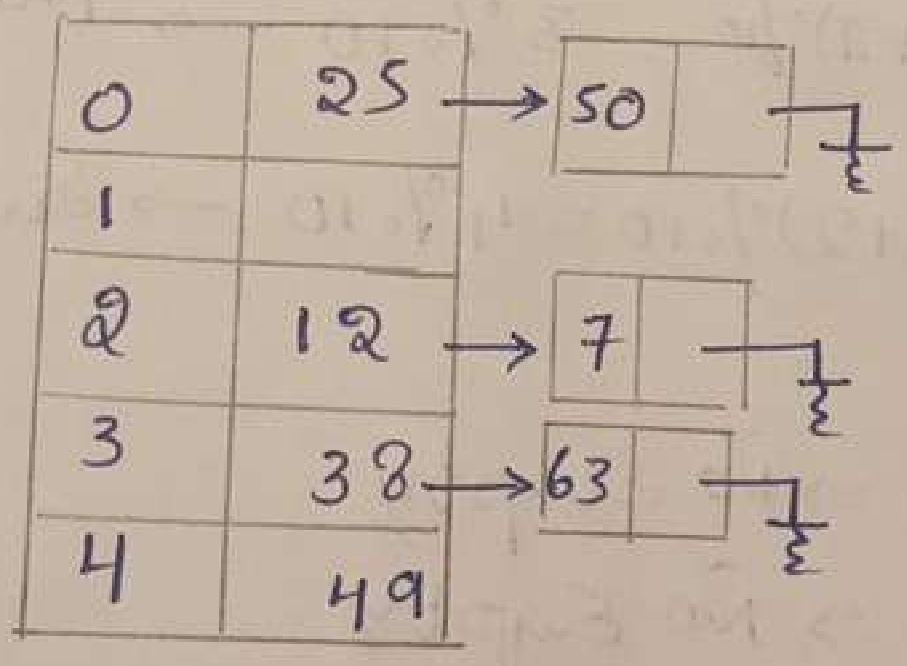
1. Given a hash table size 5 & a hash function x%5, insert the following keys into the hash table using seperate chaining to handle collisions:
12, 25, 38, 49, 50, 63, 7.

Sol.

* Given Table size 5
* Given Hash function x%5

-> 12%5 = 2
-> 25%5 = 0
-> 38%5 = 3
-> 49%5 = 4
-> 50%5 = 0
-> 63%5 = 3
-> 7%5 = 2

Hash Table:



j = 25, j > 18,
j = 10, j < 18,
Here i, j are co

10	18	25	27	36	45

2. Given hash table of size 10 & the following keys: 12, 18, 13, 2, 3, 23, 5, 15. Use the division method of hashing & linear probing to handle collisions. Show the final state of the hash table.

Sol. Given table size = 10

Elements = 12, 18, 13, 2, 3, 23, 5, 15

1) Insert '12', index = 12 % 10 = 2, i = 0

Index = (2 + 0) % 10 = 2 → Empty

2) Insert '18', 18 % 10 = 8, i = 0

Index = (8 + 0) % 10 = 8 → Empty

3) Insert '13', 13 % 10 = 3, i = 0

Index = (3 + 0) % 10 = 3 → Empty

4) Insert '2', 2 % 10 = 2, i = 0

Index = 2 → No Empty

i = 1

(2 + 1) % 10 = 3 % 10 → No Empty

i = 2

(2 + 2) % 10 = 4 % 10 → Empty

5) Insert '3'

i = 0 ⇒ No Empty

i = 1 ⇒ No Empty

i = 2 ⇒ (3 + 2) % 10 = 5 → Empty

6) Insert '23'

