FALL – SEMESTER **Name: Nidhi Singh**
Course Code: MCSE501L **Reg. No:22MAI0015**
Course-Title: – Data Structures and Algorithms
DIGITAL ASSIGNMENT - I

Faculty: SARAVANAN R - SCOPE

1. Construct a binary search tree from the letters of your name (as in passport) and then delete all vowels present in it. Show both trees (after construction, after deletion). Write all steps. Keep the duplicates (repetition of letters) at the left side (left sub tree)

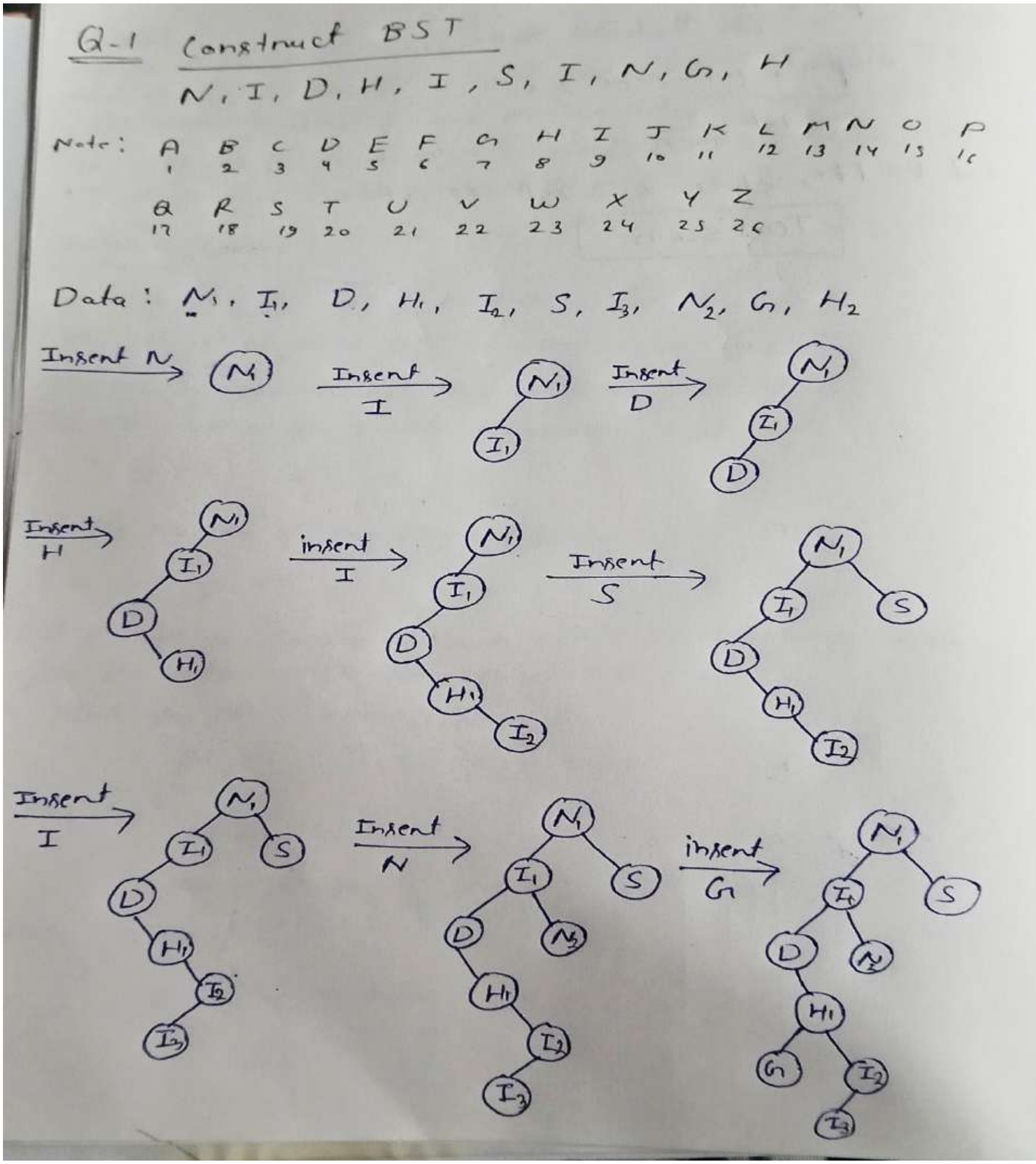
2. Using in-order, pre-order and post-order traversals, construct Threaded binary search trees from the letters of: name1||name2, where name1 is your father's name and name2 is your mother's name, || is string concatenation. Write all steps. Keep the duplicates (repetition of letters) at the right side (right sub tree)

3. Construct a red black tree from the letters of: name1:name2, where name1 is your name (as in passport) and name2 is name of your home town, || is string concatenation. Write all steps. Keep the duplicates (repetition of letters) at the right side (right sub tree)

4. Construct a splay tree from the following numbers a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13 where a1, a2, a3, a4, a5 are first two digits, 2nd two digits, …. Last two digits of your mobile number respectively, a6, a7, a8, a9, a10 are first two digits, 2nd two digits, …. Last two digits of your father's mobile number respectively, and a11, a12, a13 are first two digits, 2nd two digits and Last two digits of your hometown's pin code respectively. Write all steps. Keep the duplicates (repetition of numbers) at the left side (left sub tree)

5. Sort digits of your mobile number using radix sort, heap sort, insertion, selection sort

1. Construct a binary search tree from the letters of your name (as in passport) and then delete all vowels present in it. Show both trees (after construction, after deletion). Write all steps. Keep the duplicates (repetition of letters) at the left side (left sub tree)



Q-1 Construct BST

N, I, D, H, I, S, I, N, G, H

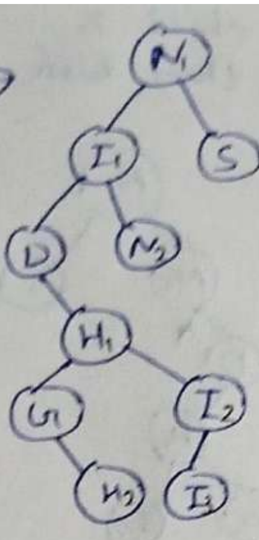
Note:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Q	R	S	T	U	V	W	X	Y	Z
17	18	19	20	21	22	23	24	25	26

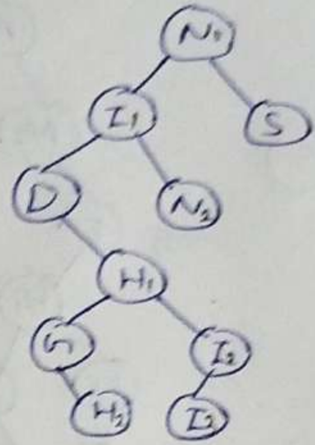
Data: N_1, I_1, D, H_1, I_2, S, I_3, N_2, G, H_2

Insert →
H



⇒

final Tree

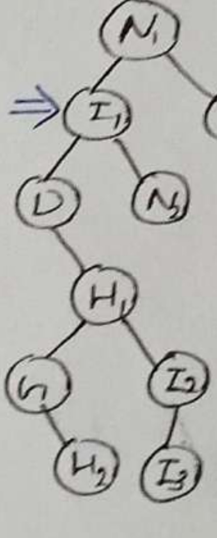


Deletion

Delete all vowels (A, E, I, O, U)

Delete order ⇒ I_1, I_2, I_3

Delete I_1



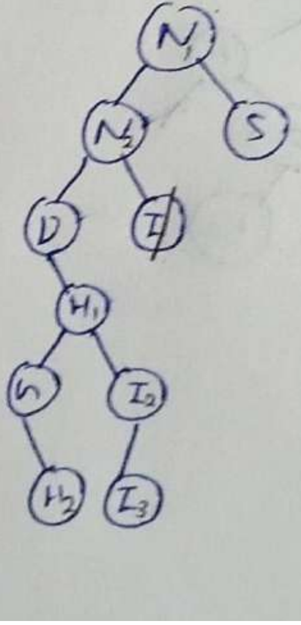
⇒ Node I has both child

⇒ Inorder successor of I is N

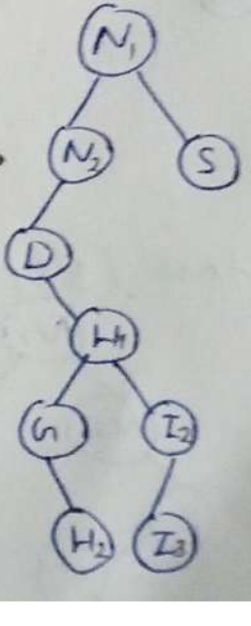
⇒ Replace I with N

⇒ then ~~delet~~ Remove I

⇒

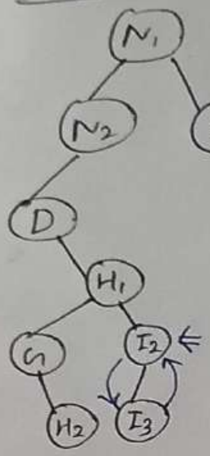


→

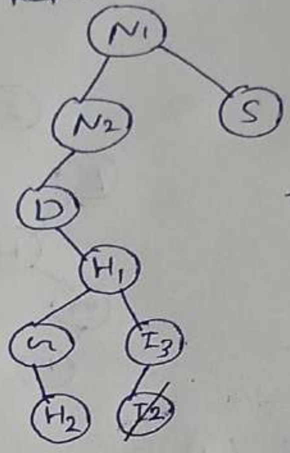


Delete I_2

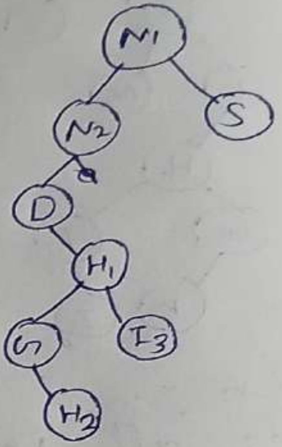
I_2 has only one child I_3
Rep Swap (I_2, I_3) then delete I_2



