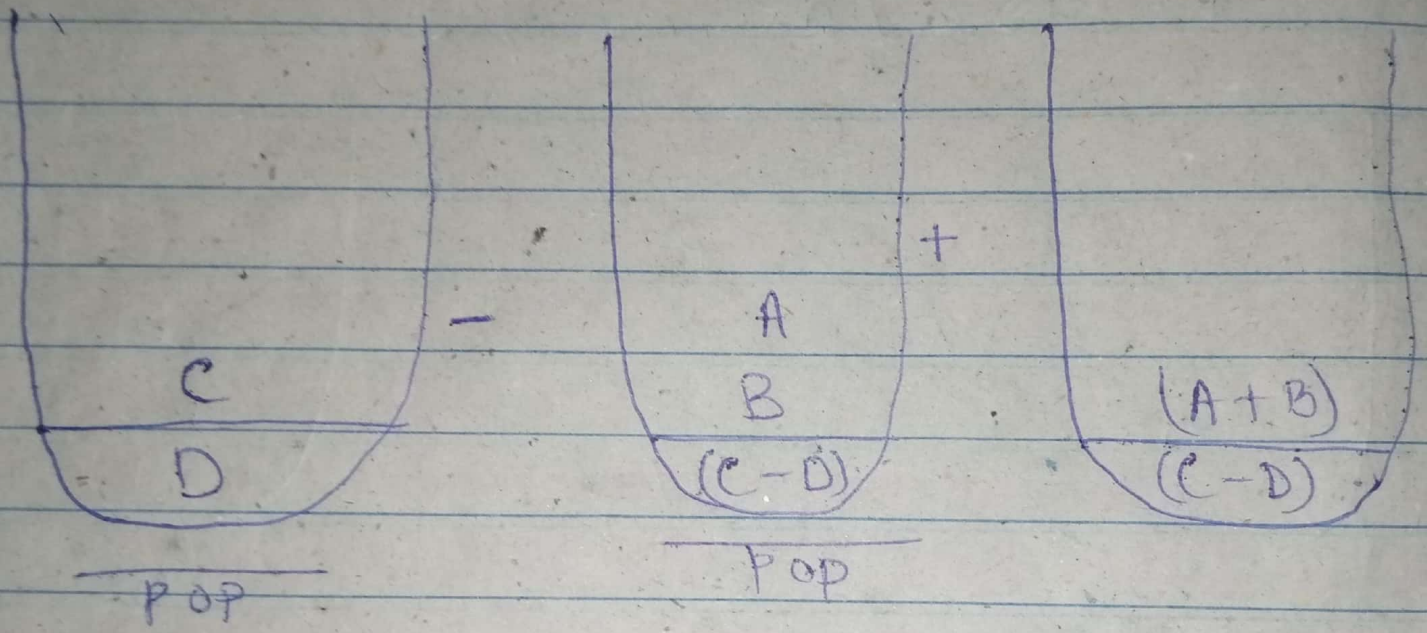


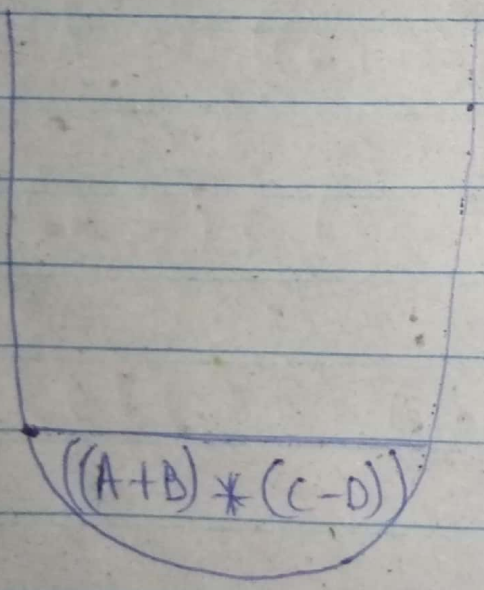


Result
(((a/b) - c) * (d/e))

example #.2

Prefix : * + AB - CD ← start from right





Result :

((A+B)*(C-D))

Topic : Post fix to Prefix example # 1

Post Fix : AB+CD+-

Direct Method for Post fix to Prefix

(A) operator (B)

AB+

(A+B) CD+

(A+B) (C+D)-

((A+B)-(C+D))

Now reverse

((D+C)-(B+A)) Now postfix

ADC+BA+-

Now reverse this

-+AB+CD

Result

-+AB+CD

example # 2

Postfix : AB - CD + EF * / +

Direct Method :

AB -

(A-B) CD+

(A-B) (C+D) EF *

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

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

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

Now reverse

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

Now ;

FE*DC+/BA-+

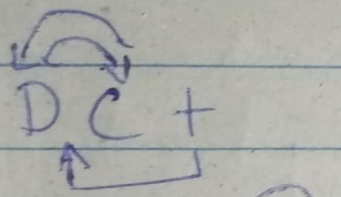
+-AB/+CD*EF

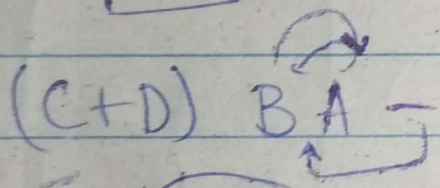
Postfix : +-AB/+CD*EF

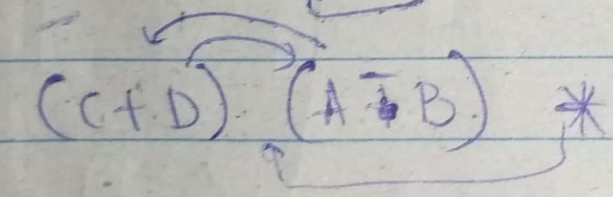
TOPic: Prefix to ~~Infix~~ Postfix example # 1

Prefix: * - AB + CD ←

Direct Method:







((A-B) * (C+D)) → Prefix

Now.

((A - B) * (C + D))

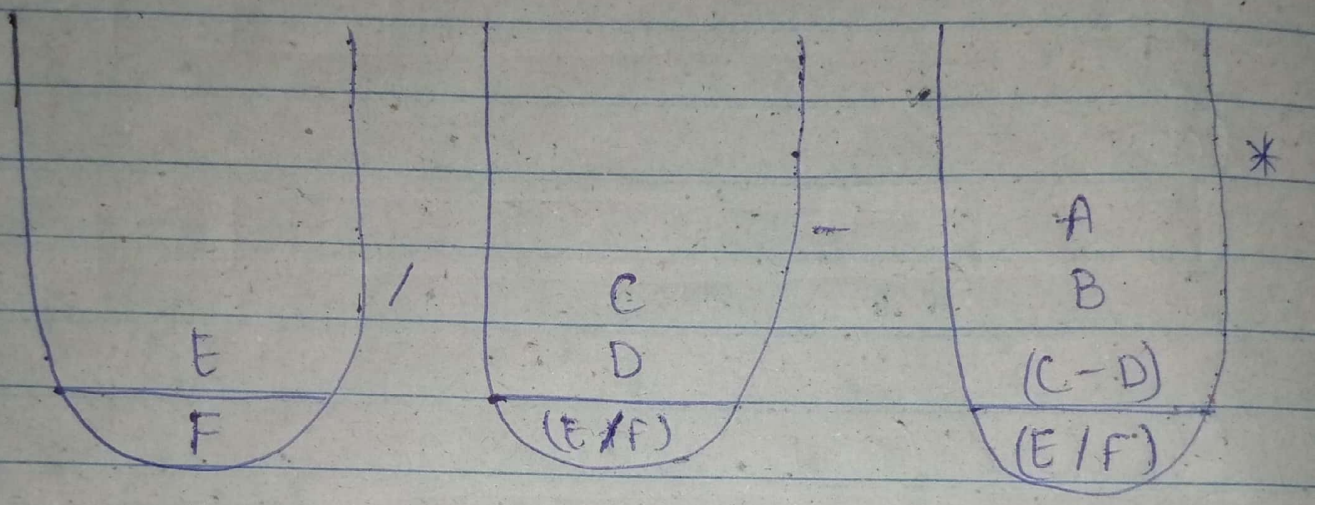
AB - CD + *

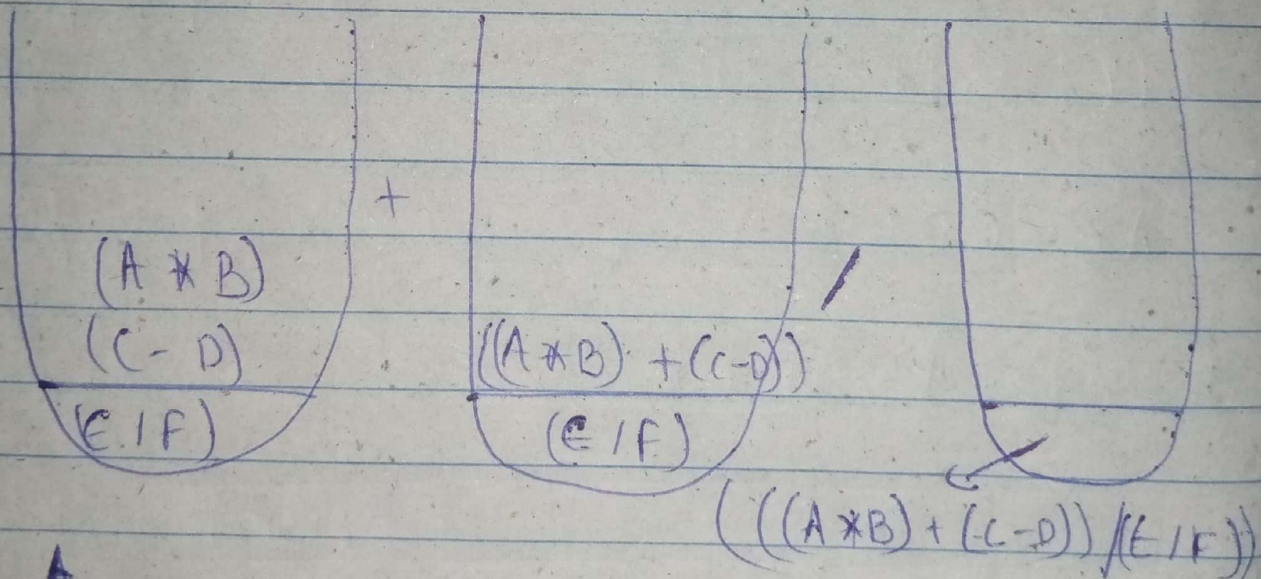
Post Fix: AB - CD + *

example: # 2

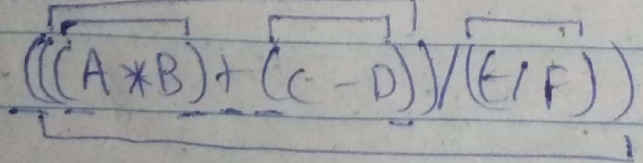
Prefix: /+*AB-CD/EF

Sol:





Now:



AB*CD-+EF// → Post Fix

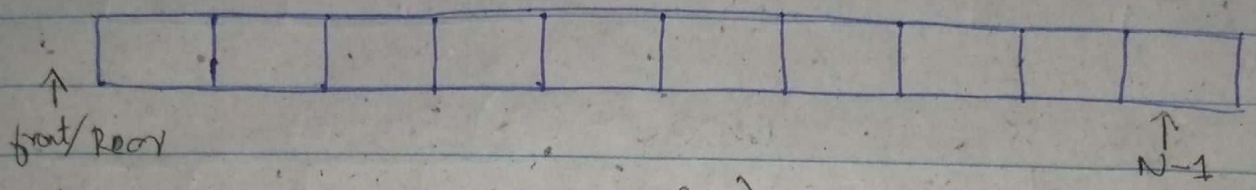
Result: ABx CD-+EF//

TOPIC: Queue

example # 1

Size = 10

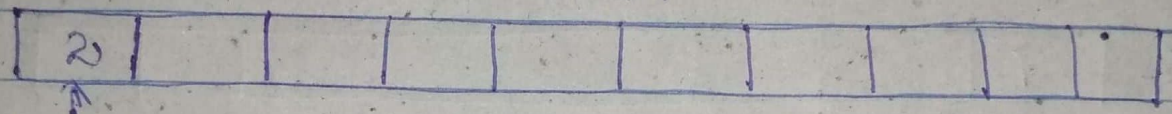
front = rear = -1



enqueue (2)

if (Rear == N-1) X

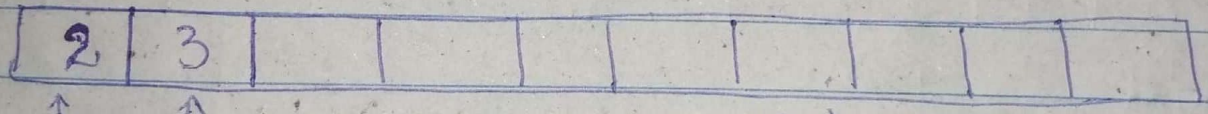
Rear ++



enqueue (3)

isfull = false

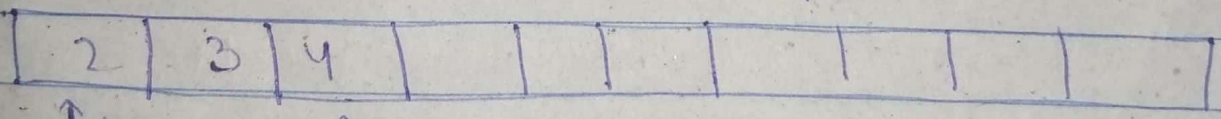
Rear ++



enqueue (4)

isfull = false

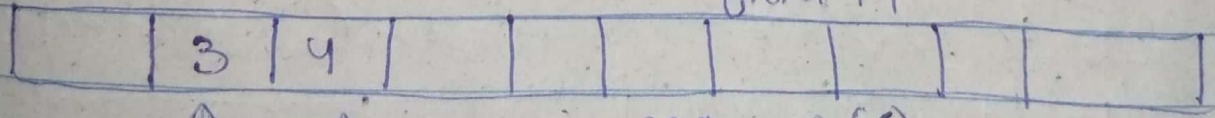
Rear ++



dequeue ()

if is empty = false

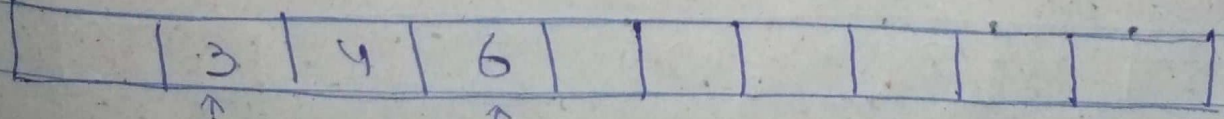
front ++



enqueue (6)