23 % 10 = 3

i = 0 ⇒ No Empty

i = 1 ⇒ No Empty

i = 2 ⇒ No Empty

i = 3 ⇒ 6 % 10 ⇒ Empty

[redacted]

j = 10, j

Here i, j are

10	18	25	27	36	45

↓
Sorted

7) Insert '5' , 5%10 = 5 => No Empty

i=0 => No Empty

i=1 => No Empty

i=2 => (5+2) % 10 = 7 → Empty

8) Insert '25' , 25%10 = 5

i=0 => No Empty

i=1 => No Empty

i=2 => No Empty

i=3 => (5+3)%10 = 8 → No Empty

i=4 => 9%10 = 9 → Empty

	After '12'	18	13	2	3	23	5	15
0								
1								
2	12	12	12	12	12	12	12	12
3			13	13	13	13	13	13
4				2	2	2	2	2
5					3	3	3	3
6						23	23	23
7							5	5
8		18	18	18	18	18	18	18
9								15
10								

j =10, j<18, s to

Here i,j are corssed

10	18	25	27	36	45

↓
Sorted

3) a) Implement a Bubble Sort algorithm with an optimization to reduce the number of comparisons in cases where array is partially sorted.

Sol.

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items, & swaps them if they are in the wrong order. The pass through the list is repeated untill the list is sorted. An optimized version of Bubble Sort can reduce the number of comparisons when the array is partially sorted by adding a flag to monitor if any swaps were made during a pass. If no swaps were made, the array is already sorted, & we can exit easily.

3) b) Sort the following [38, 27, 43, 3, 9, 82, 10] using Quick sort.

Sol.

Choose Pivot

38	27	43	3	9	82	10
P	i ↷ i					

Compare P & i

i = 27, 27 < P, increment

i = 43, 4 > P, stop

38	27	43	3	9	82	10
P		i			↶	j

compare P & j

j = 10, j < P, stop

38	27	10	3	9	82	43
P						

swap i & j

j = 25, j > 18, ''

j = 10, j < 18, stop

Here i, j are corssed

10	18	25	27	36	45

↓

Sorted

27	10	25	18	36	45

P j j j

compare P&j

j=45, j>P,

j=36, 36>P, decrement

j=18, 18<27, stop

Here i×j (icross

swap P&j

18	10	25	27	36	45

→ partition result

38	27	10	3	9	82	43

P i →i ↪i →i ↪i

compare i&P

i = 27, 27<P, increment

i = 10, 10<P, increment

i =3, 3<P, increment

i =9, 9<P, increment

i=82, 82>P, stop

38	27	10	3	9	82	43

P

j ↖ j ↜ j

j= 43, j>P, decrement

j =82, j>P, decrement

j=9, j<P, stop

38	27	10	3	9	82	43

P

i×j

swap j&P

9	27	10	3	38	82	43

P

Now choose New P

9	27	10	3	38	82	43

i ↪ P i ↪ i ↪ i

Compare P&i

i =9, i <P, increment

i=10, i<P, "

i =3, i<P, "

i =38, i>P, stop

9	27	10	3	38	82	43

P

↖ i j ↜ j

compare j&P

j = 43, j >P, decrem

j =82, j>P, "

j =3, j<P. stop

9	27	10	3	38	82	43

P j i

i×j

i crossed j

swap j&P

9	3	10	27	38	82	43

9	3	10	27	38	82	43

choose new P

9	3	10	27	38	82	43
i						P

compare i&P
i=9, i<P, increment
i=3, i<P, "
i=10, i<P, "
i=27, i<P, "
i=38, i<P, "
i=82, i>P, stop