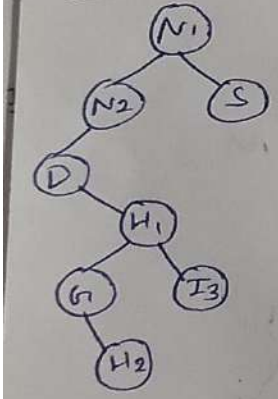
⇒



⇒

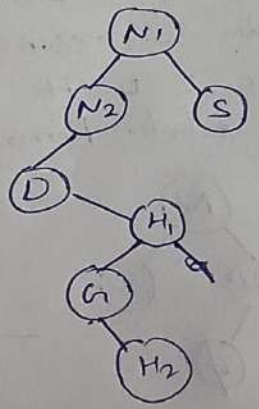


Delete I_3

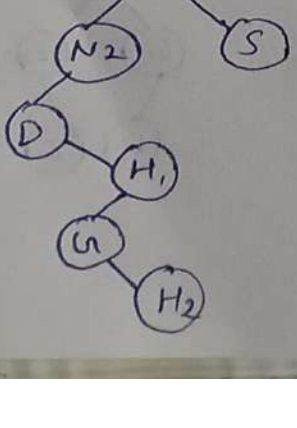


I_3 → leaf Node
Replace I_3 with NULL

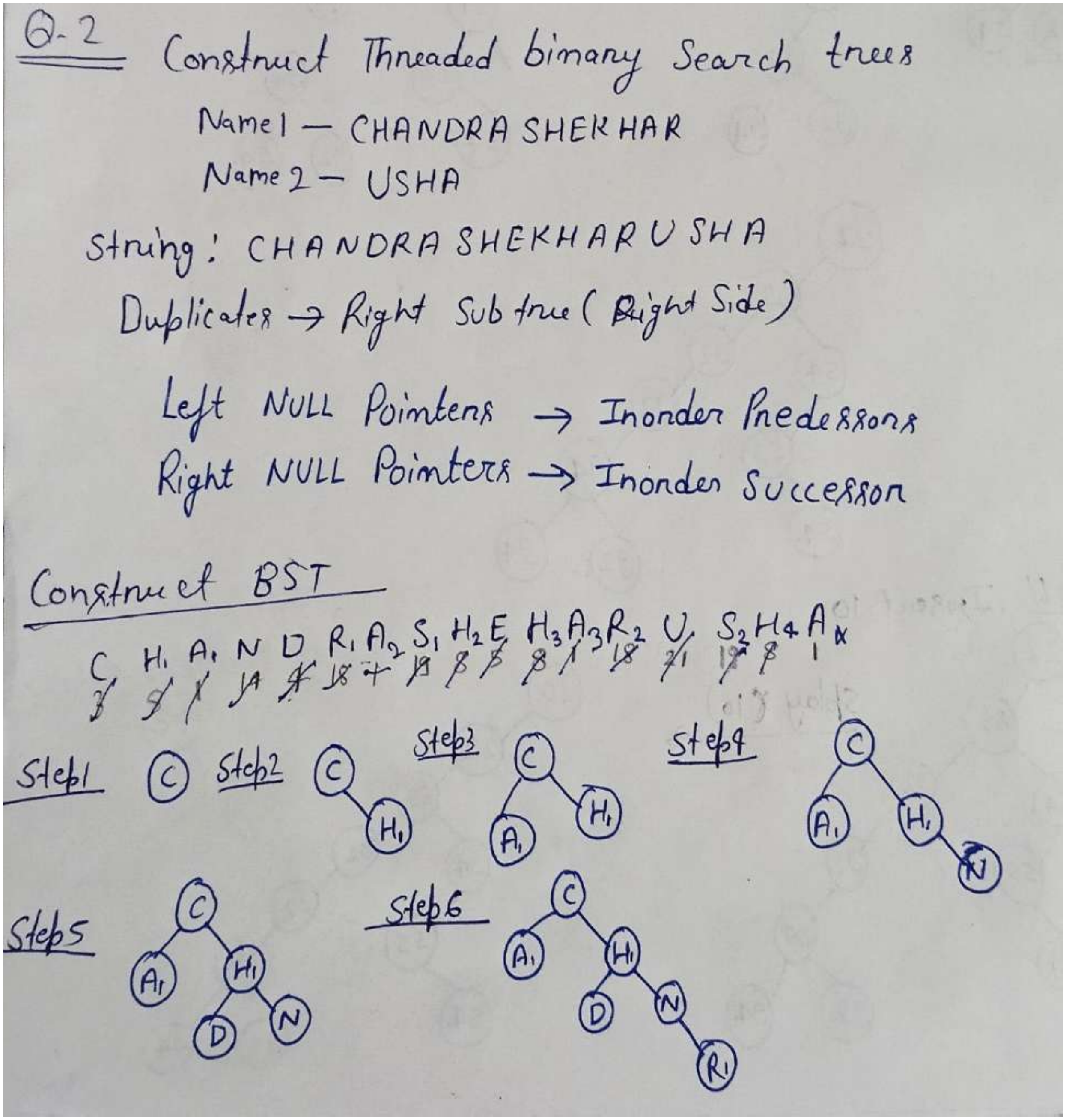
⇒



Final tree (BST)



2. Using in-order, pre-order and post-order traversals, construct Threaded binary search trees from the letters of: name1‖name2, where name1 is your father's name and name2 is your mother's name, || is string concatenation. Write all steps. Keep the duplicates (repetition of letters) at the right side (right sub tree).



Q-2 Construct Threaded binary Search trees

Name1 — CHANDRA SHEKHAR

Name 2 — USHA

String : CHANDRASHEKHARUSHA

Duplicates → Right Sub tree (Right Side)

Left NULL Pointers → Inorder Predessors

Right NULL Pointers → Inorder Successor

Construct BST

C H_1 A_1 N D R_1 A_2 S_1 H_2 E H_3 A_3 R_2 U S_2 H_4 A_4

~~3~~ ~~8~~ ~~1~~ ~~14~~ ~~4~~ ~~18~~ ~~1~~ ~~19~~ ~~8~~ ~~5~~ ~~8~~ ~~1~~ ~~18~~ ~~21~~ ~~19~~ ~~8~~ 1