is full = false

Rear = ++



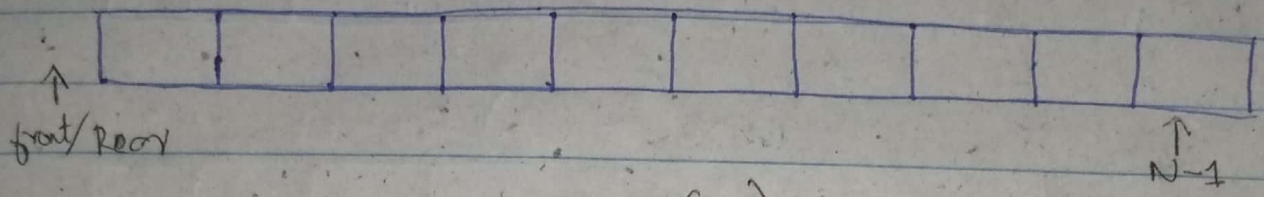
these are basic operations of Queue

Topic: Queue

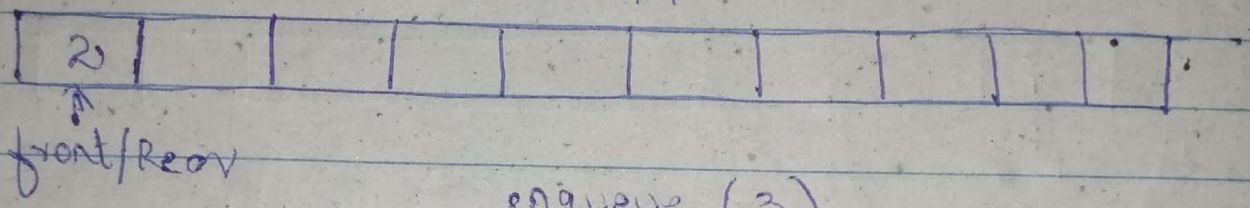
example # 1

Size = 10

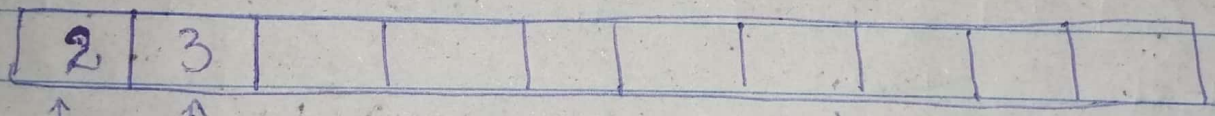
Front = Rear = -1



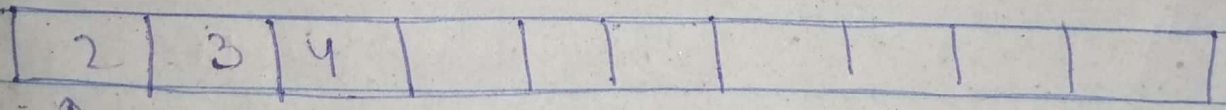
enqueue (2)
if (Rear == N-1) X
Rear ++



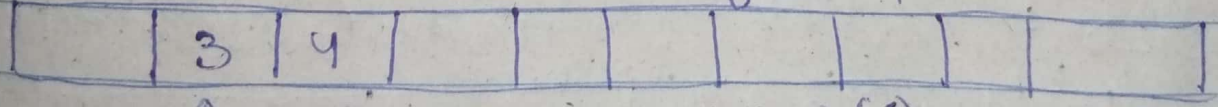
enqueue (3)
is full = false
Rear ++



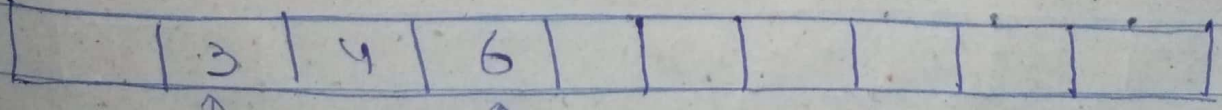
enqueue (4)
is full = false
Rear ++



dequeue ()
is empty = false
front ++

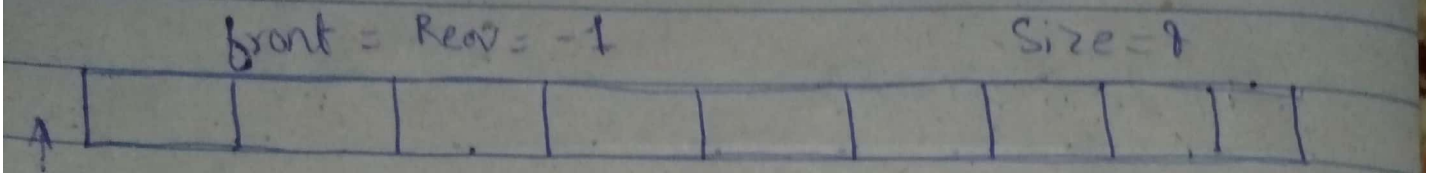


enqueue (6)
is full = false
rear = ++

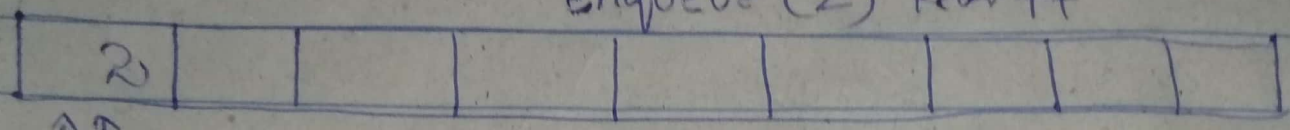


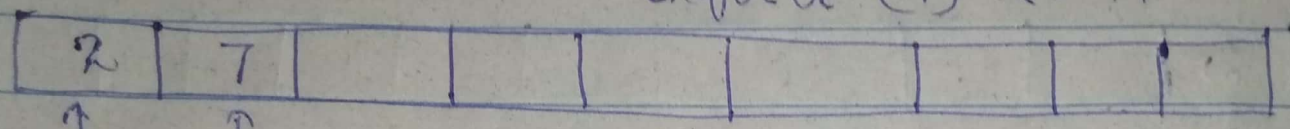
these are basic operations of Queue

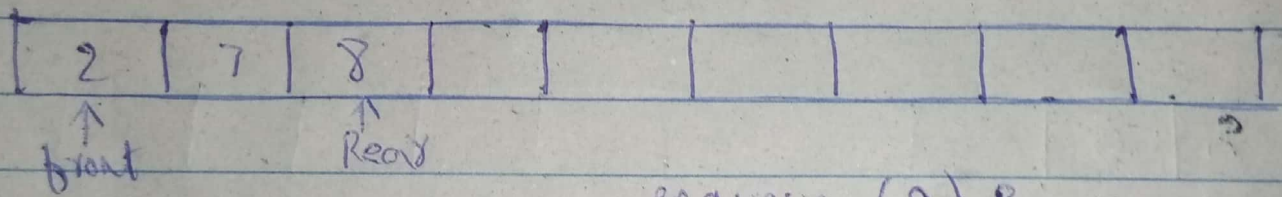
example # 2

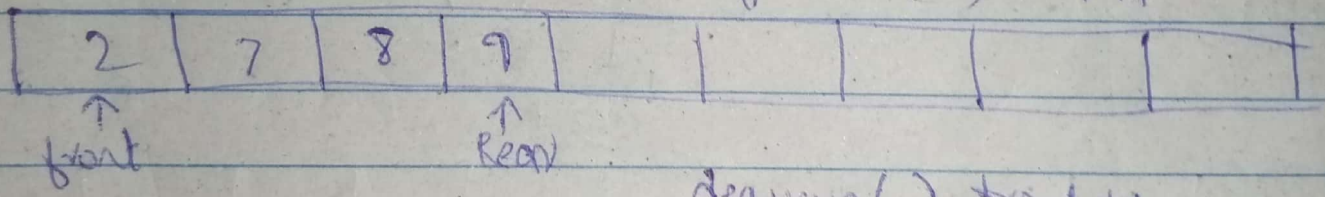


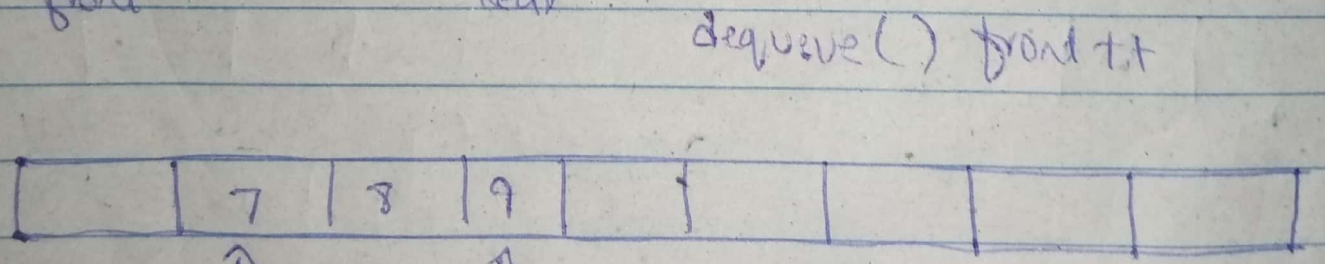
b/r

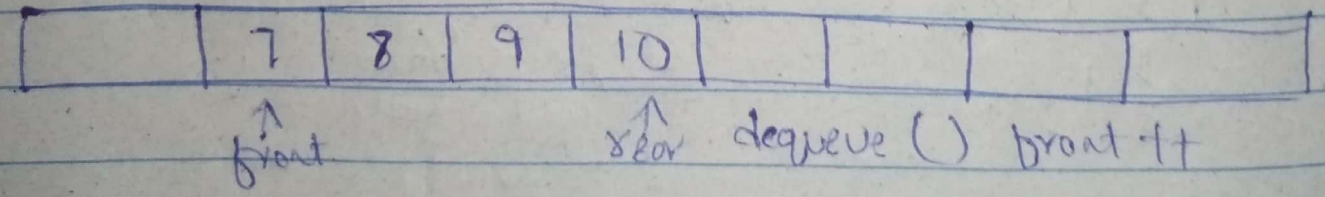


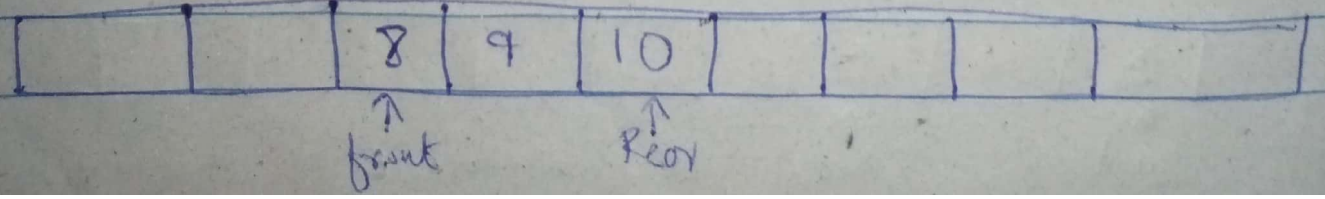






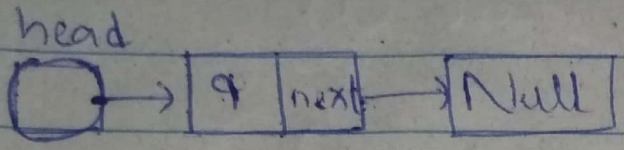






TOPIC : linked list

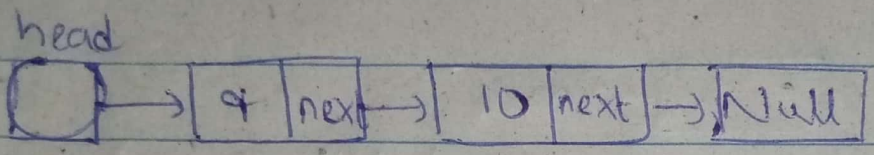
example #1



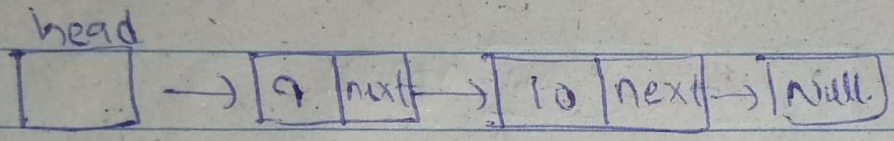
insert (9)

(Root = Null)

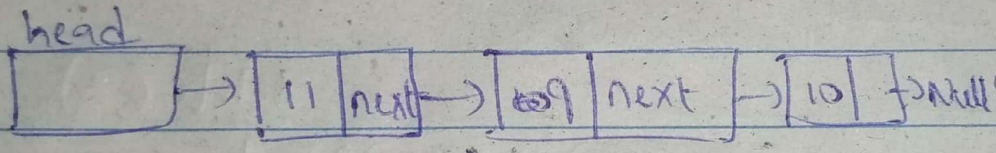
Newnode.next = Null.



insert (10)



insertAtfirst (11)

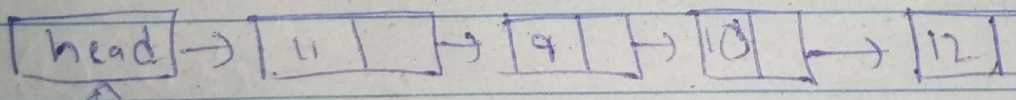


Current = head.