9	3	10	27	38	82	43
				i	j	P

9	3	10	27	38	82	43
				i×j	j	P

compare j&P
j= 82, j>P, decr
j= 38, j<P, stop

9	3	10	27	38	82	43
				j	j	P

swap i&P

9	3	10	27	38	43	82

9	3	10	27	38	43	82
P	i → i					

choose new pivot
compare i&P
i=3, i<P, increment
i=10, i>P, stop

9	3	10	27	38	43	82
P		i	j			

compare j&P
j=27, j>P, decrement
j = 3, j<P, stop

9	3	10	27	38	43	82
P	j	i				

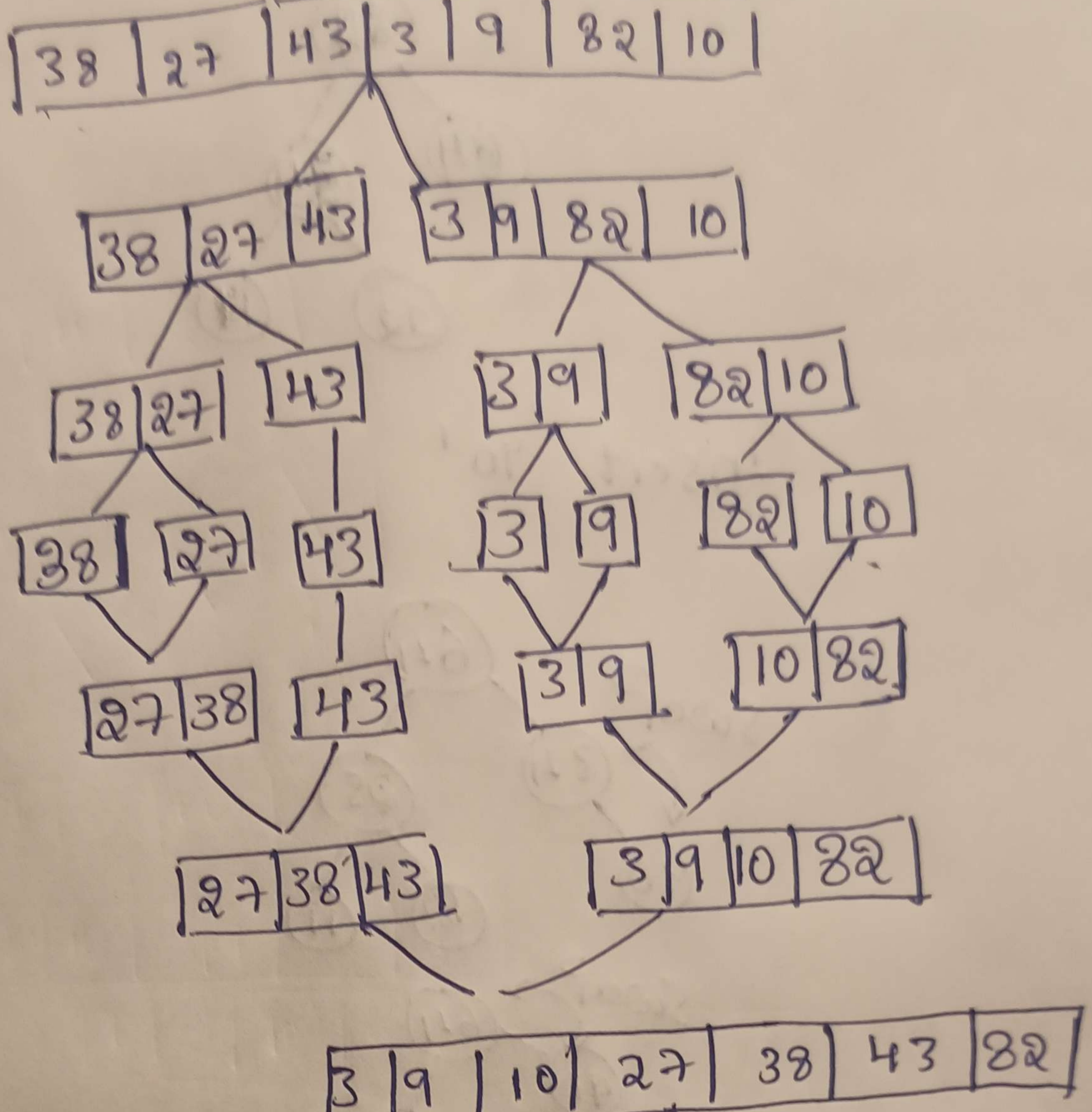
swap j&P

Sorted

3	9	10	27	38	43	82

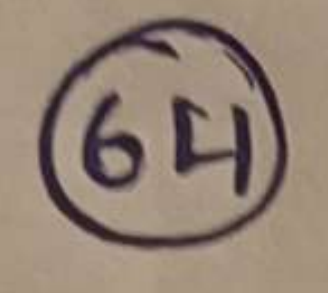
4. a) Sort the array [38,27,43,3,82,10] using Merge Sort.