Step1

Step2

Step3

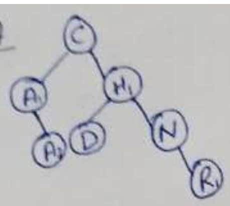
Step4

Step5

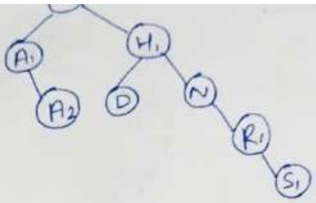
Step6

C
C
H1
C
A1
H1
C
A1
H1
N
C
A1
H1
D
N
C
A1
H1
D
N
R1

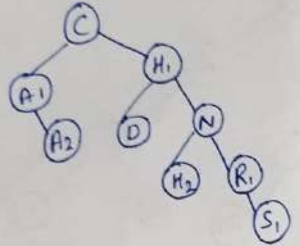
step7



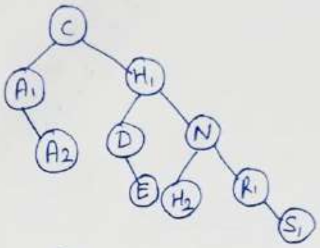
Step8



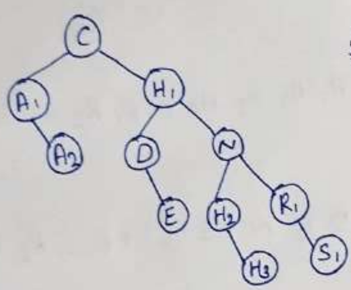
Step9



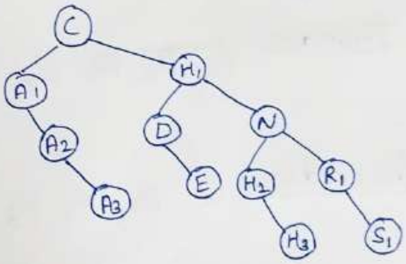
Step10



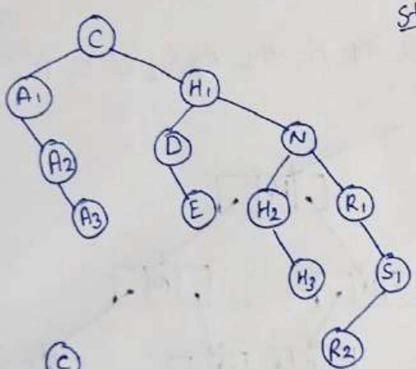
Step11



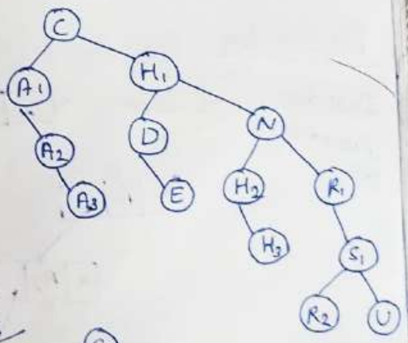
step 12



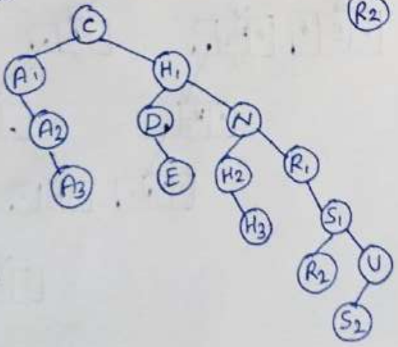
Step13



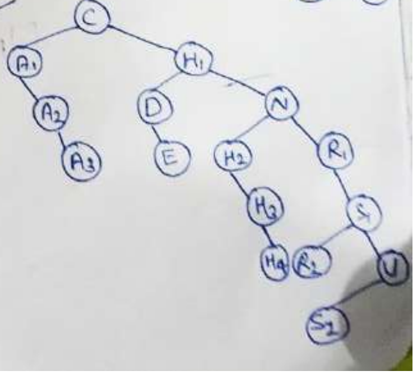
Step 14

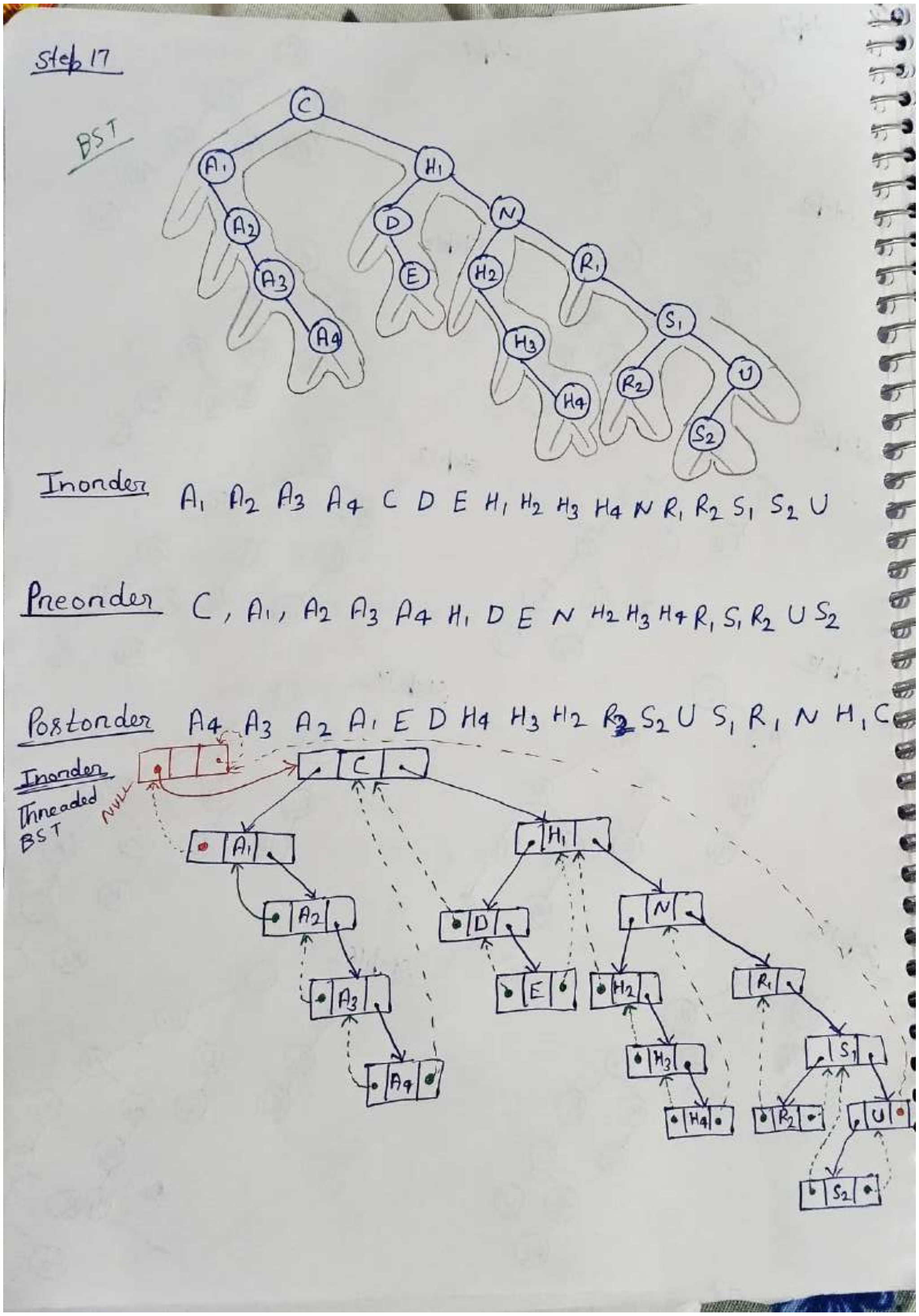


Step15



Step16





Step 17

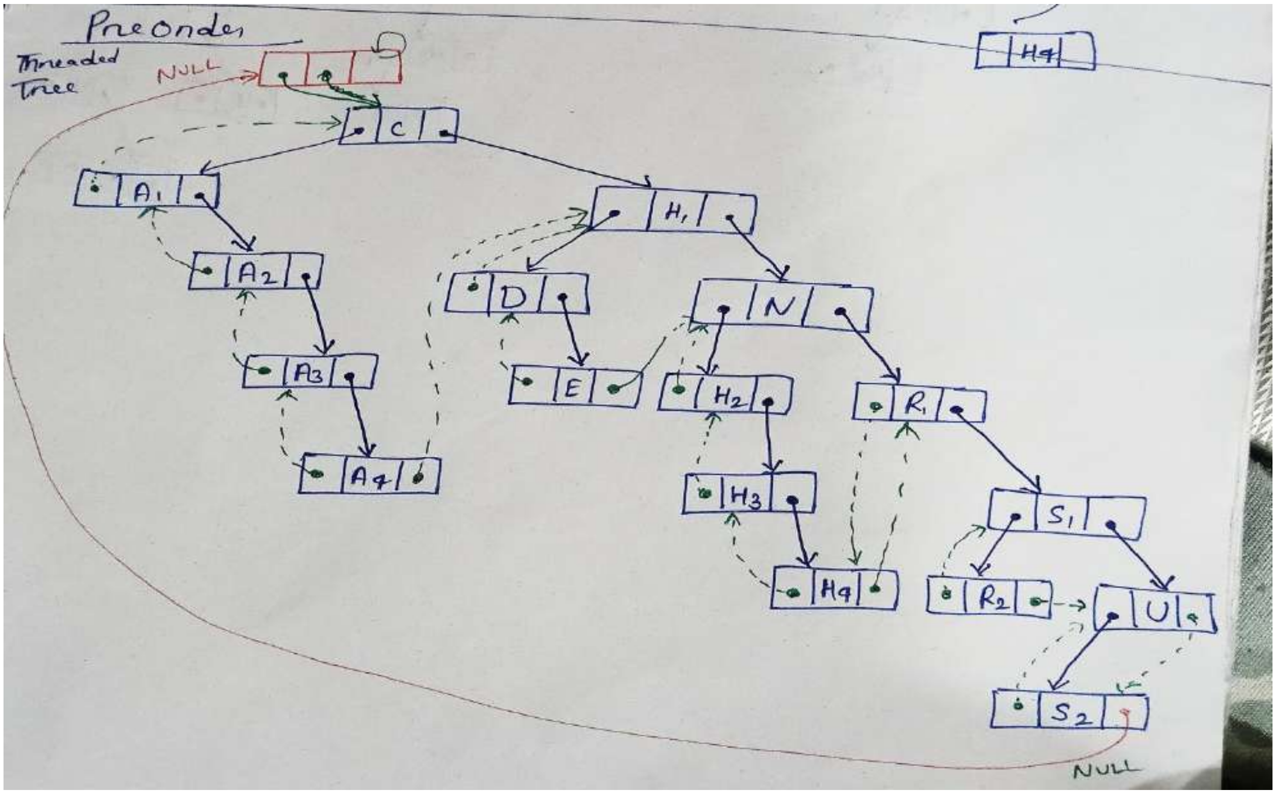
BST

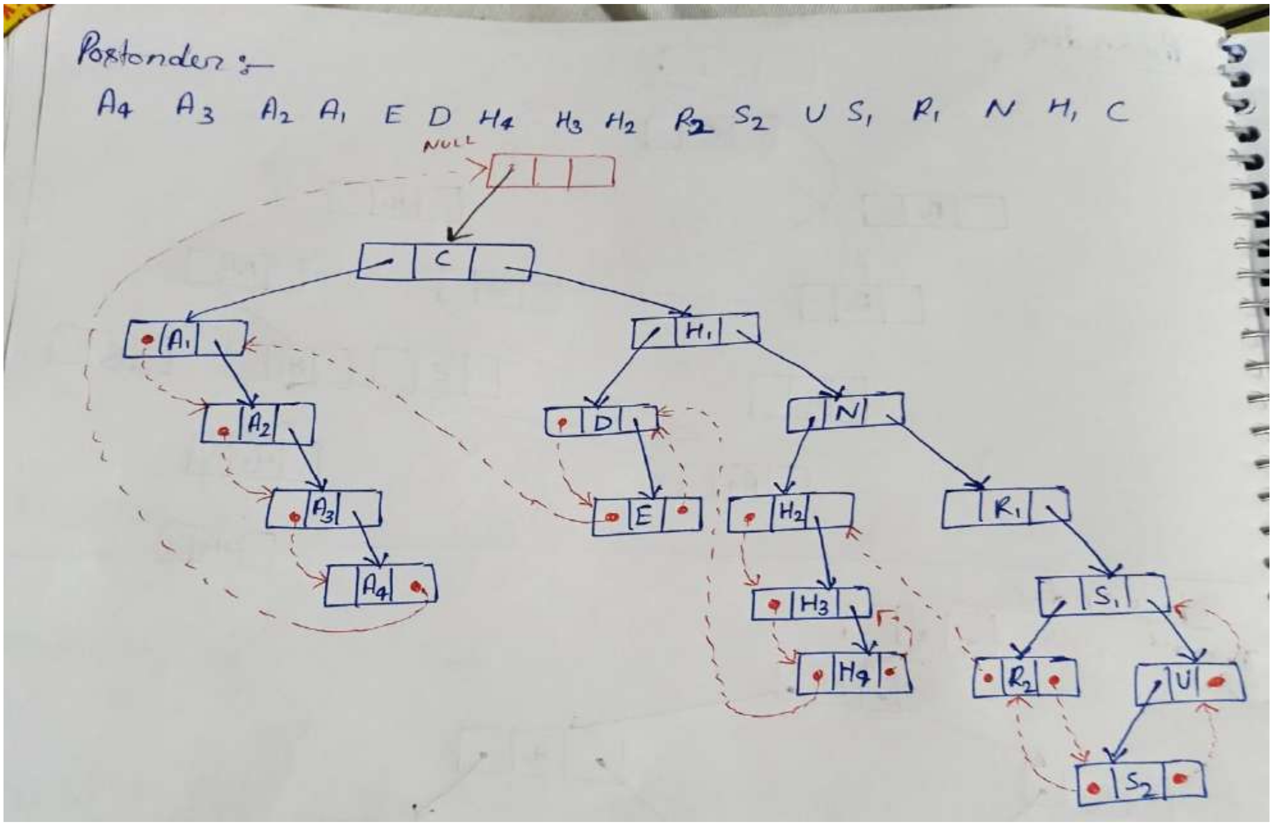
Inorder A_1 A_2 A_3 A_4 C D E H_1 H_2 H_3 H_4 N R_1 R_2 S_1 S_2 U

Preorder C, A_1, A_2 A_3 A_4 H_1 D E N H_2 H_3 H_4 R_1 S_1 R_2 U S_2

Postorder A_4 A_3 A_2 A_1 E D H_4 H_3 H_2 R_2 S_2 U S_1 R_1 N H_1 C

Inorder Threaded BST

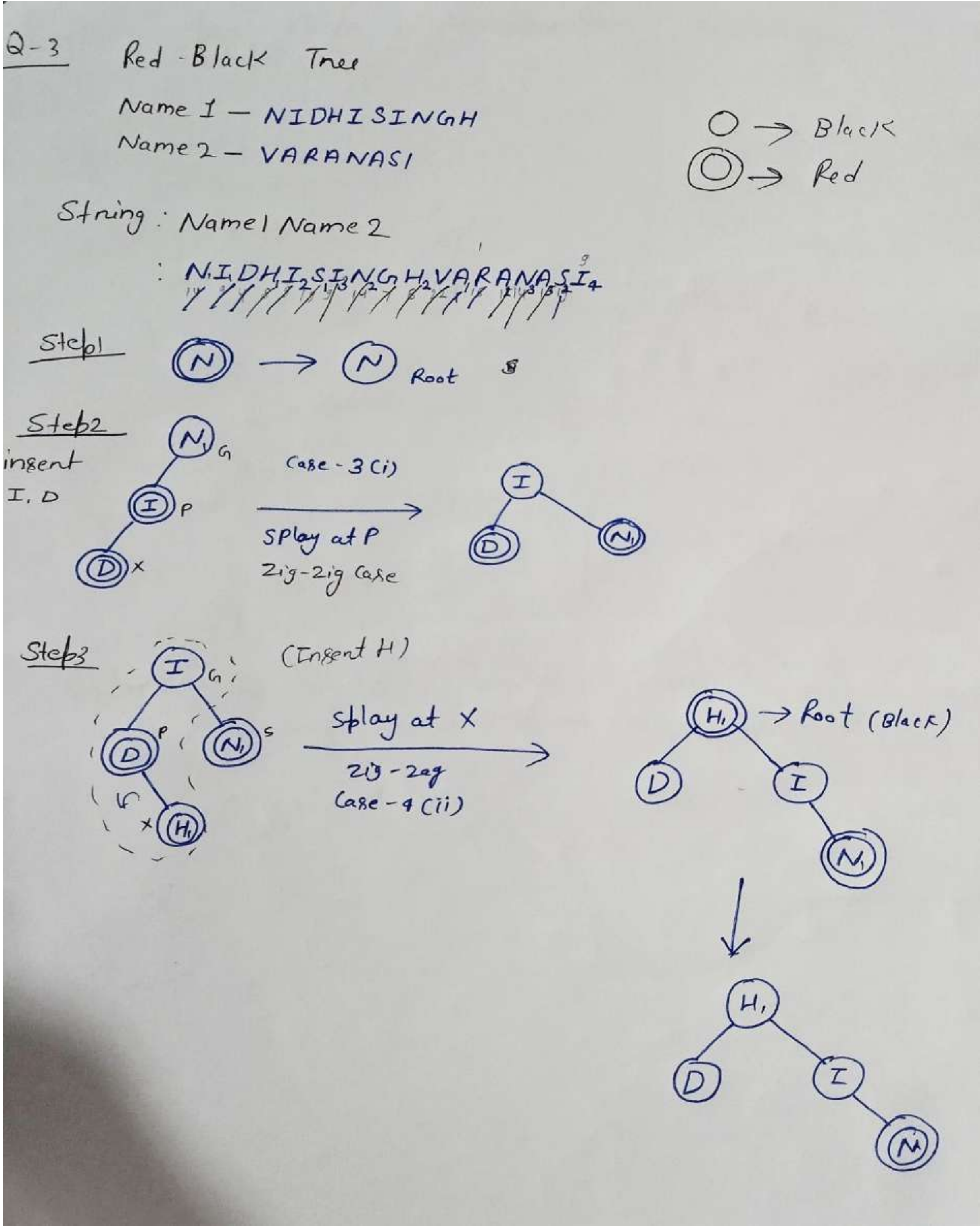




Postorder:-

A_4 A_3 A_2 A_1 E D H_4 H_3 H_2 R_2 S_2 U S_1 R_1 N H_1 C

3. Construct a red black tree from the letters of: name1:name2, where name1 is your name (as in passport) and name2 is name of your home town, || is string concatenation. Write all steps. Keep the duplicates (repetition of letters) at the right side (right sub tree)



Q-3 Red-Black Tree

Name 1 — NIDHISINGH

Name 2 — VARANASI

String : Name1 Name 2

: $N_1 I_1 D H_1 I_2 S_1 I_3 N_2 G H_2 V A_1 R A_2 N_3 A_3 S_2 I_4$

○ → Black

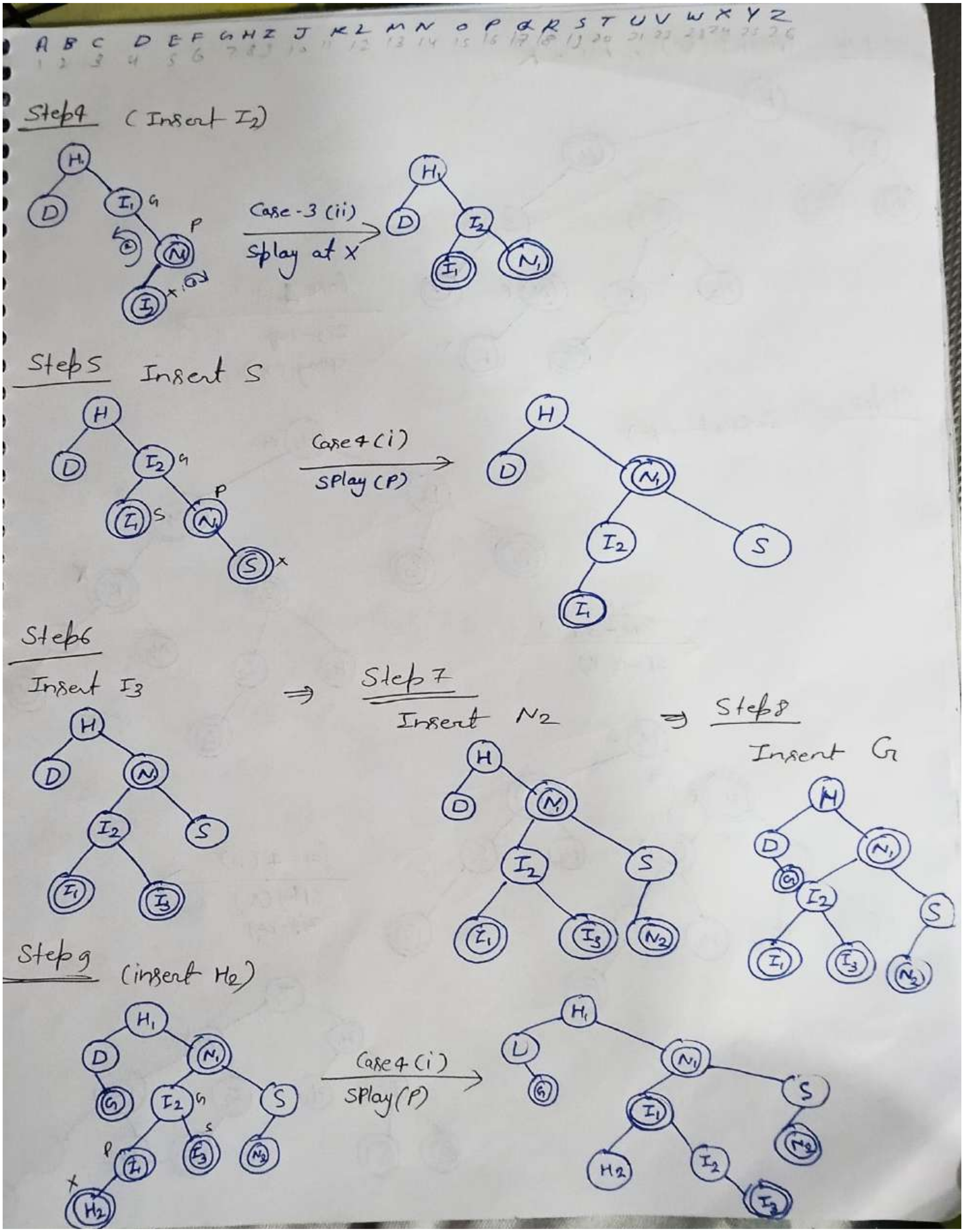
◎ → Red

Step1

Step2

insert I, D

Step3



Step4 (Insert I_2)