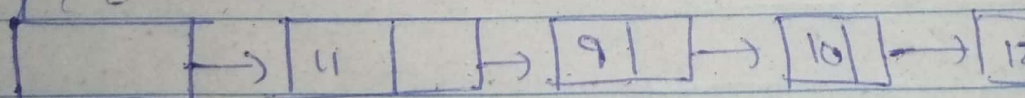
newnode.next = temp

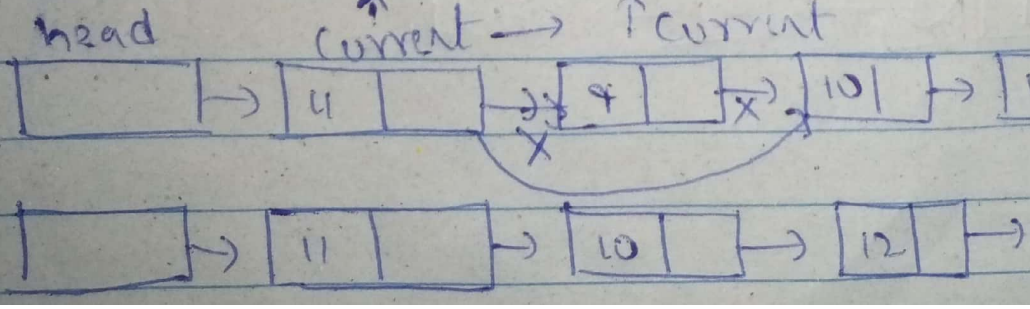
insert (12)

delete (9)

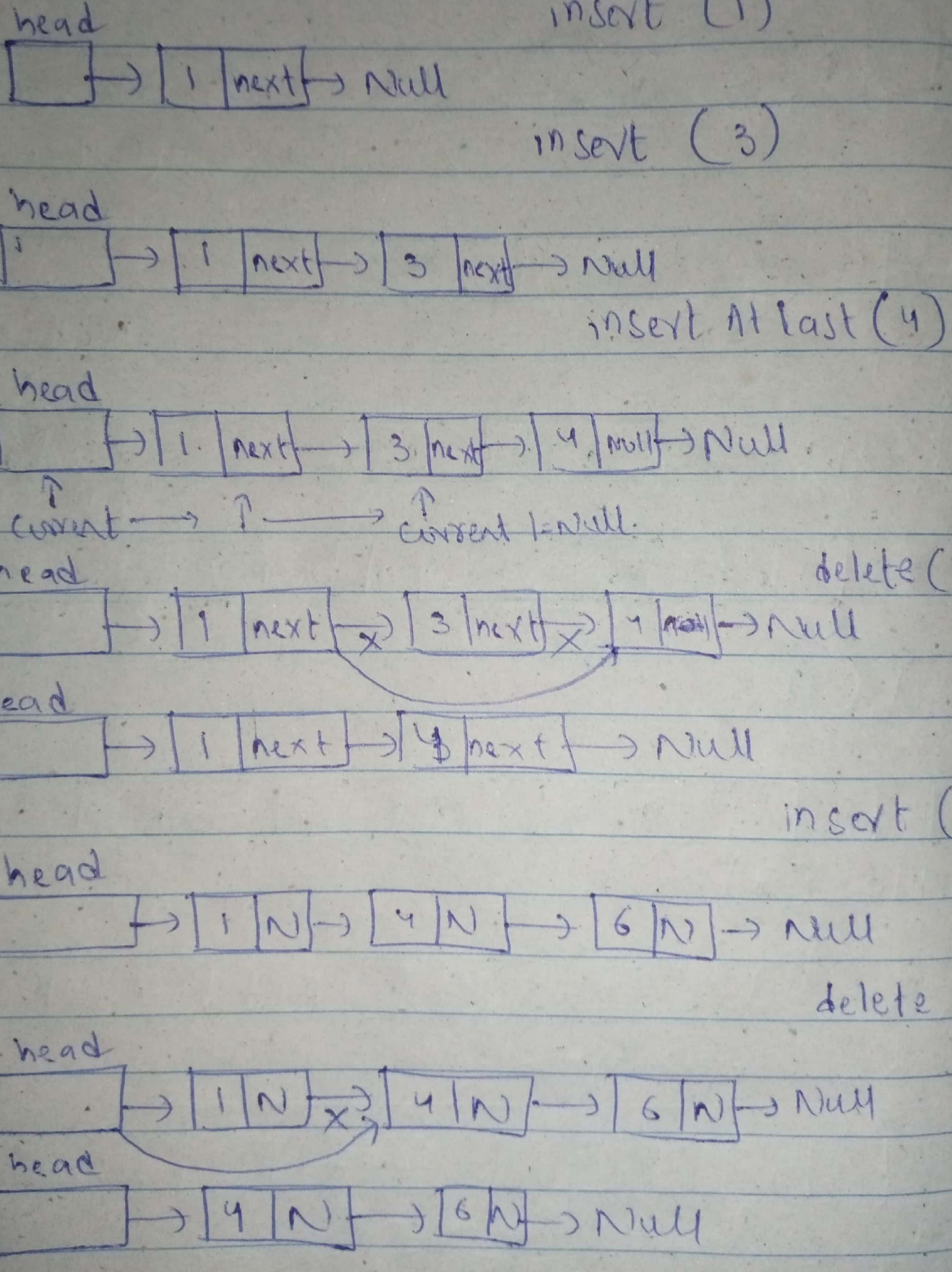


Current (Current.data == key) ✗





example # 2

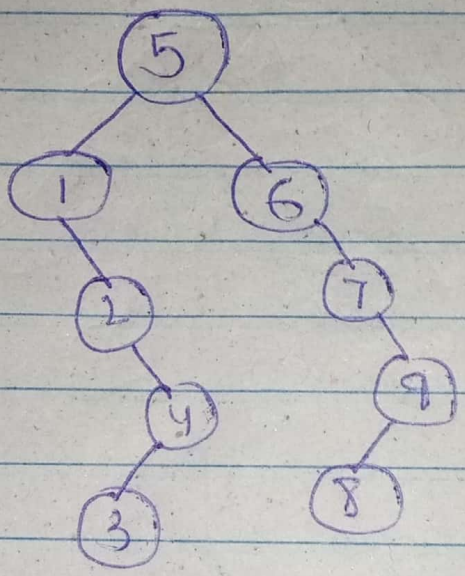


Topic: Binary Search tree

example # 1

insert : 5 1 6 2 4 7 9 8 3

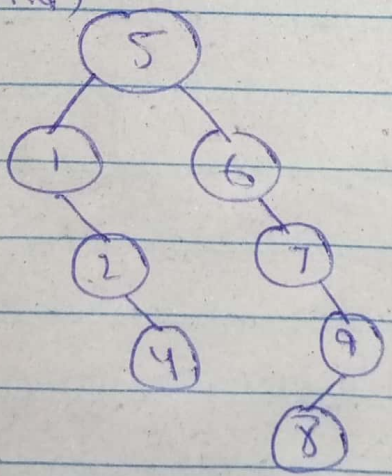
Root



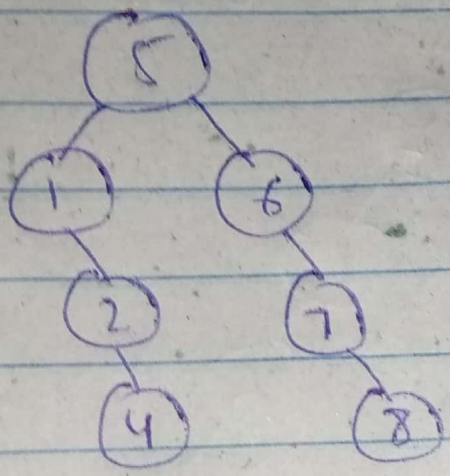
Left (1 < 5) ✓
Right (6 > 5) ✓
Left (2 < 5) ✓
Right (2 > 1) ✓
Left (4 < 5) ✓
Righ (4 > 2) ✓
Right (7 > 5) ✓
Left (7 < 6) ✗
Right (7 > 6) ✓
" (9 > 5) ✓
" (9 > 6) ✓
" (9 > 7) ✓
Right (8 > 5) ✓
Right (8 > 6) ✓
Right (8 > 7) ✓
Left (8 < 9) ✓
Left (3 < 5)
Right (3 > 1)
Right (3 > 2)
left (3 < 4)

Detete : 3, 9, 5

Case 1 (No child)

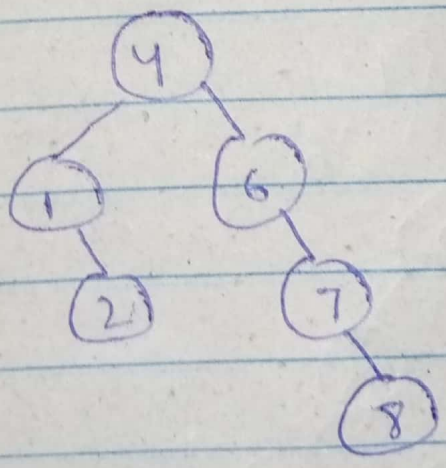


Case: 2 (only 1 child)



Case: 3 : 5 (both child)

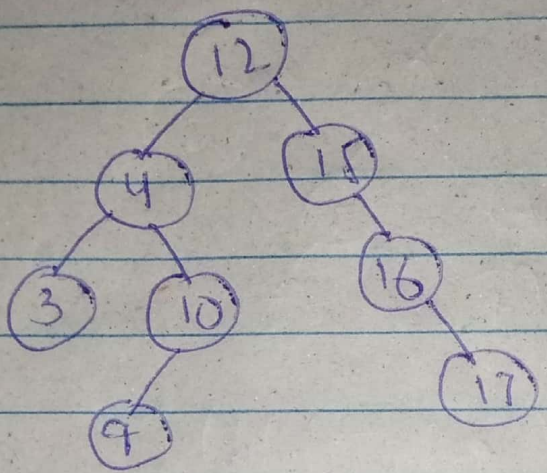
✓ left the maximum or Right the minimum.