Sol.

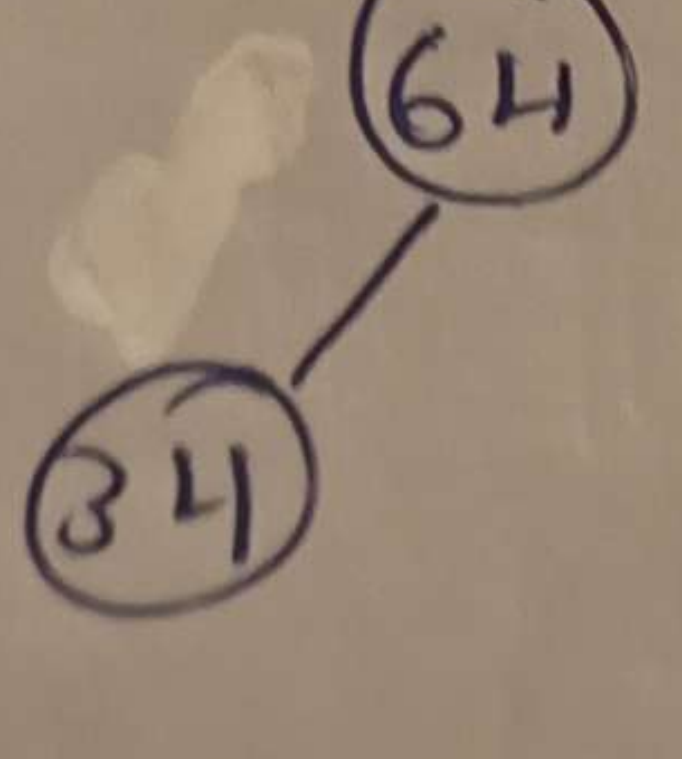


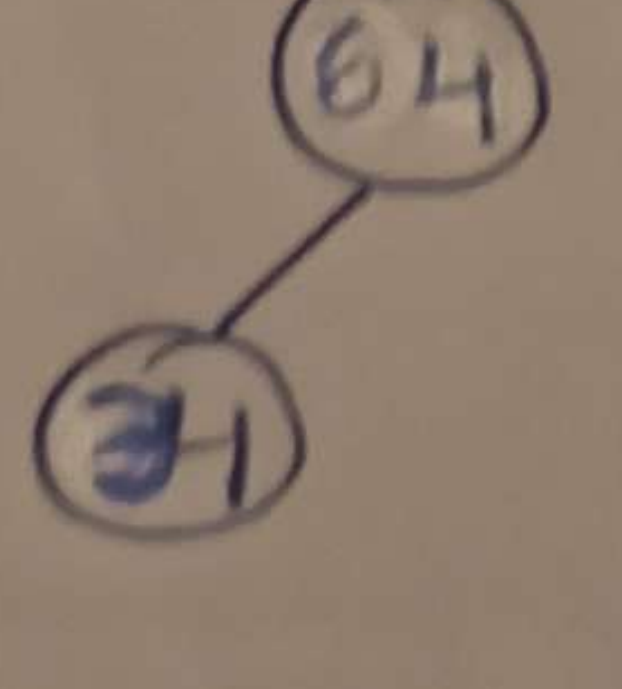
b) Sort the array using Heap sort [64,34,25,12,11,90]

Sol. Insert '64'