Case-3 (ii)

Splay at x

Step5 Insert S

Case4(i)

Splay(P)

Step6

Insert I_3 ⇒

Step7

Insert N_2 ⇒

Step8

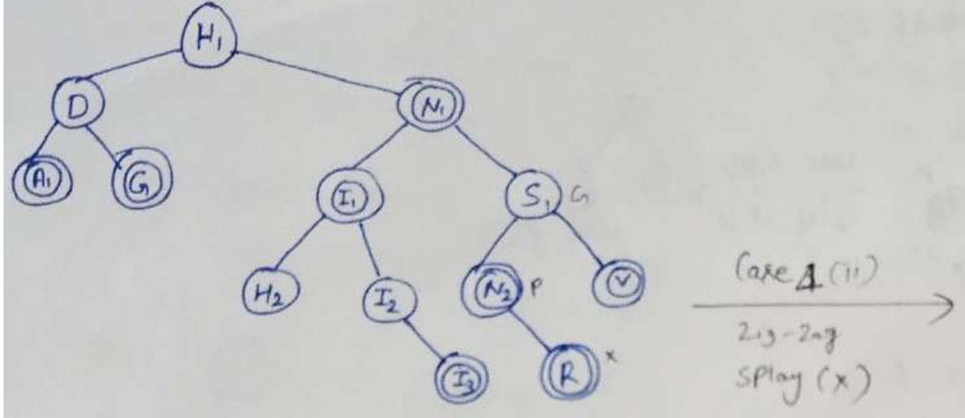
Insert G

Step9 (insert H_2)

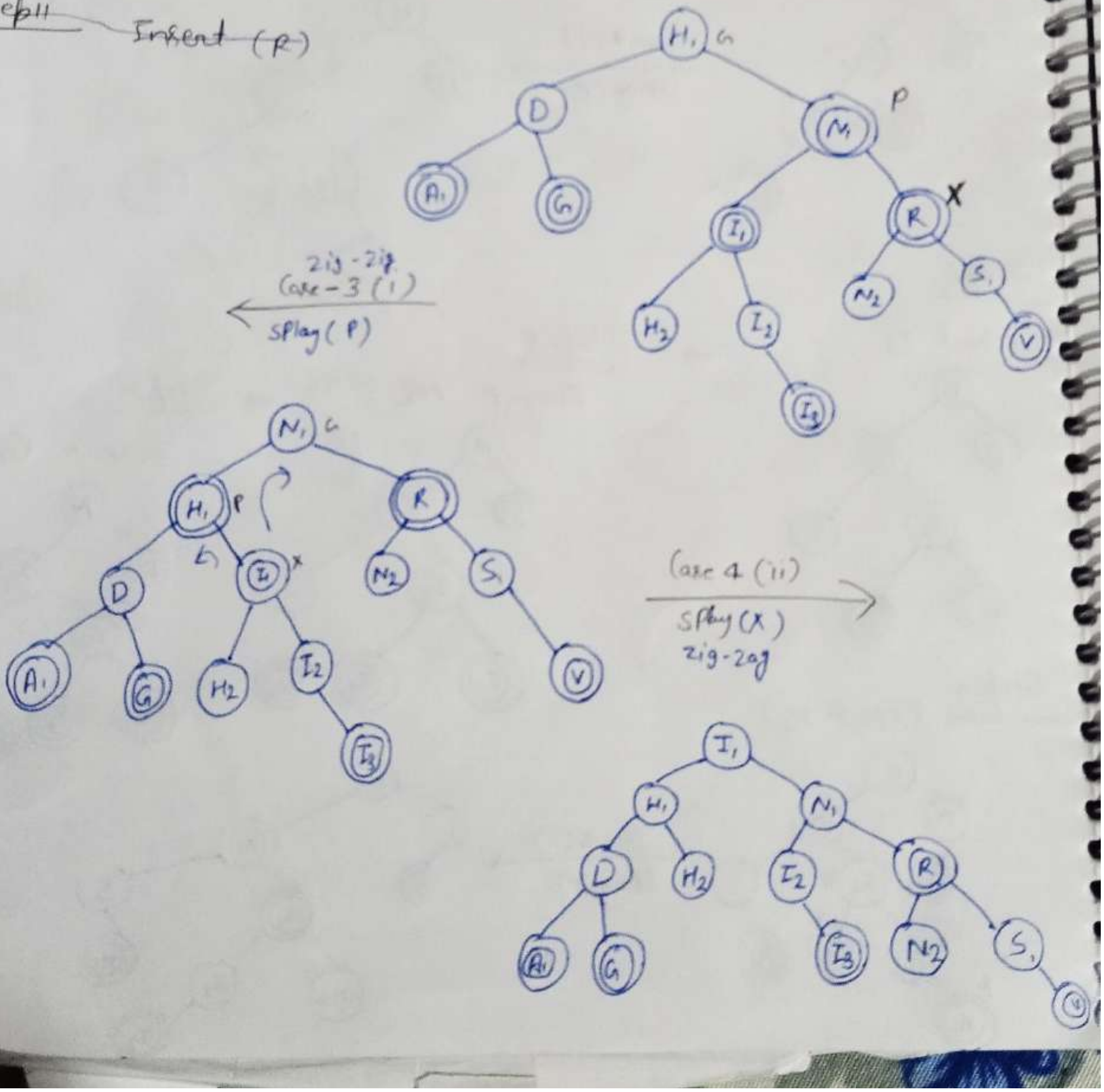
Case4(i)

Splay(P)