Inorder : 1 2 4 6 7 8
Preorder: 4 1 2 6 7 8
PostOrder: 2 1 8 7 6 4

example # 2

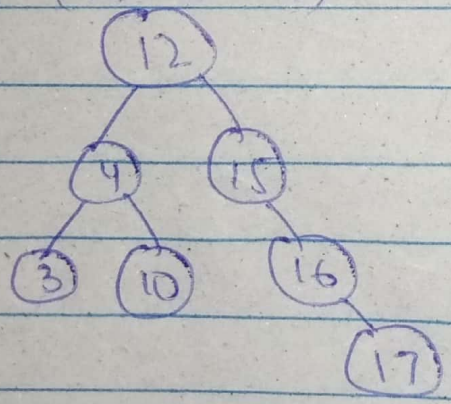
insert: 12 4 10 15 16 3 9 17



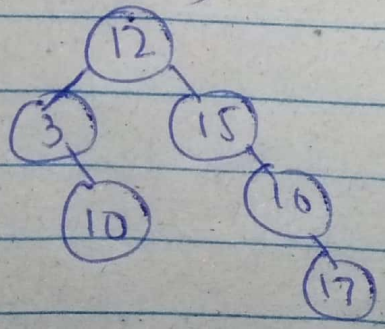
(12) Root
(4 < 12) Left
(10 < 12) "
(10 > 4) Right
(15 > 12) Right
(16 > 12) "
(16 > 15) Right
(3 < 12) Left
(3 < 4) Left
(9 < 12) Left
(9 > 4) Right
(9 < 10) Left
(17 > 12) Right
(17 > 15) Right
(17 > 16) Right

Deletion 9, 4, 17

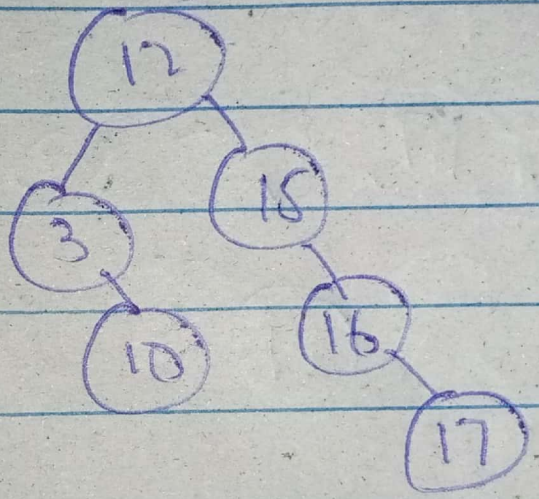
Case 1: 9 (No child)



Case 3: 4 (both child)



Case 1: 17 (No child)



~~Preorder~~

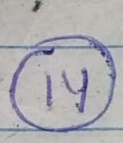
inorder : 3 10 12 15 16 17

Preorder : 12 3 10 15 16 17

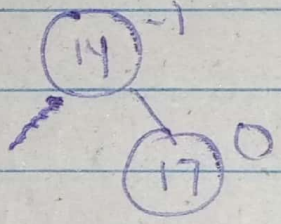
Postorder: 10 3 17 16 15 12

Topic : AVL Tree

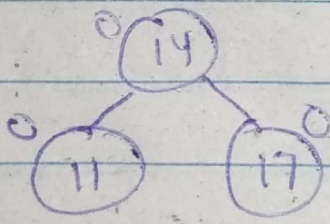
Insert: 14, 17, 11, 7, 53, 4, 13, 12, 8

(14) → 

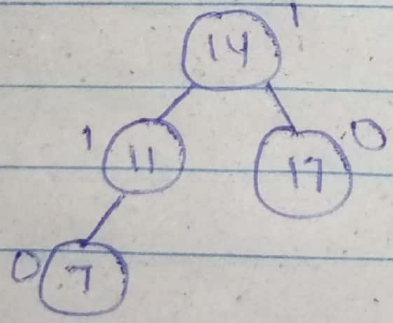
Balance factor = 0
(L - R)

(17) → 

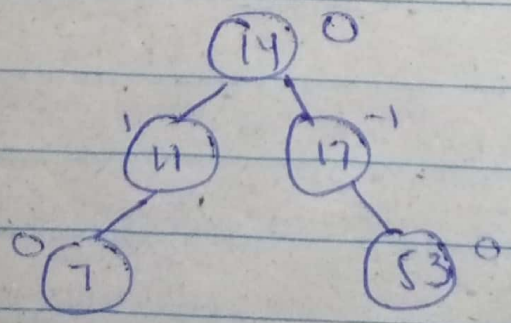
(17 > 14) Right

(11) → 

(11 < 14) left

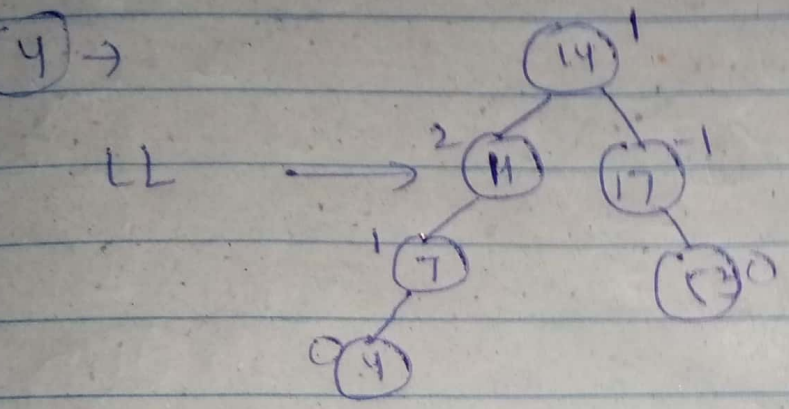
(7) → 

(7 < 14) left
(7 < 11) left

(53) → 

(53 > 14) Right
(53 > 17) Right

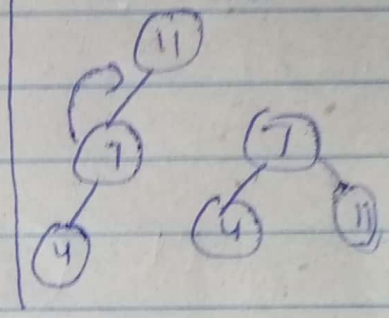
(4) →



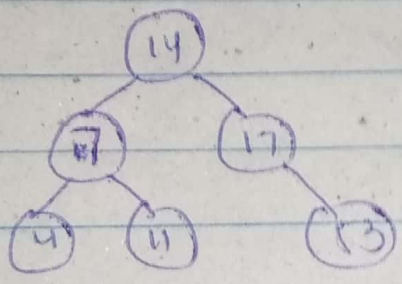
LL

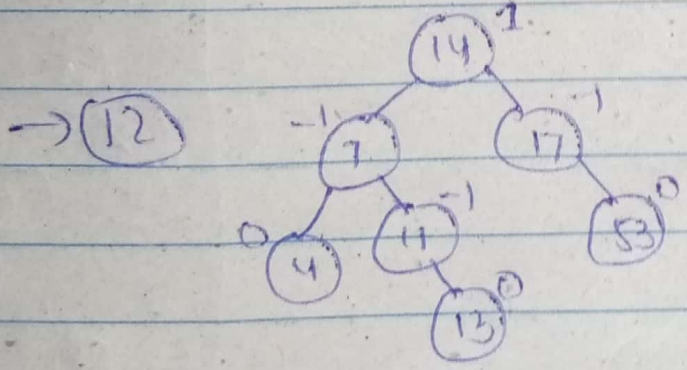
(4 < 14) left
(4 < 11) "
(4 < 7) "

LL Case



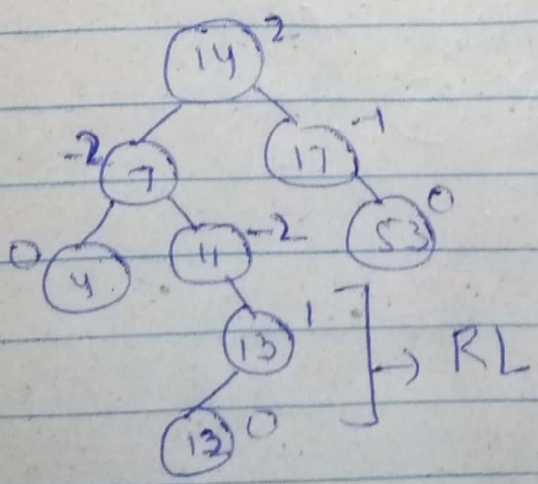
→ (13)





→ (12)

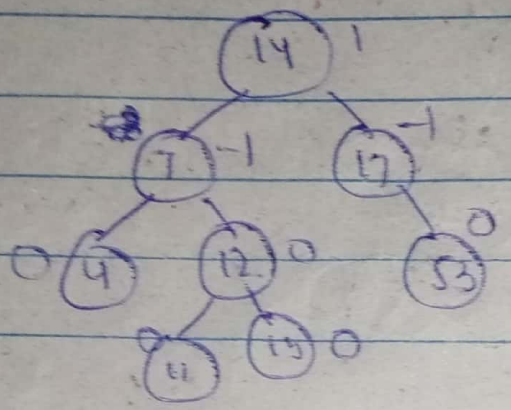
(13 < 14) left
(13 > 7) Right
(13 > 11) Right



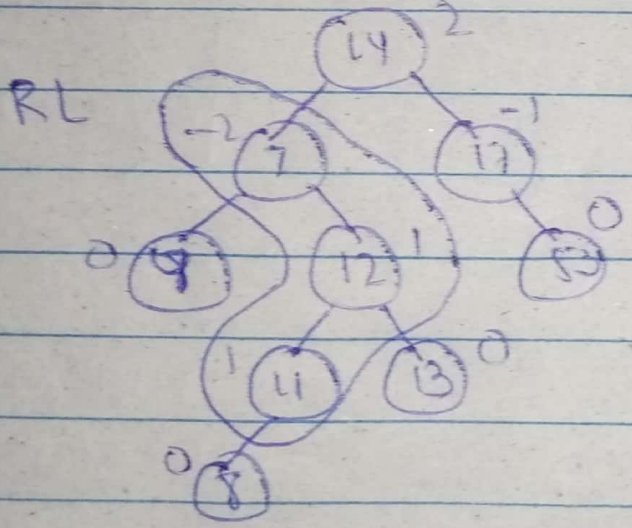
(12 < 14) L
(12 > 7) R
(12 > 11) Right
(12 < 13) left