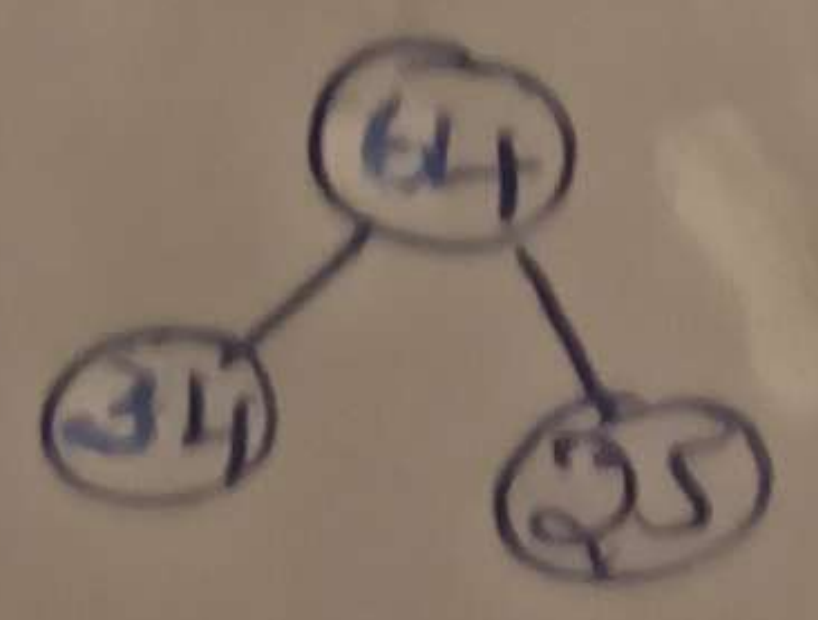


Insert '34'