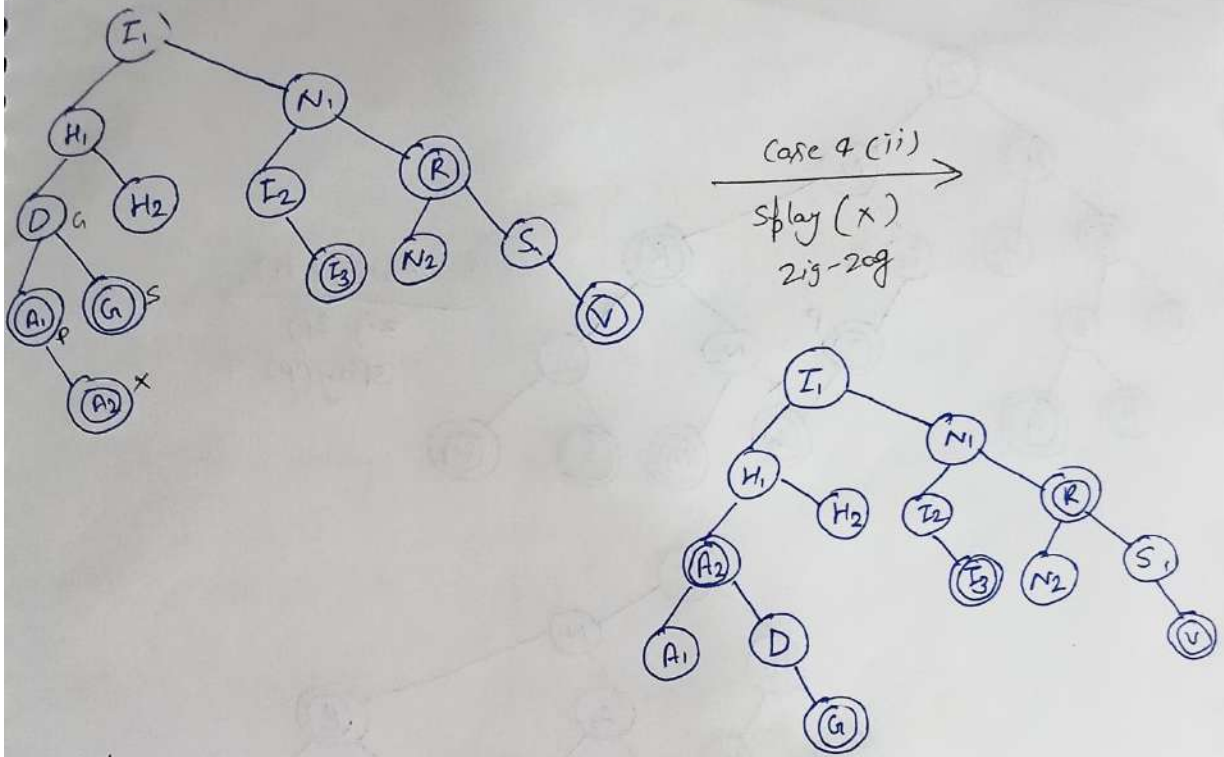
Step10 insert(V) , A_1 , R



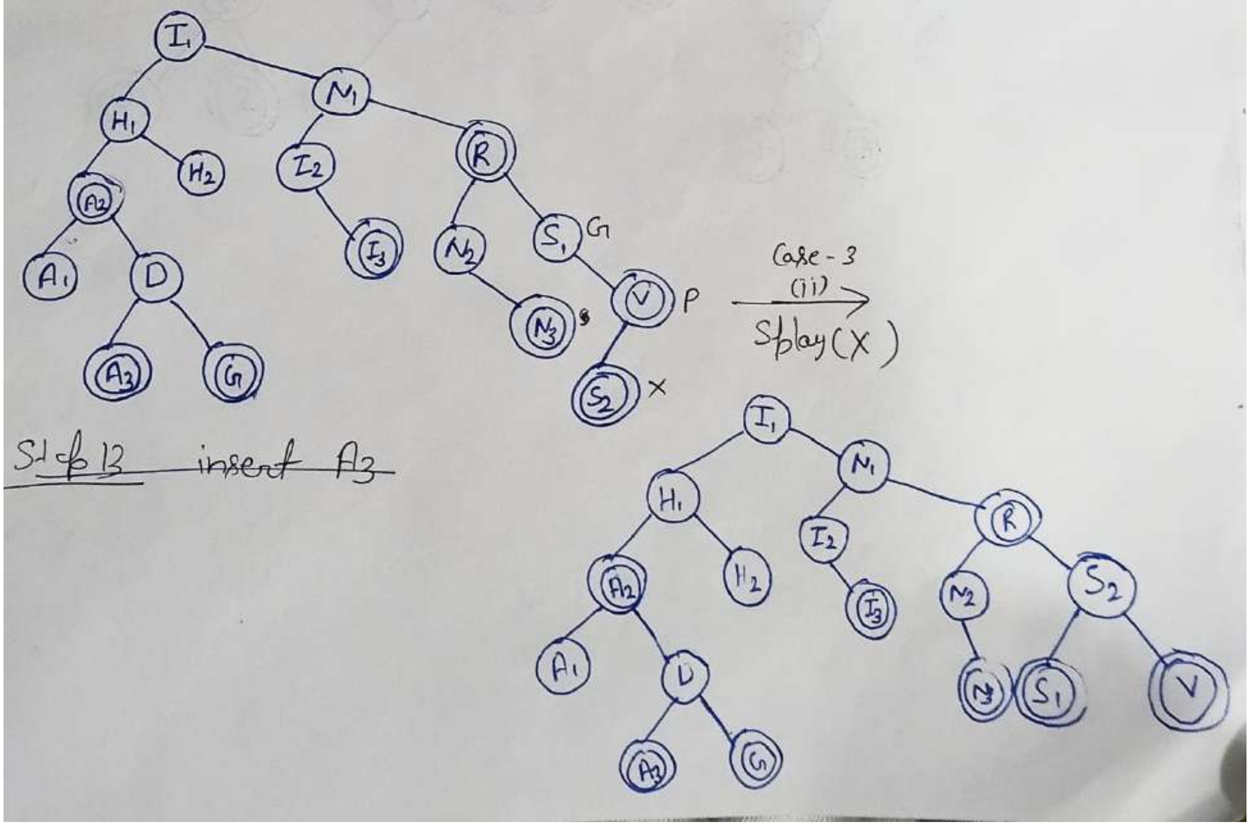
Step11 ~~Insert (R)~~



Step 11 Insert A_2

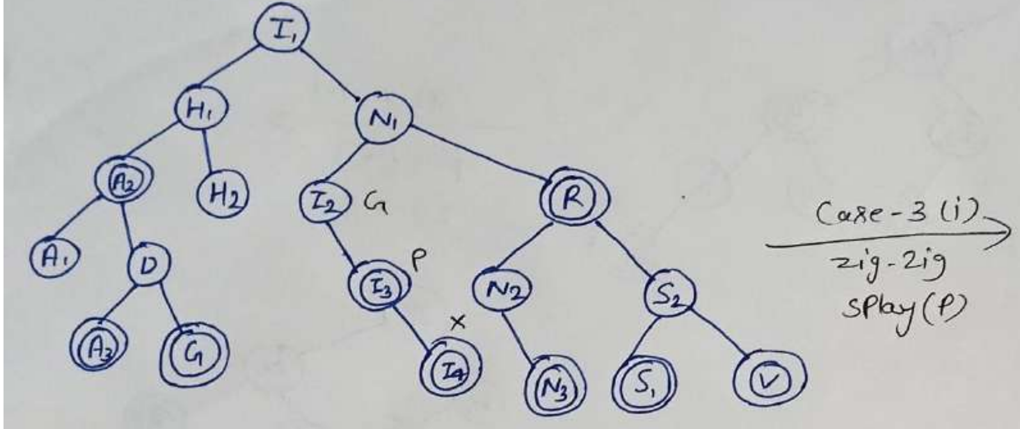


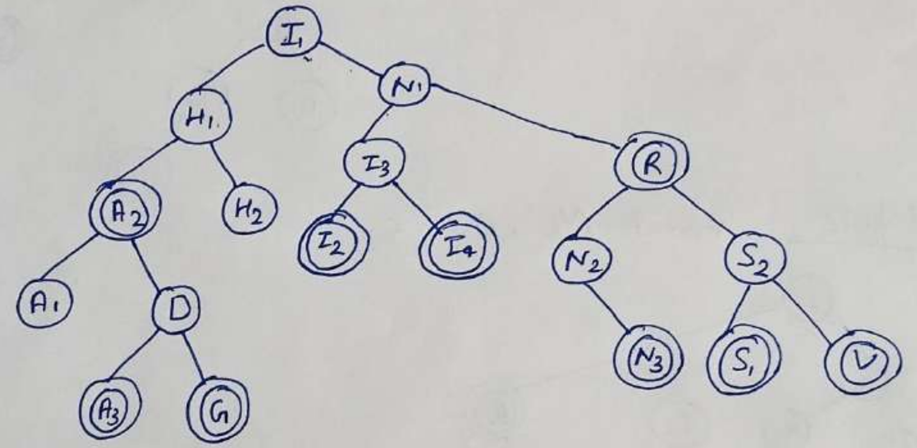
Step 12 insert N_3 , A_3 , S_2



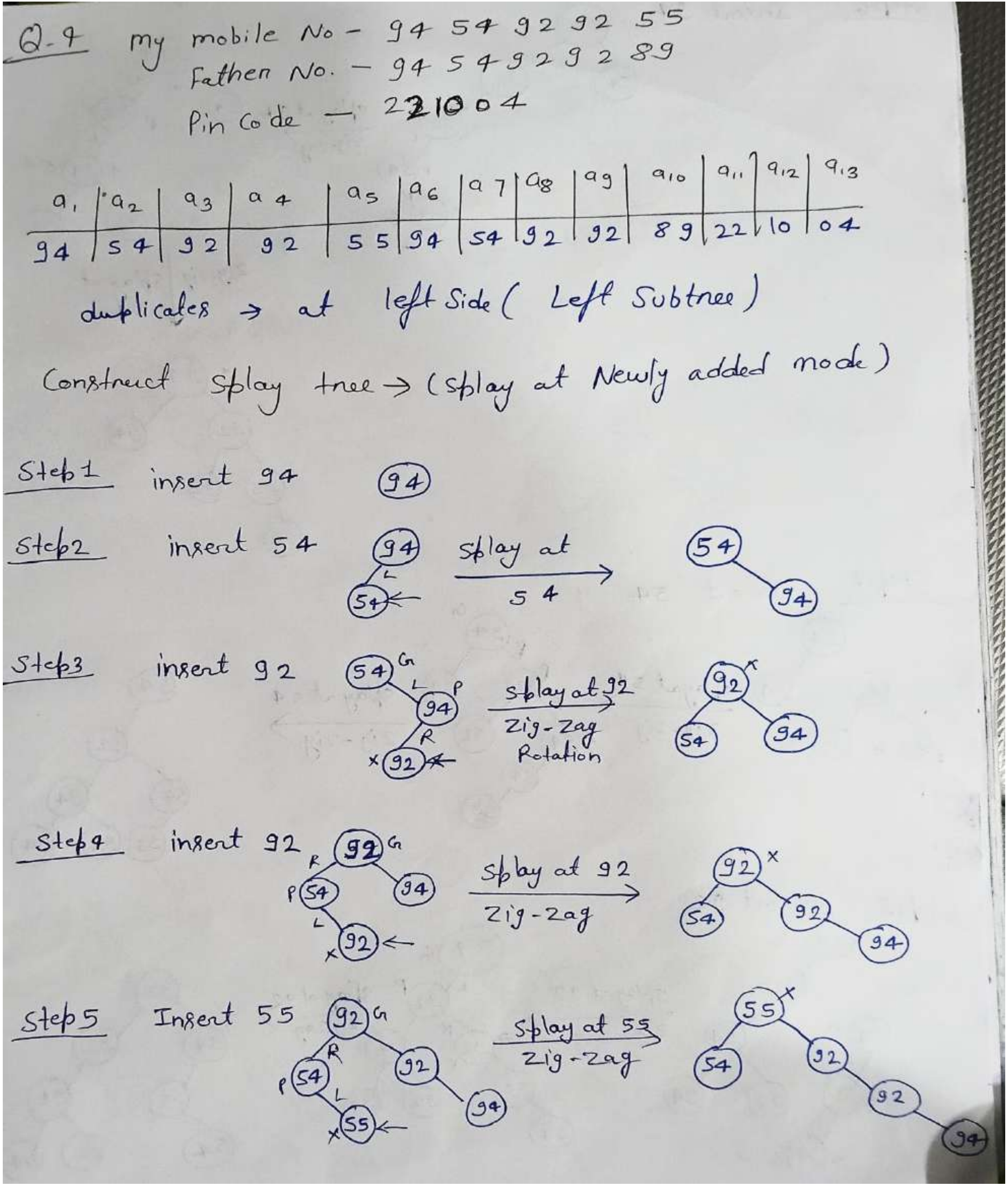
~~Step 13 insert A3~~

Step-13 insert (I_4)





4. Construct a splay tree from the following numbers a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13 where a1, a2, a3, a4, a5 are first two digits, 2nd two digits, …. Last two digits of your mobile number respectively, a6, a7, a8, a9, a10 are first two digits, 2nd two digits, …. Last two digits of your father's mobile number respectively, and a11, a12, a13 are first two digits, 2nd two digits and Last two digits of your hometown's pin code respectively. Write all steps. Keep the duplicates (repetition of numbers) at the left side (left sub tree)



Q-4 my mobile No - 94 54 92 92 55
Father No. - 94 54 92 92 89
Pin Code - 221004

a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13
94	54	92	92	55	94	54	92	92	89	22	10	04

duplicates → at left Side (Left Subtree)

Construct splay tree → (splay at Newly added node)

Step 1 insert 94

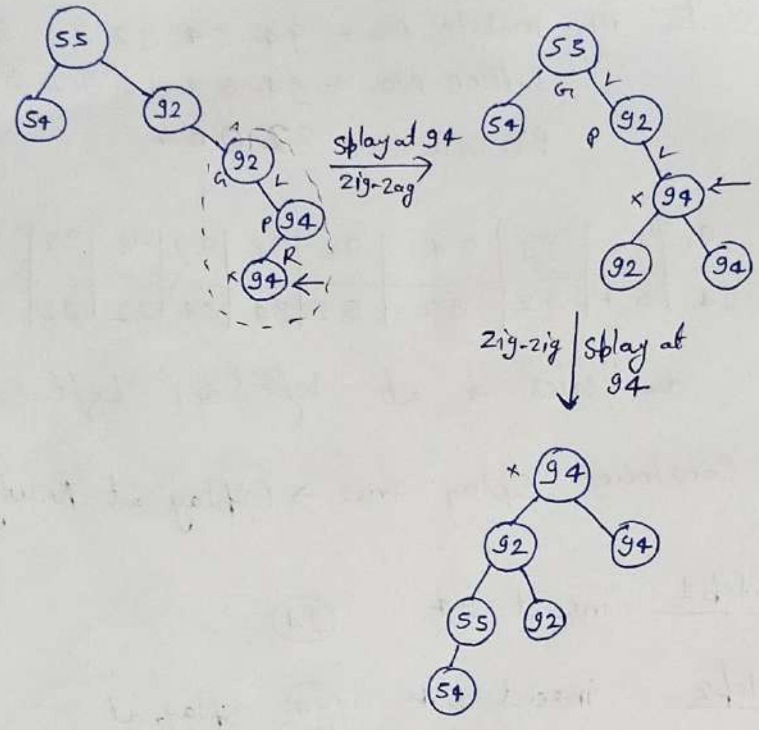
Step2 insert 54 $\xrightarrow{\text{splay at } 54}$

Step3 insert 92 $\xrightarrow{\text{splay at } 92}$ Zig-Zag Rotation

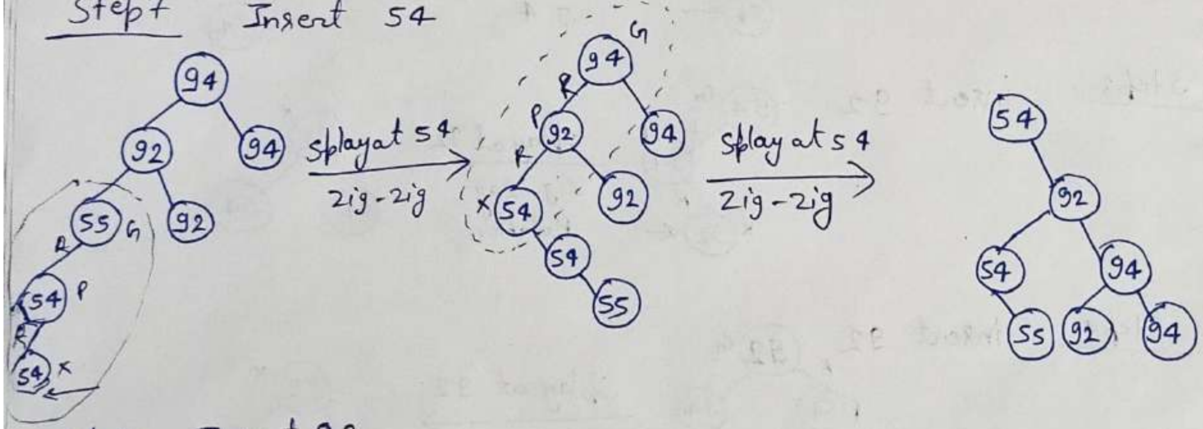
Step4 insert 92 $\xrightarrow{\text{splay at } 92}$ Zig-zag