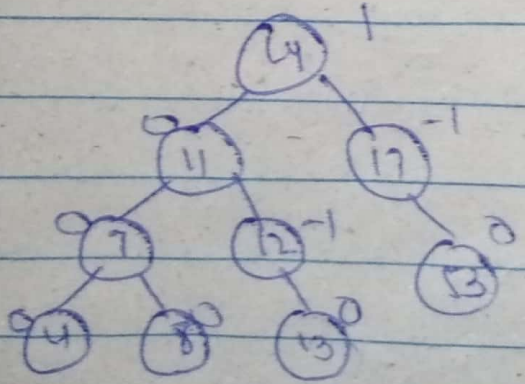
→ RL

5 →

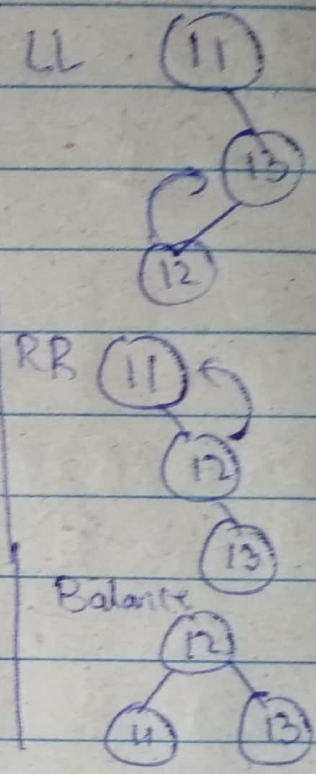


RL

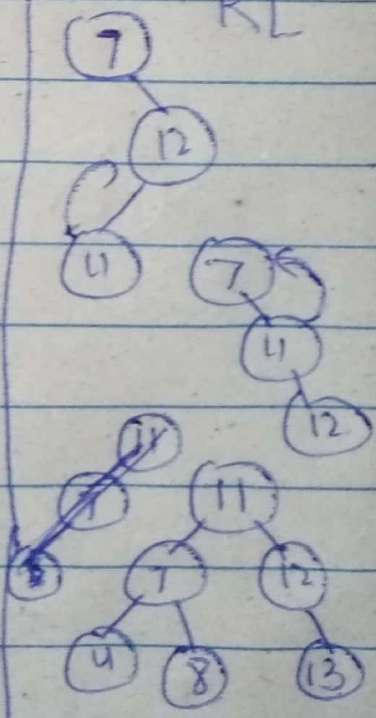




RL

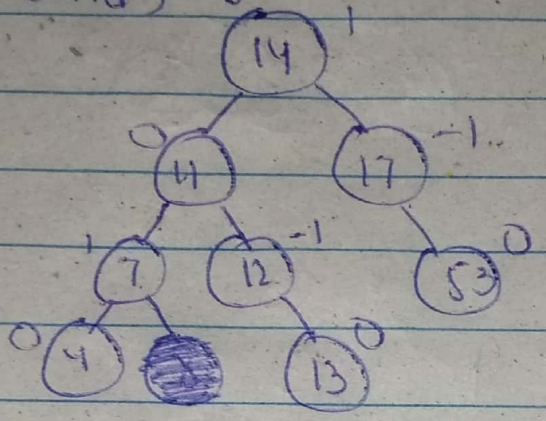


RL

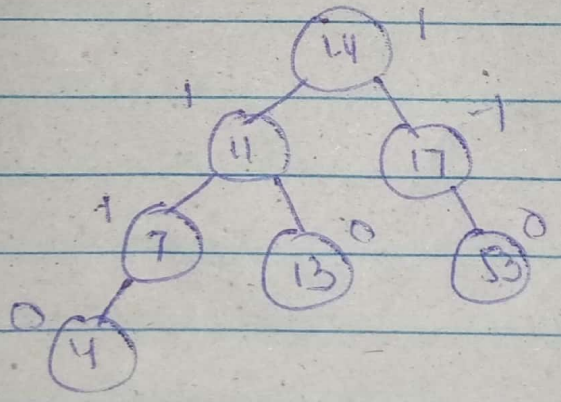


Deleteration: 8, 12, 17

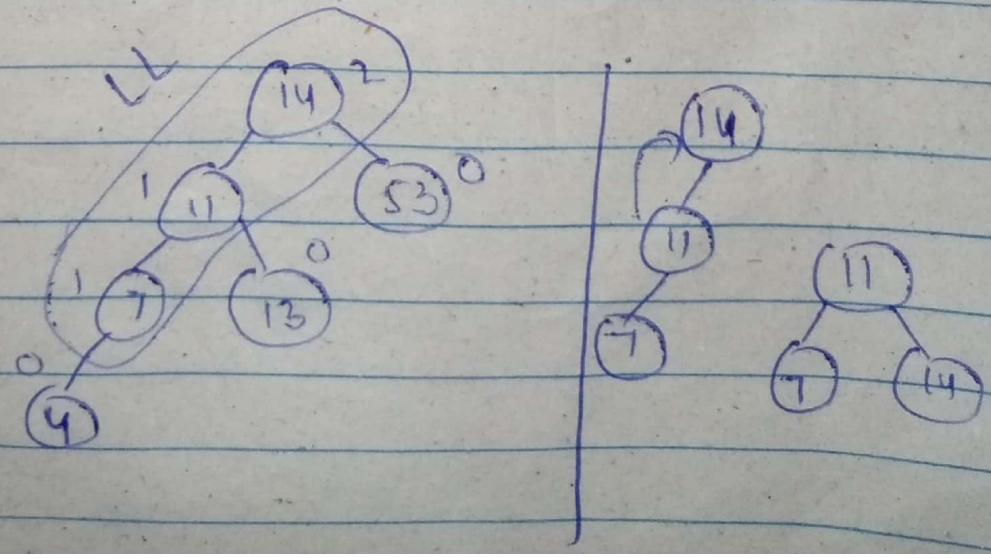
Case1: (no child) 8



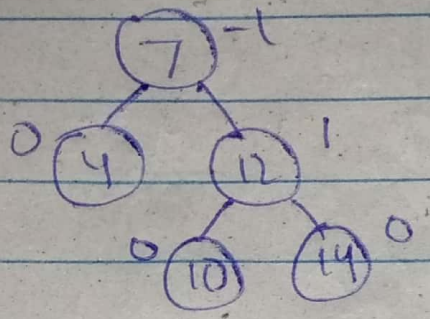
Case 2: (only one child) 12



Case 3: (only one child) 17

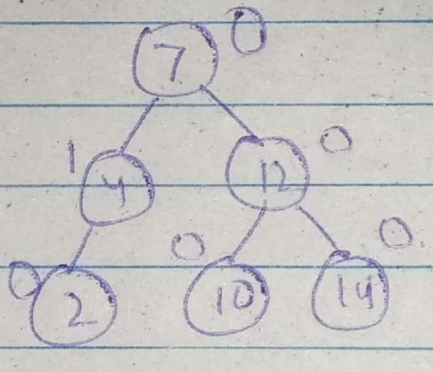


(14) →



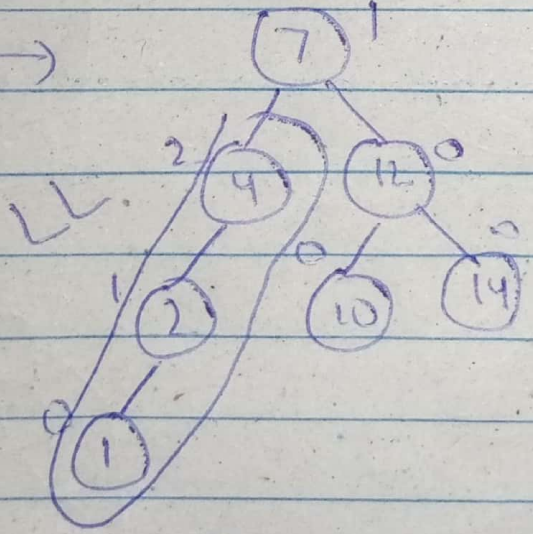
(14 > 7) R
(14 > 12) B

(2) →

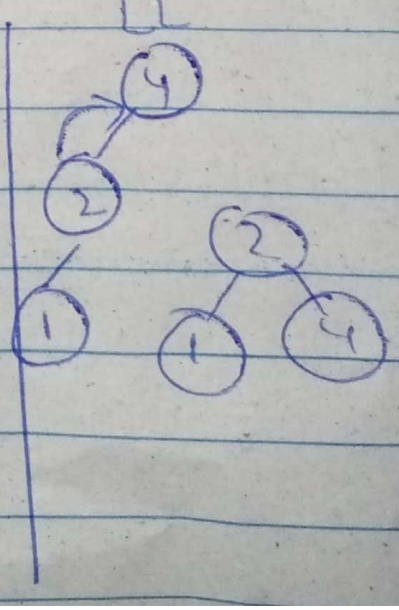


(2 < 7) L
(2 < 4) L

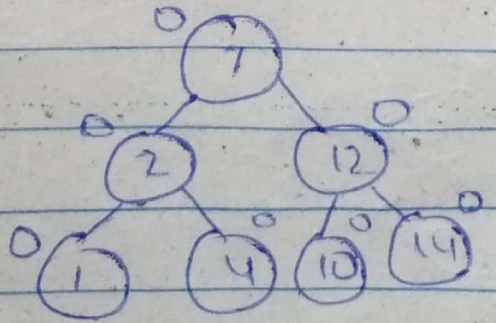
(1) →



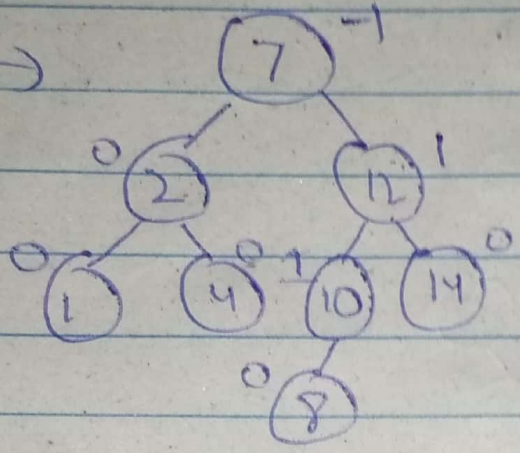
(1 < 7) L
(1 < 4) L
(1 < 2) L



(3) →



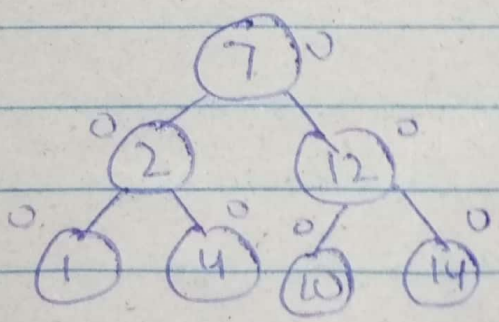
8 →



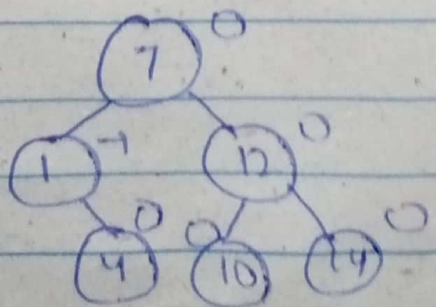
(8 > 7) R
(8 < 12) L
(8 < 10) L

Deletion: 8, 2, 14.

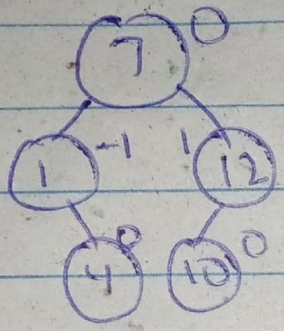
Case 1: 8 (~~only~~ no child)



Case 3: 2, (both child)



Case 1 : (No child) 14



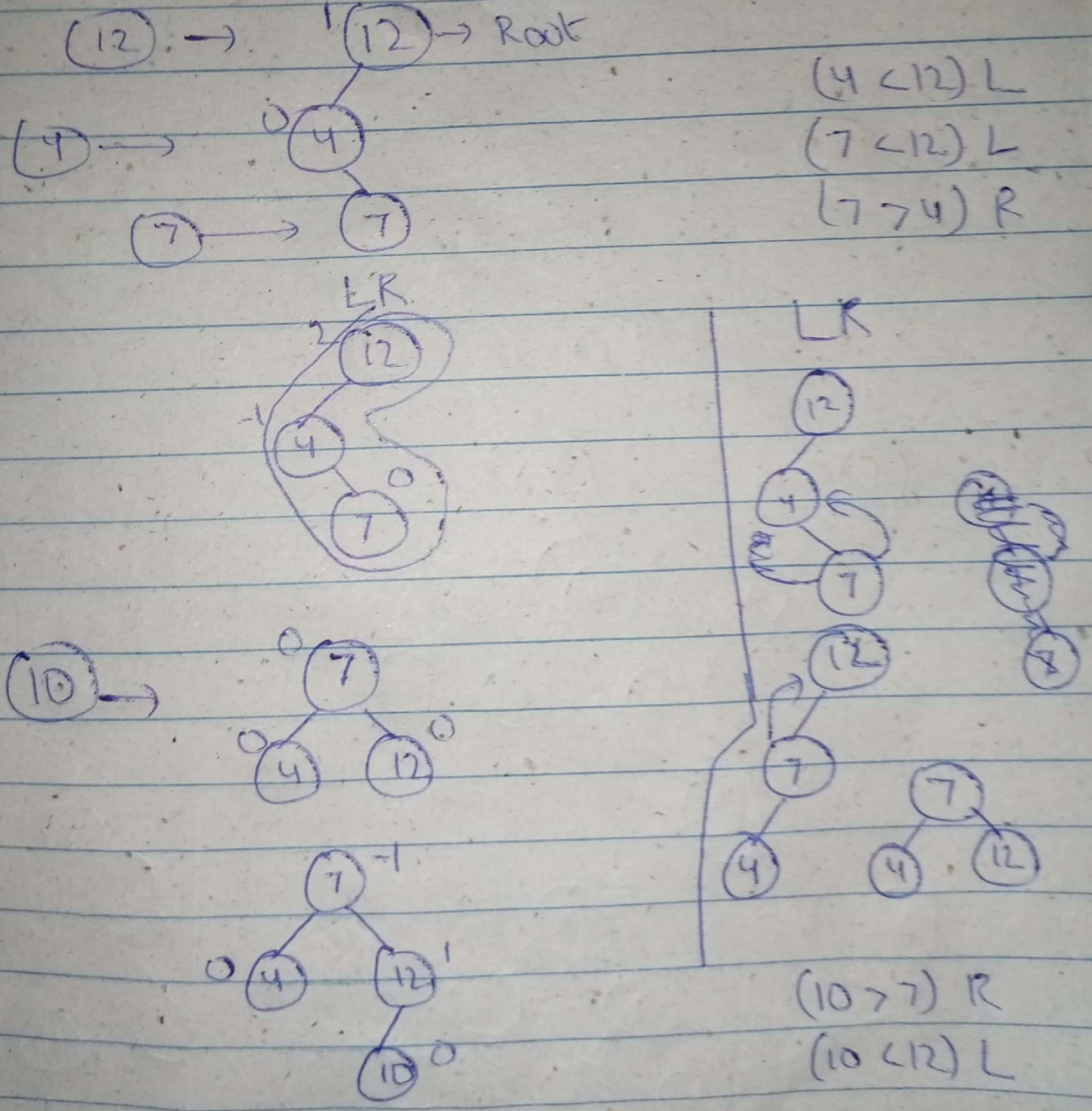
inorder : 1 4 7 10 12

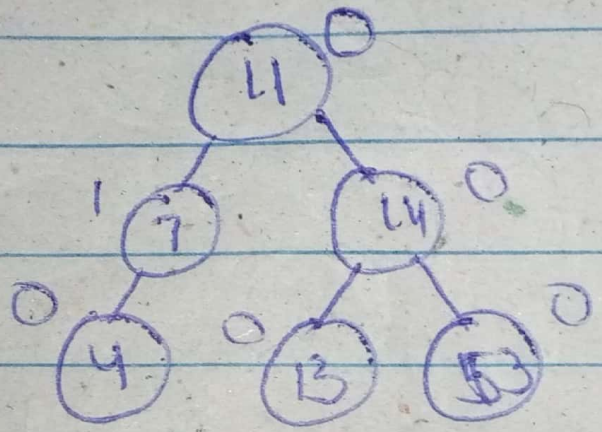
preorder : 7 1 4 12 10

post order : 1 4 1 10 12 7

example # 2)