Step5 Insert 55 $\xrightarrow{\text{splay at } 55}$ Zig-zag

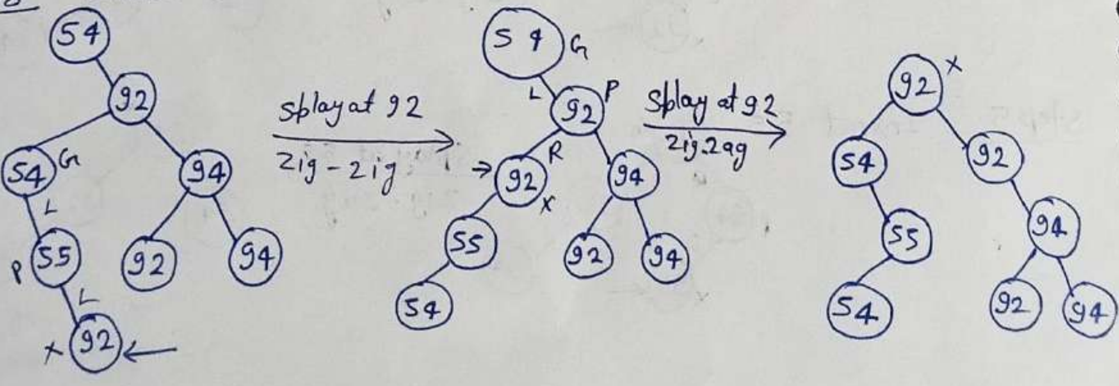
Step 6 Insert 94



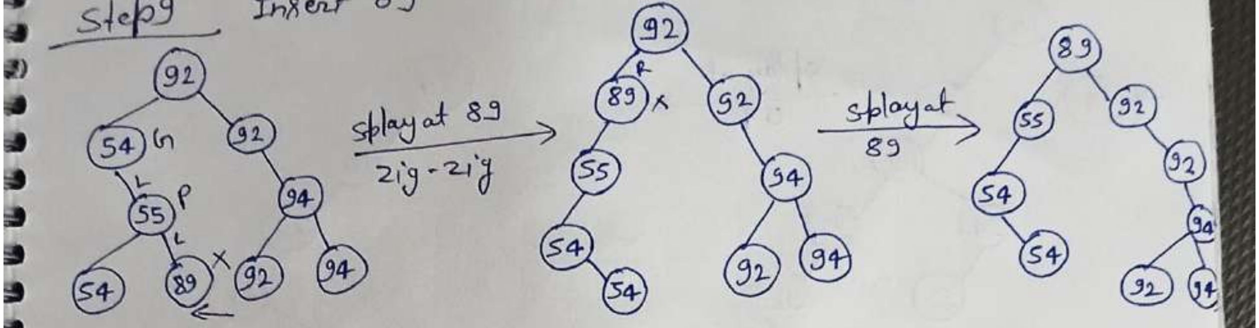
Step 7 Insert 54



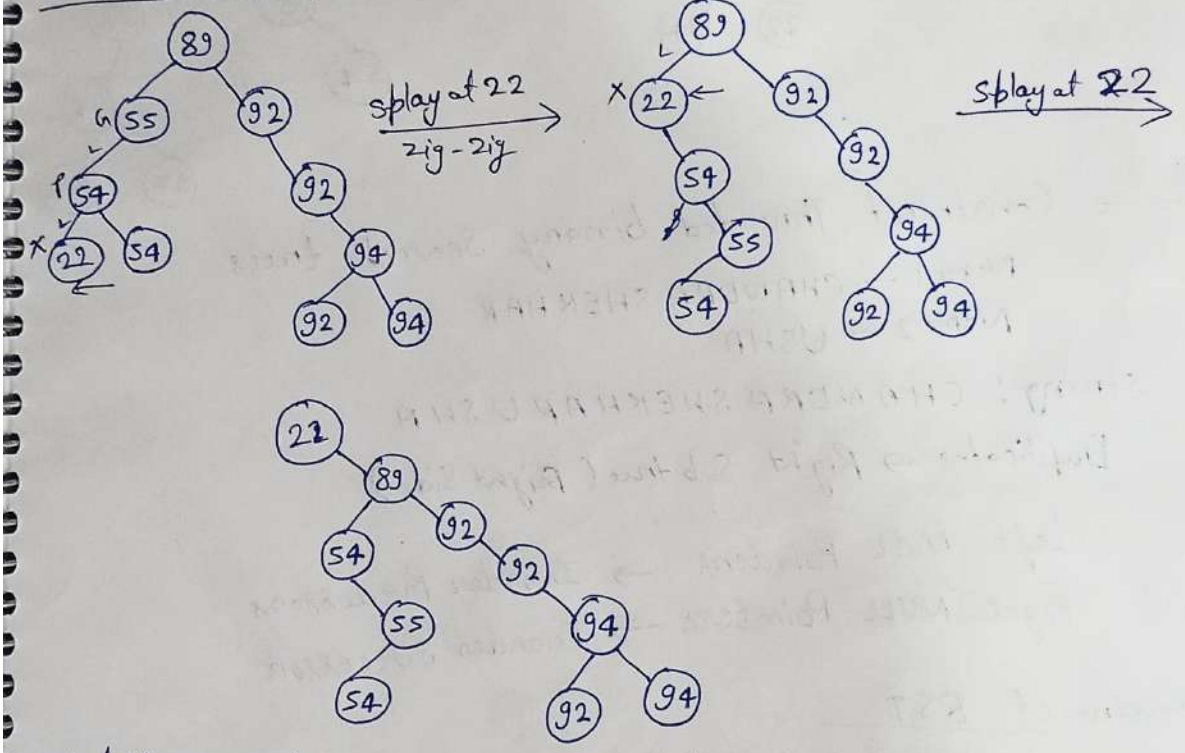
Step 8 Insert 92



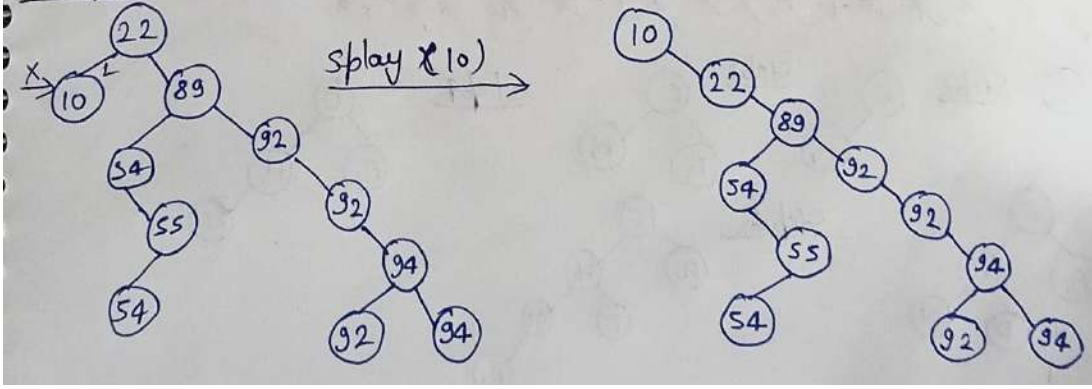
Step9 Insert 89

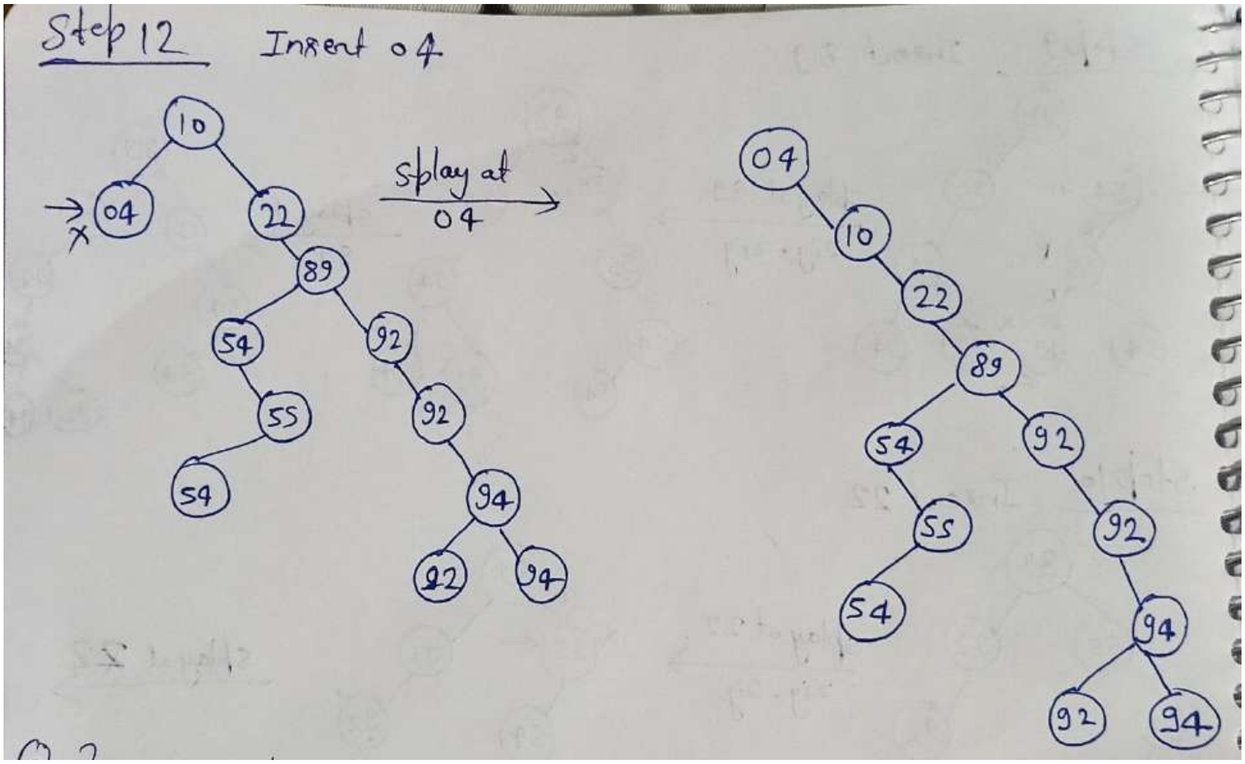


Step10 Insert 22



Step11 Insert 10





5. Sort digits of your mobile number using radix sort, heap sort, insertion, selection sort

Ans-5

mobile no — 94 54 92 92 55

① Radix Sort

A									
9	4	5	4	9	2	9	2	5	5

Pass 1 Each Pass Can apply Counting Sort

Step1 max digit = 9 (Size of Count array)

	0	1	2	3	4	5	6	7	8	9
Count	0	0	~~0~~ ~~1~~ 2	0	~~0~~ ~~1~~ 2	~~0~~ ~~1~~ ~~2~~ 3	0	0	0	~~0~~ ~~1~~ ~~2~~ 3

Initially 0.

Step 2 find Cumulative Count

Count[0] = Count[0]
Count [i] = Count[i-1] + Count[i]

	0	1	2	3	4	5	6	7	8	9
Count	0	0	~~2~~ ~~1~~ 0	2	~~3~~ ~~4~~ 2	4 ~~7~~ ~~6~~ ~~5~~	7	7	7	~~7~~ ~~8~~ ~~10~~ ~~9~~

Step3 ouatput array of Size 10.
Scan array A from right.
~~00~~ Output [Count [a[i]]]

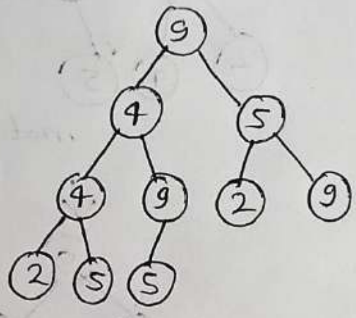
	1	2	3	4	5	6	7	8	9	10
output	2	2	4	4	5	5	5	9	9	9