insert : 12, 4, 7, 10, 14, 2, 1, 8





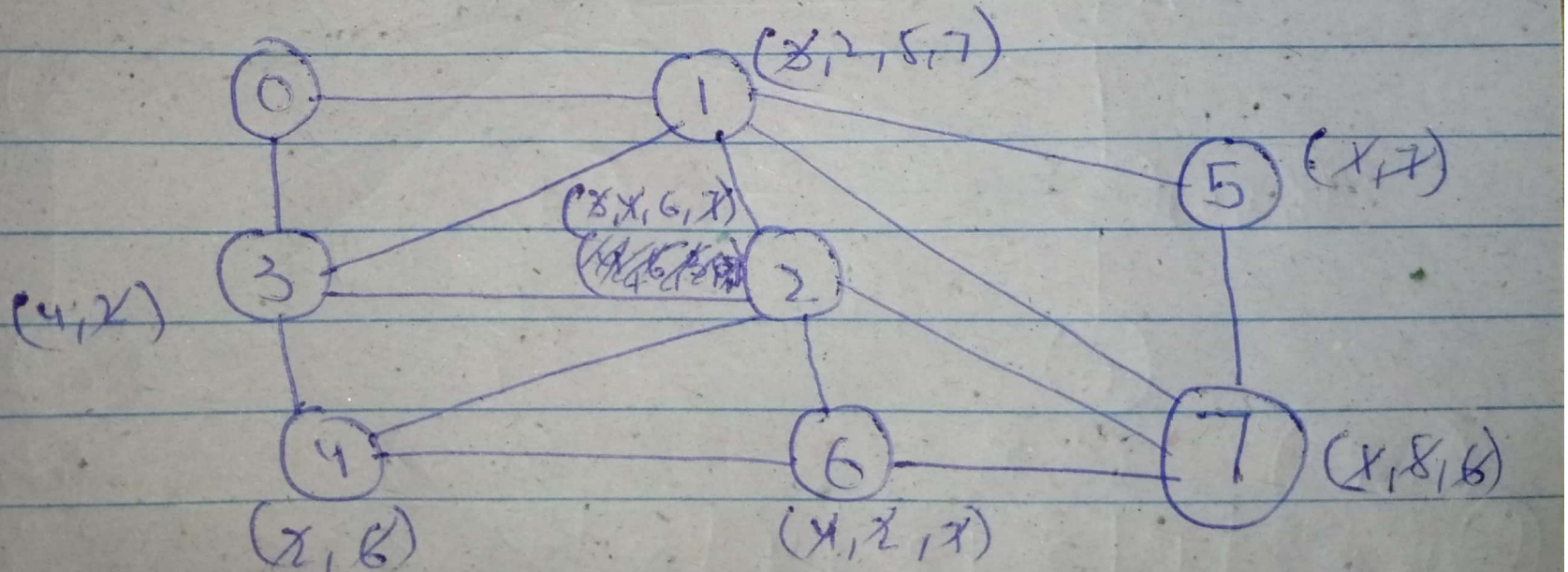
Inorder : 4 7 11 13 14 53

Preorder : 11 7 4 14 13 53

Postorder : 4 7 13 53 14 11

TOPIC : Graphs

example# 1



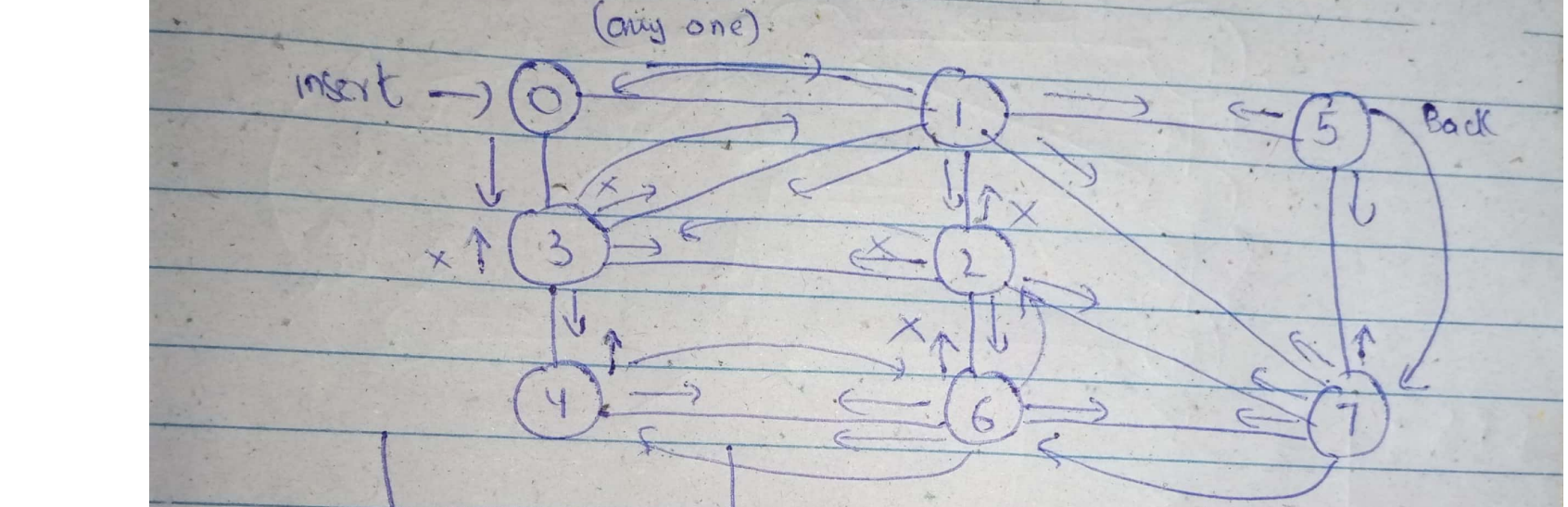
BFS : | 0 | 1 | 3 | 2 | 5 | 7 | 4 | 6 |

Result : 0 1 3 2 5 7 4 6

(1, 6) (1, 2, 7) (1, 5, 6)

BFS : | 0 | 1 | 3 | 2 | 5 | 7 | 4 | 6 |

Result : 0 1 3 2 5 7 4 6

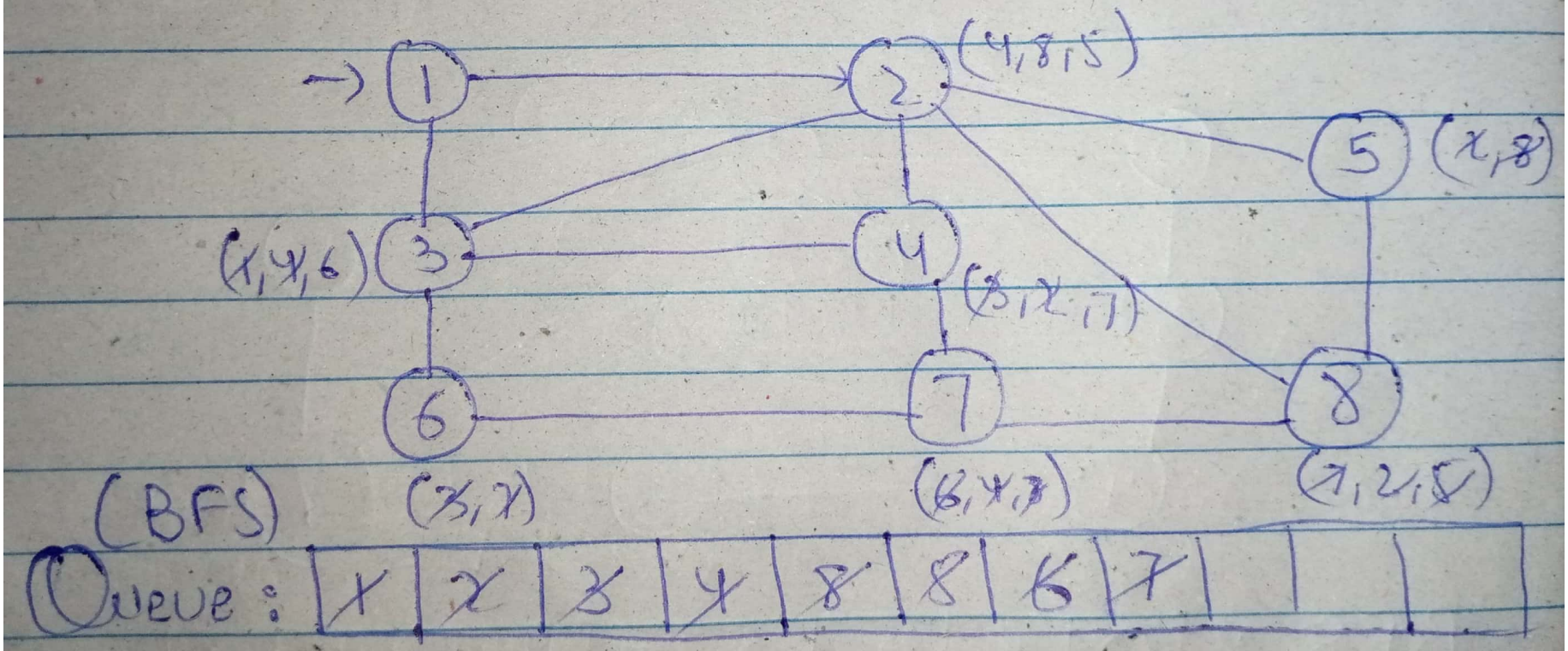


DFS :

0
1
3
2
5
7
4
6

Result : 0 1 3 2 6 7 5 4

example # 2

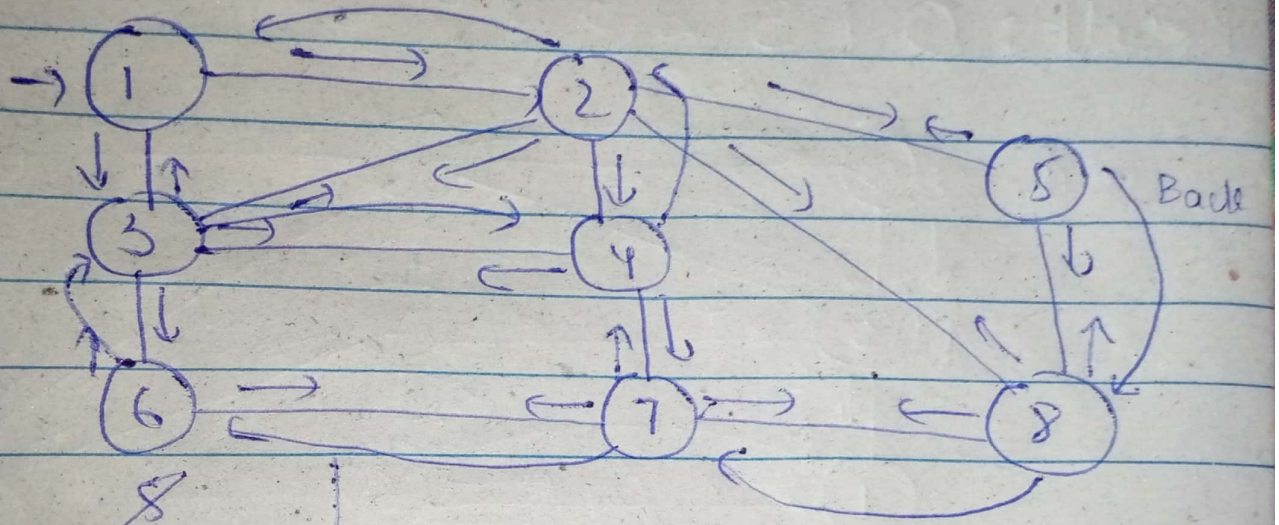


(BFS)

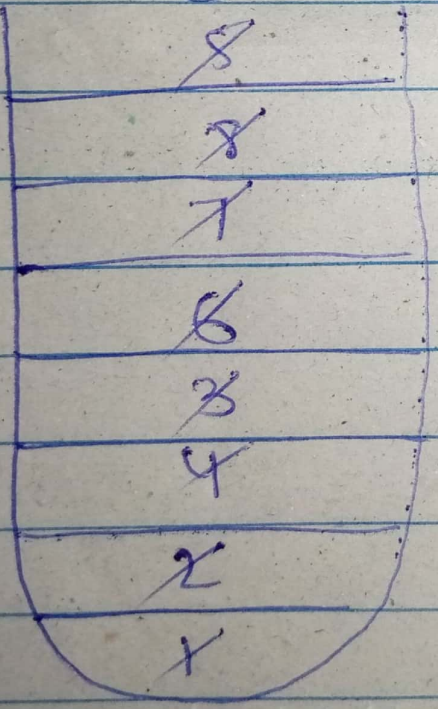
Queue : | 1 | 2 | 3 | 4 | 8 | 8 | 6 | 7 | | | |

Result : 1 2 3 4 8 5 6 7

Result : 1 2 3 4 8 5 6 7



DFS :



Result : 1 2 4 3 6 7 8 5