Sorted no. — 2 2 4 4 5 5 5 9 9 9

② Heap Sort

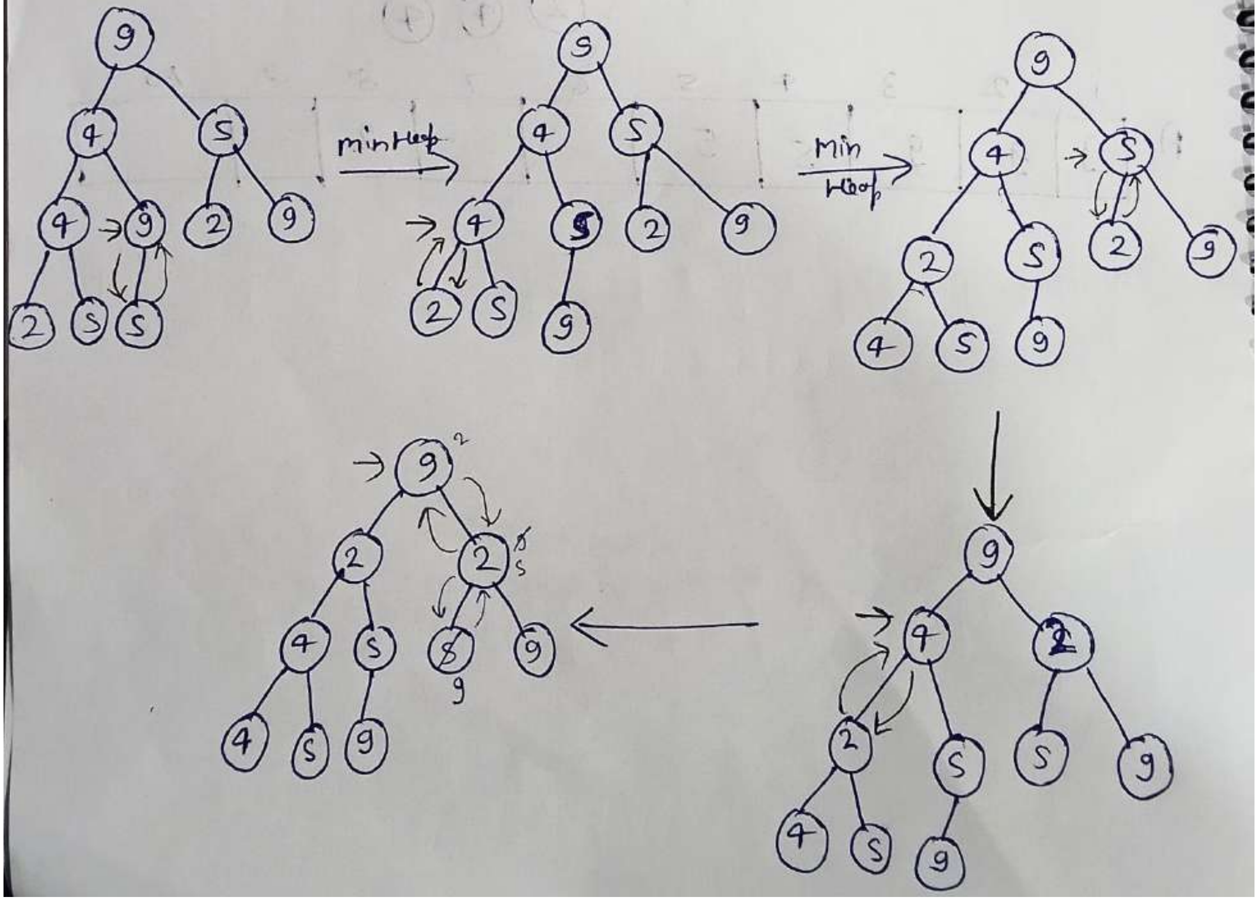
9 4 5 4 9 2 9 2 5 5

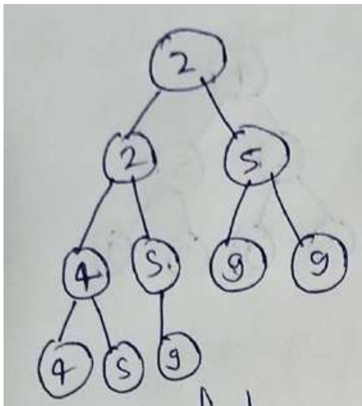
Build heap



min heap

Apply heapify



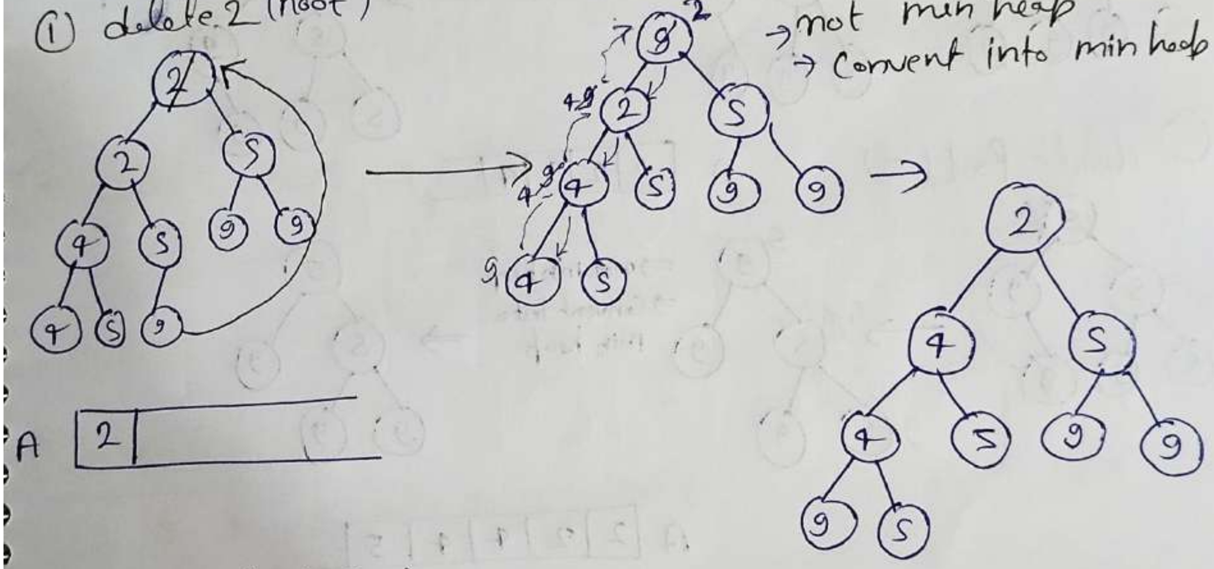


1	2	3	4	5	6	7	8	9	10	11
2	2	5	4	5	9	9	4	5	9	

min heap

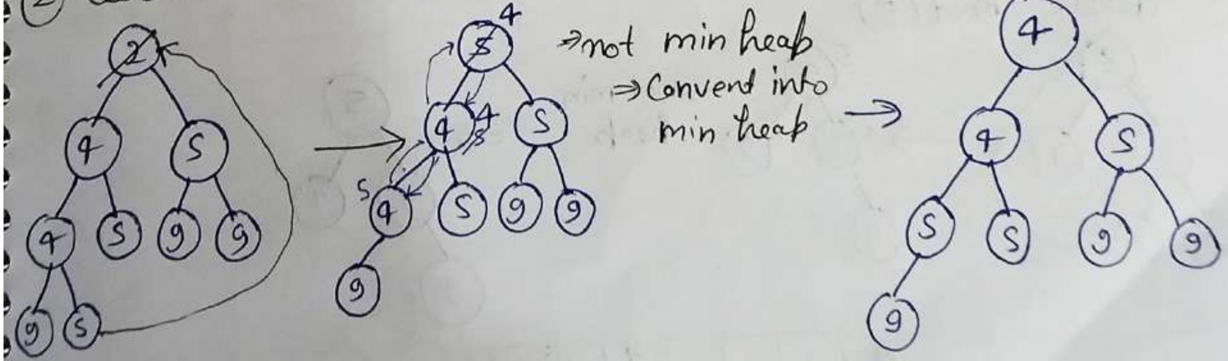
Delete element from Root ~~S~~to find the Sorted list.

① delete 2 (root)



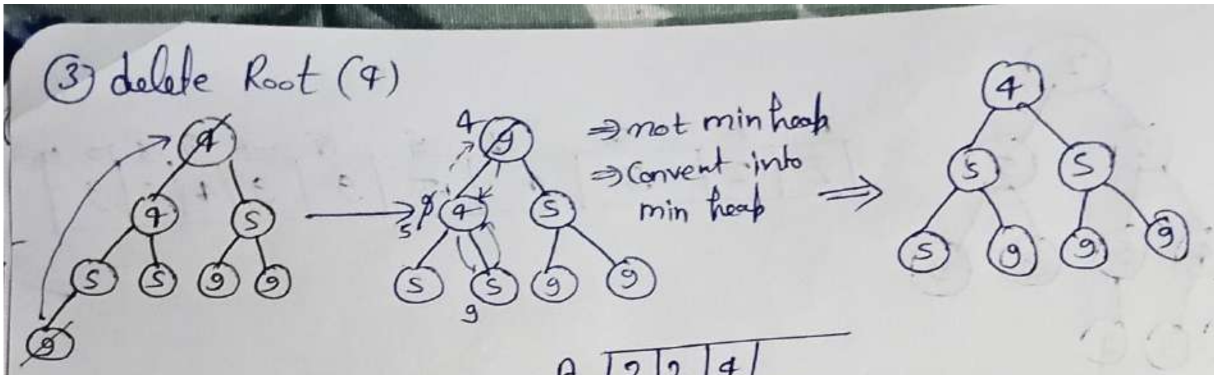
A	2	

② delete Root (2)



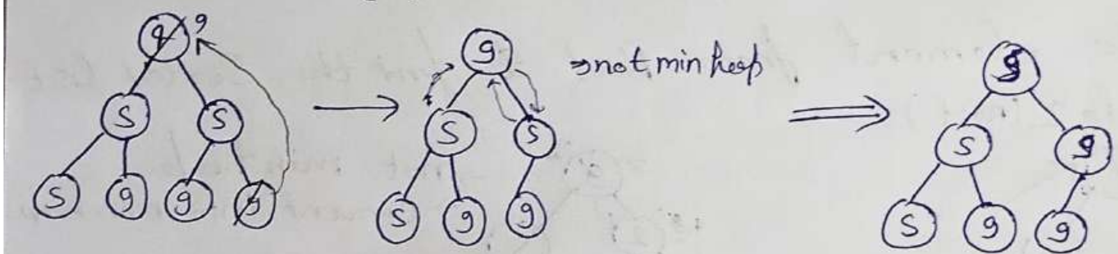
A	2	2	

③ delete Root (4)



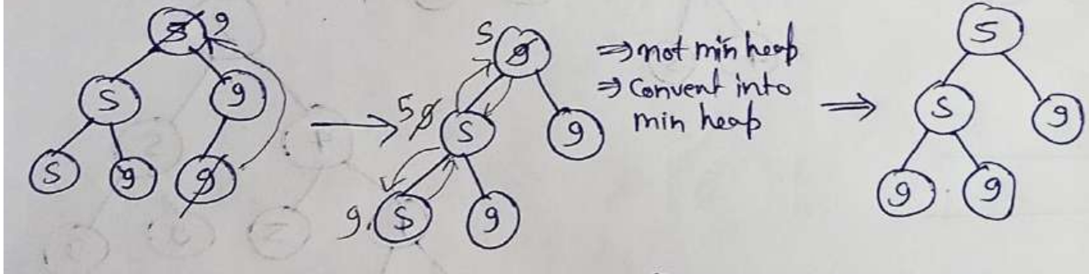
A				
2	2	4		

④ delete Root (4)



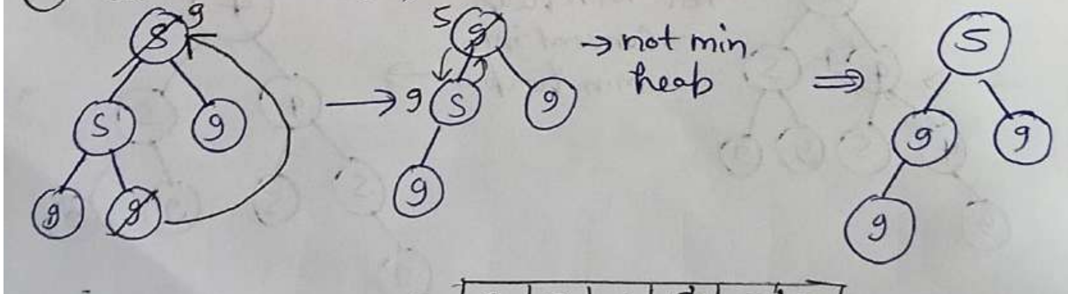
⑤ delete Root (5)

A				
2	2	4	4	

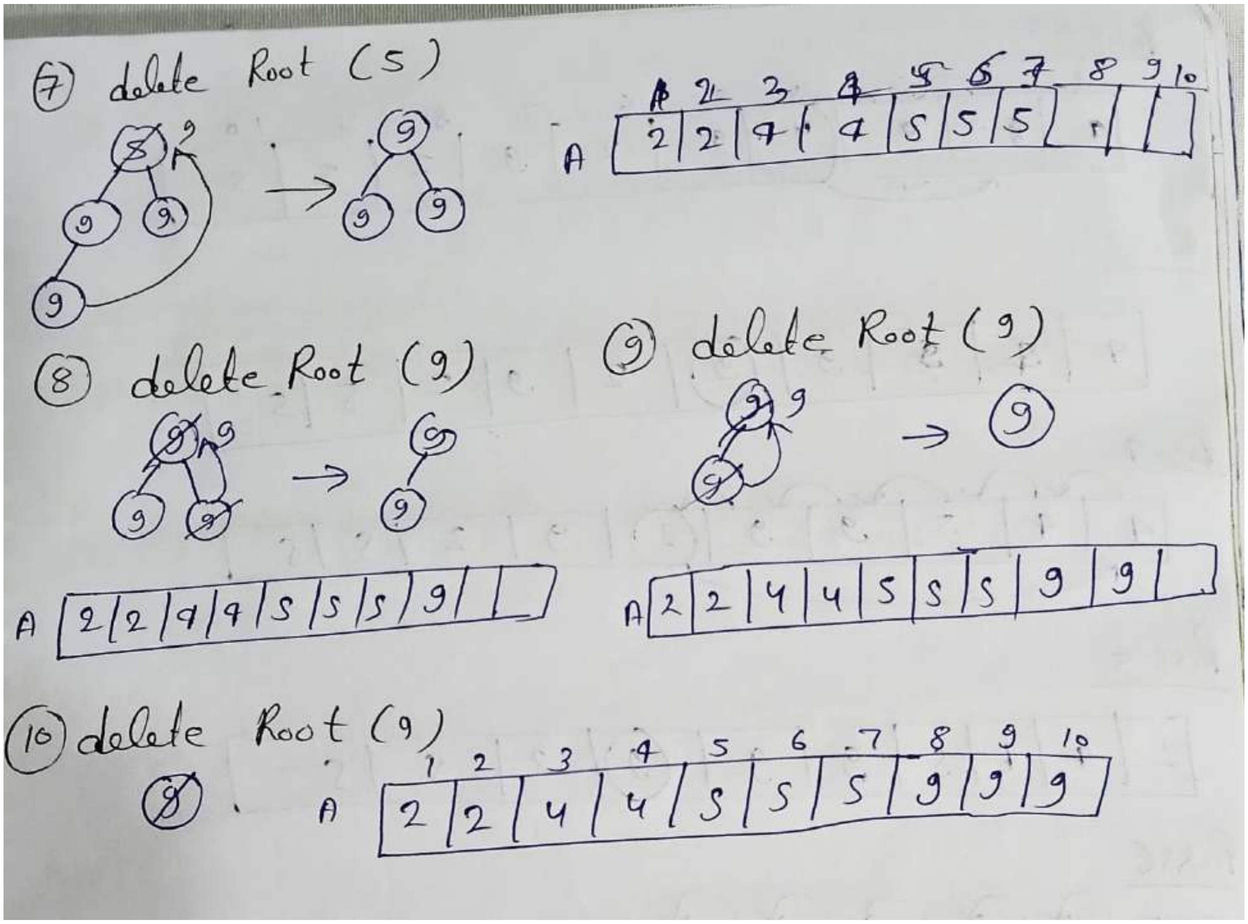


A					
2	2	4	4	5	

⑥ delete Root (5)



A					
2	2	4	4	5	5



⑦ delete Root (5)

~~8~~ 9
9
9
9
9
9
9

	1	2	3	4	5	6	7	8	9	10
A	2	2	4	4	5	5	5			

⑧ delete Root (9)

~~9~~ 9
9
~~9~~
9
9

A	2	2	4	4	5	5	5	9		

⑨ delete Root (9)

~~9~~ 9
~~9~~
9

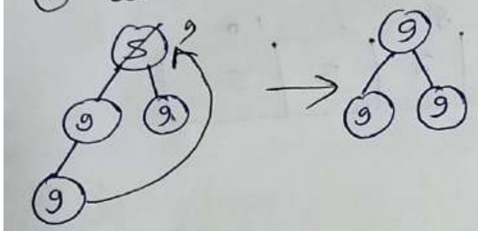
A	2	2	4	4	5	5	5	9	9	

⑩ delete Root (9)

~~9~~

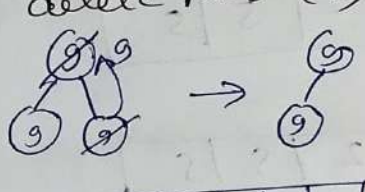
	1	2	3	4	5	6	7	8	9	10
A	2	2	4	4	5	5	5	9	9	9

⑦ delete Root (5)



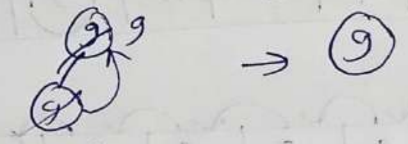
	1	2	3	4	5	6	7	8	9	10
A	2	2	4	4	5	5	5			

⑧ delete Root (9)



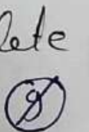
A	2	2	4	4	5	5	5	9		

⑨ delete Root (9)



A	2	2	4	4	5	5	5	9	9	

⑩ delete Root (9)



	1	2	3	4	5	6	7	8	9	10
A	2	2	4	4	5	5	5	9	9	9

③ Insertion Sort

9 4 5 4 9 2 9 2 5 5

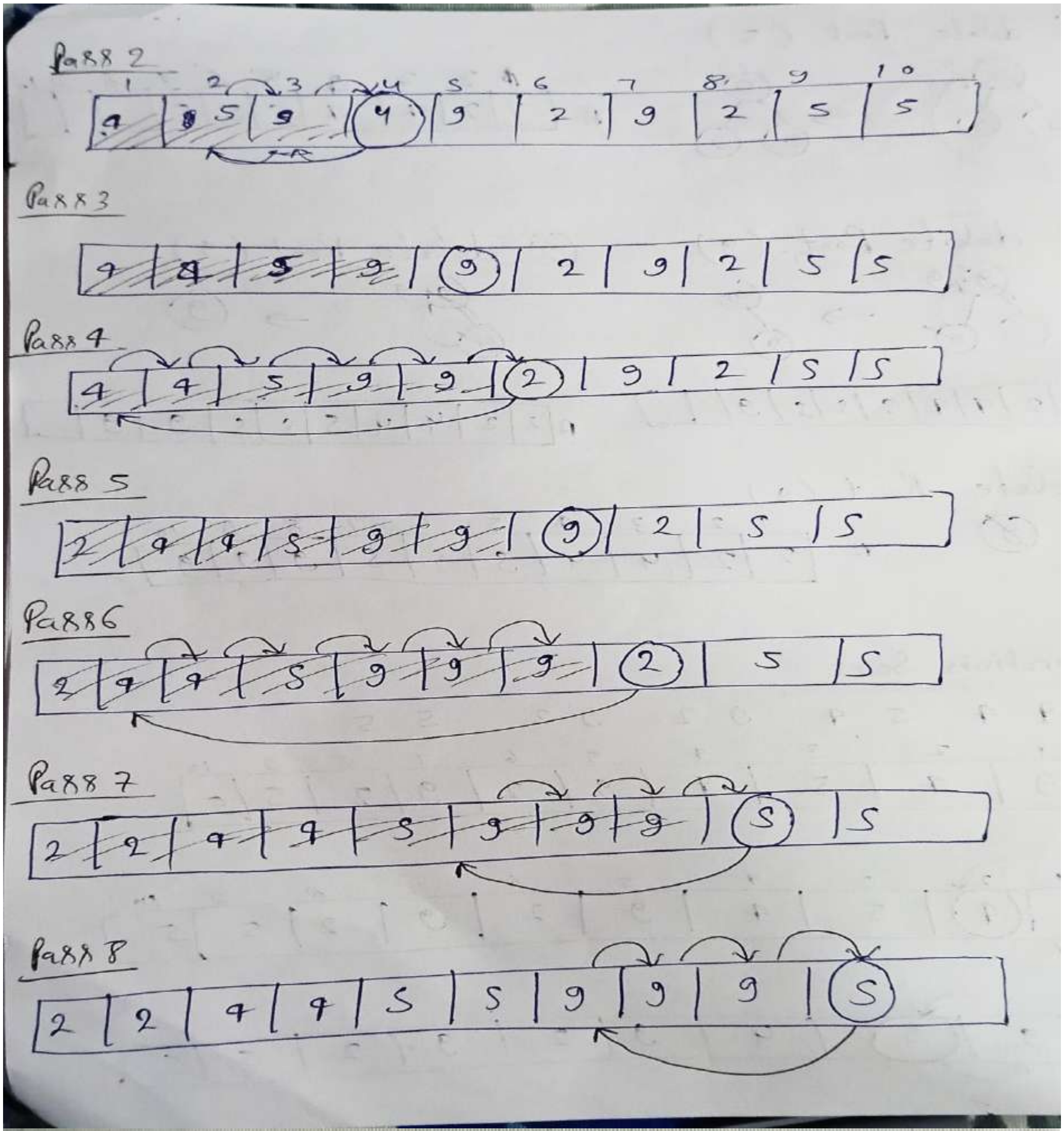
	1	2	3	4	5	6	7	8	9	10
A	9	4	5	4	9	2	9	2	5	5

Pass 1

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

Pass 2

4	9	5	4	9	2	9	2	5	5



Pass 2

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

Pass 3

4	4	5	9	9	2	9	2	5	5

Pass 4

4	4	5	9	9	2	9	2	5	5

Pass 5

2	4	4	5	9	9	9	2	5	5

Pass 6

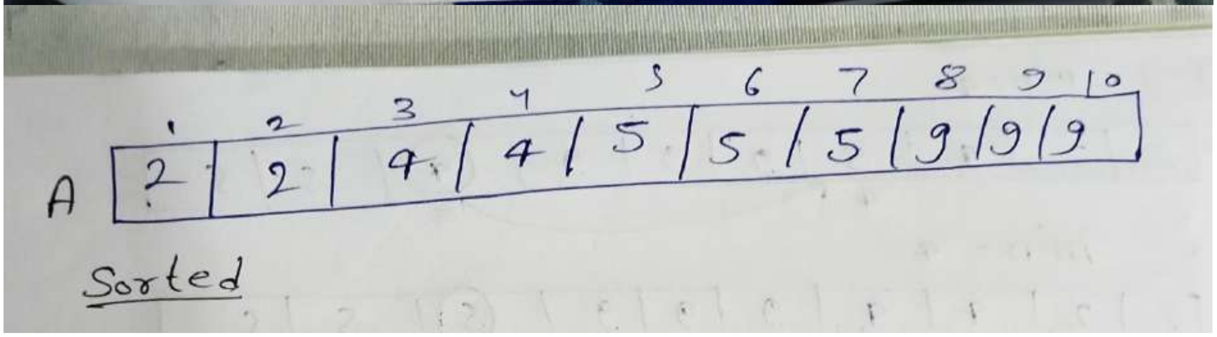
2	4	4	5	9	9	9	2	5	5

Pass 7

2	2	4	4	5	9	9	9	5	5

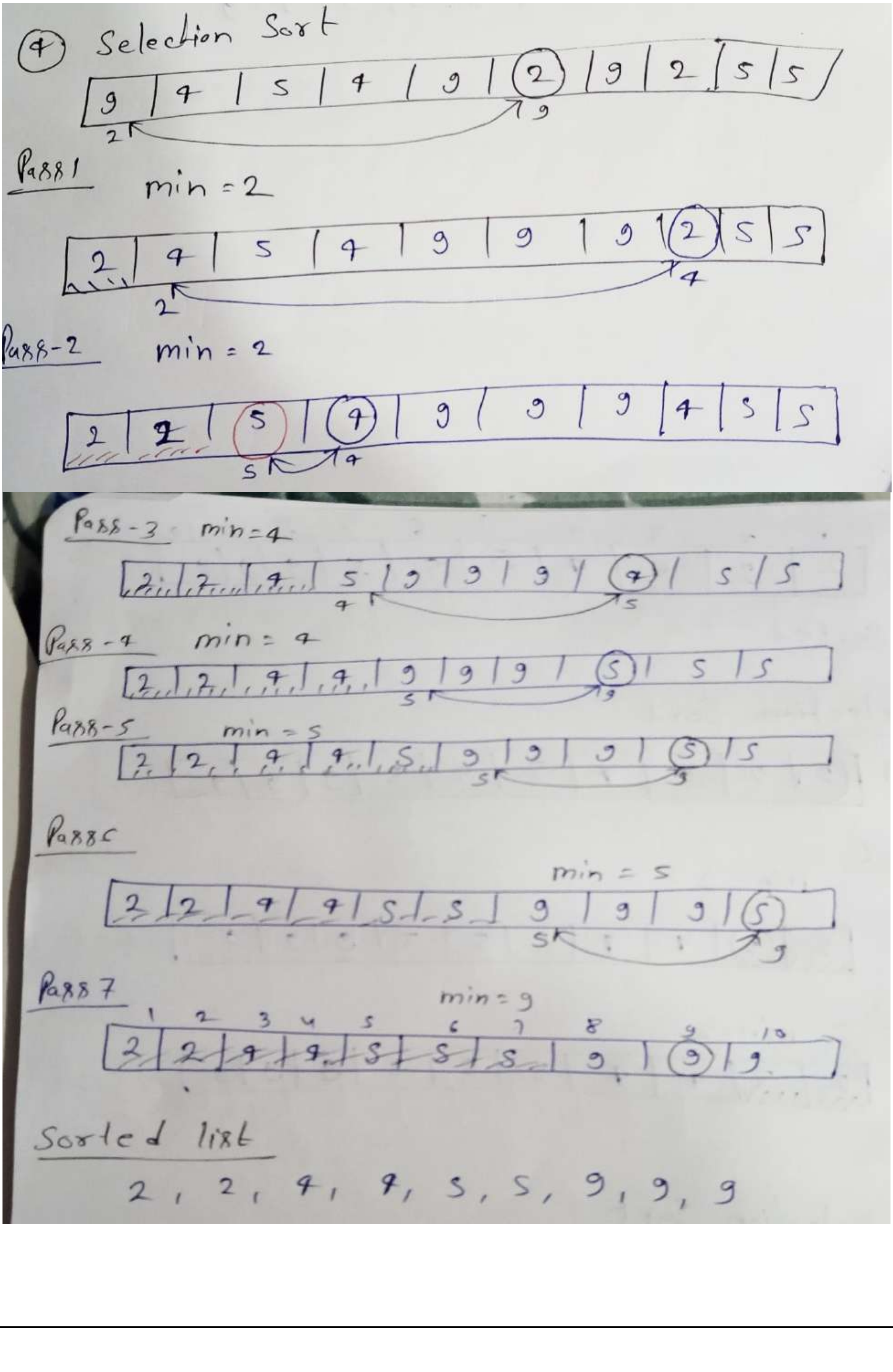
Pass 8

2	2	4	4	5	5	9	9	9	5



	1	2	3	4	5	6	7	8	9	10
A	2	2	4	4	5	5	5	9	9	9

Sorted



④ Selection Sort

9	4	5	4	9	2	9	2	5	5
2					9				

Pass 1 min = 2

2	4	5	4	9	9	9	2	5	5
	2						4		

Pass-2 min = 2

2	2	5	4	9	9	9	4	5	5
		5	4						

Pass-3 min=4

2	2	4	5	9	9	9	4	5	5
			4				5		

Pass-4 min = 4

2	2	4	4	9	9	9	5	5	5
				5			9		

Pass-5 min = 5

2	2	4	4	5	9	9	9	5	5
					5			9	

Pass6

min = 5

2	2	4	4	5	5	9	9	9	5
						5			9

Pass 7

min = 9

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

Sorted list

2, 2, 4, 4, 5, 5, 9, 9, 9