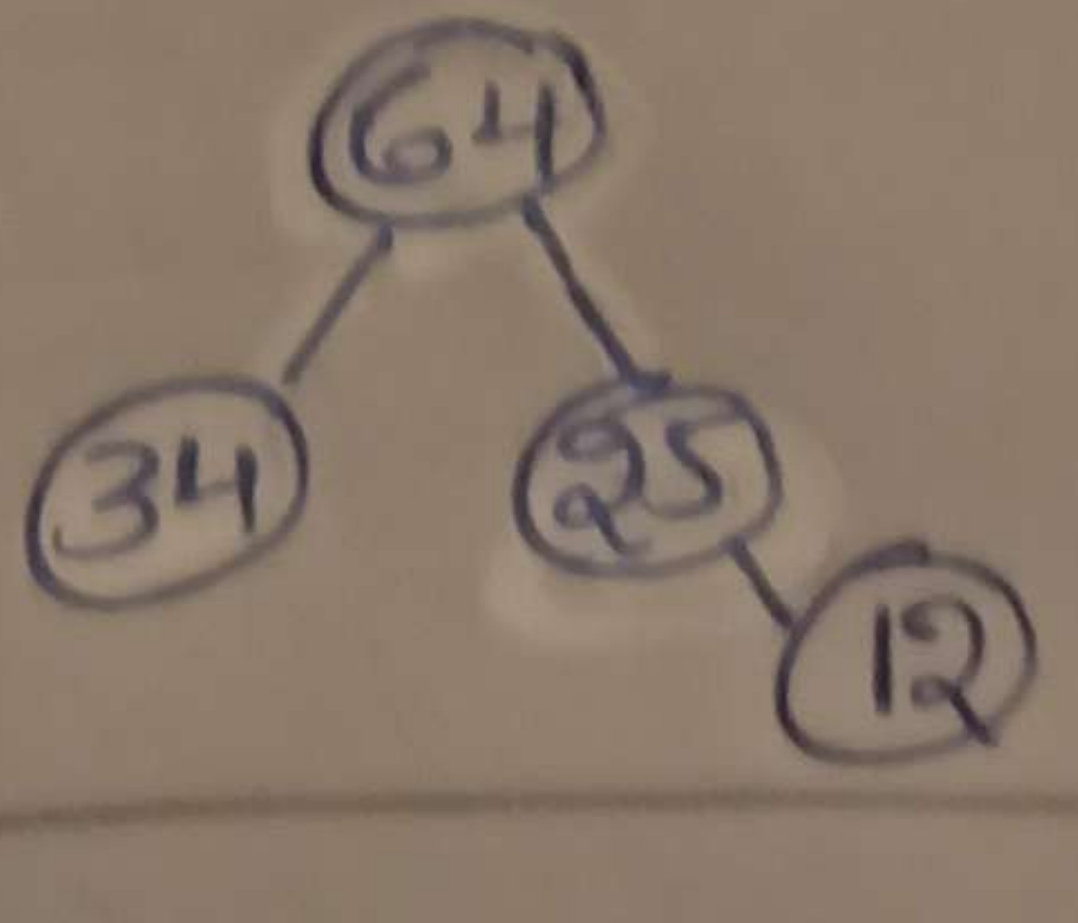




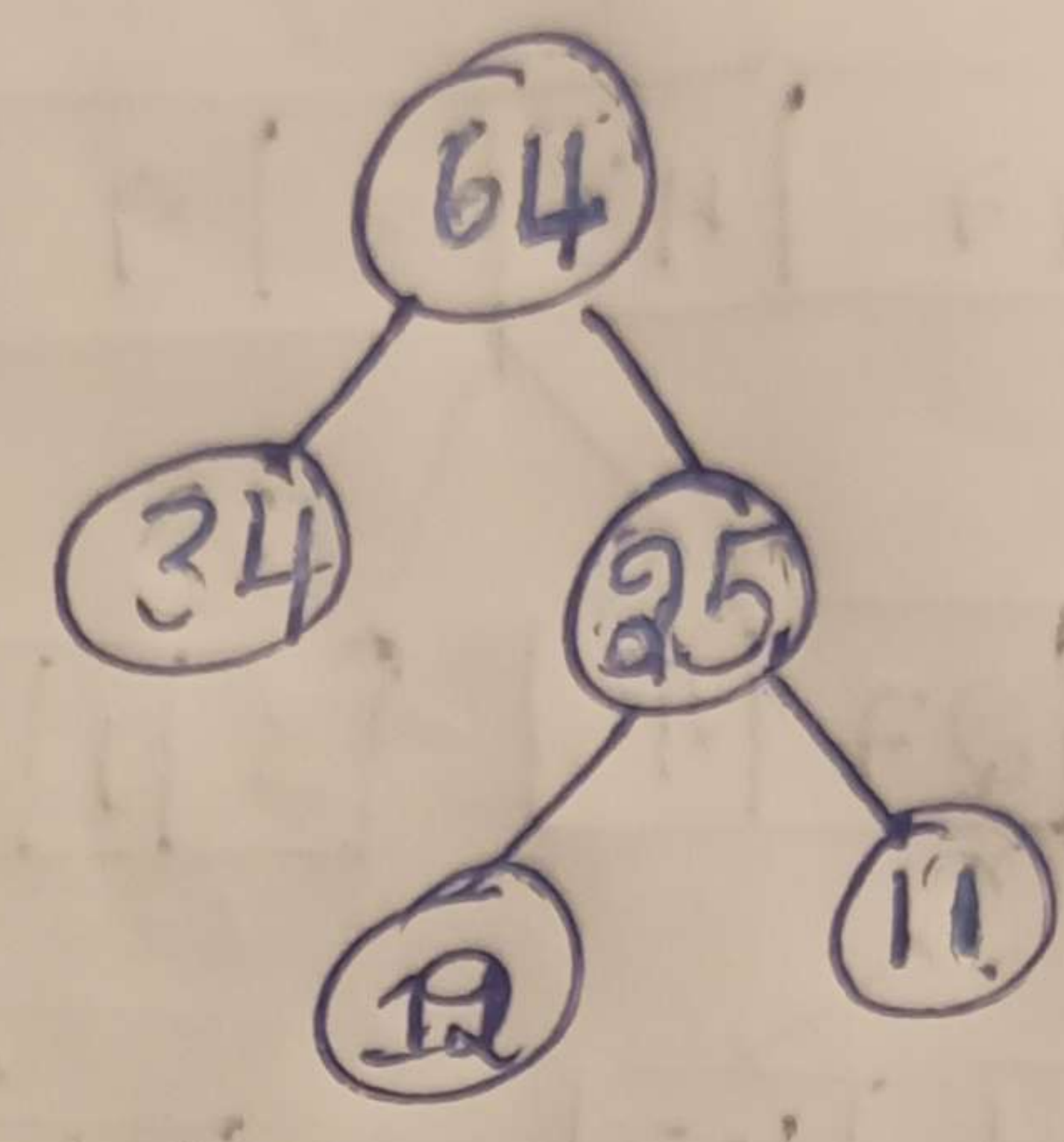
Insert '25'



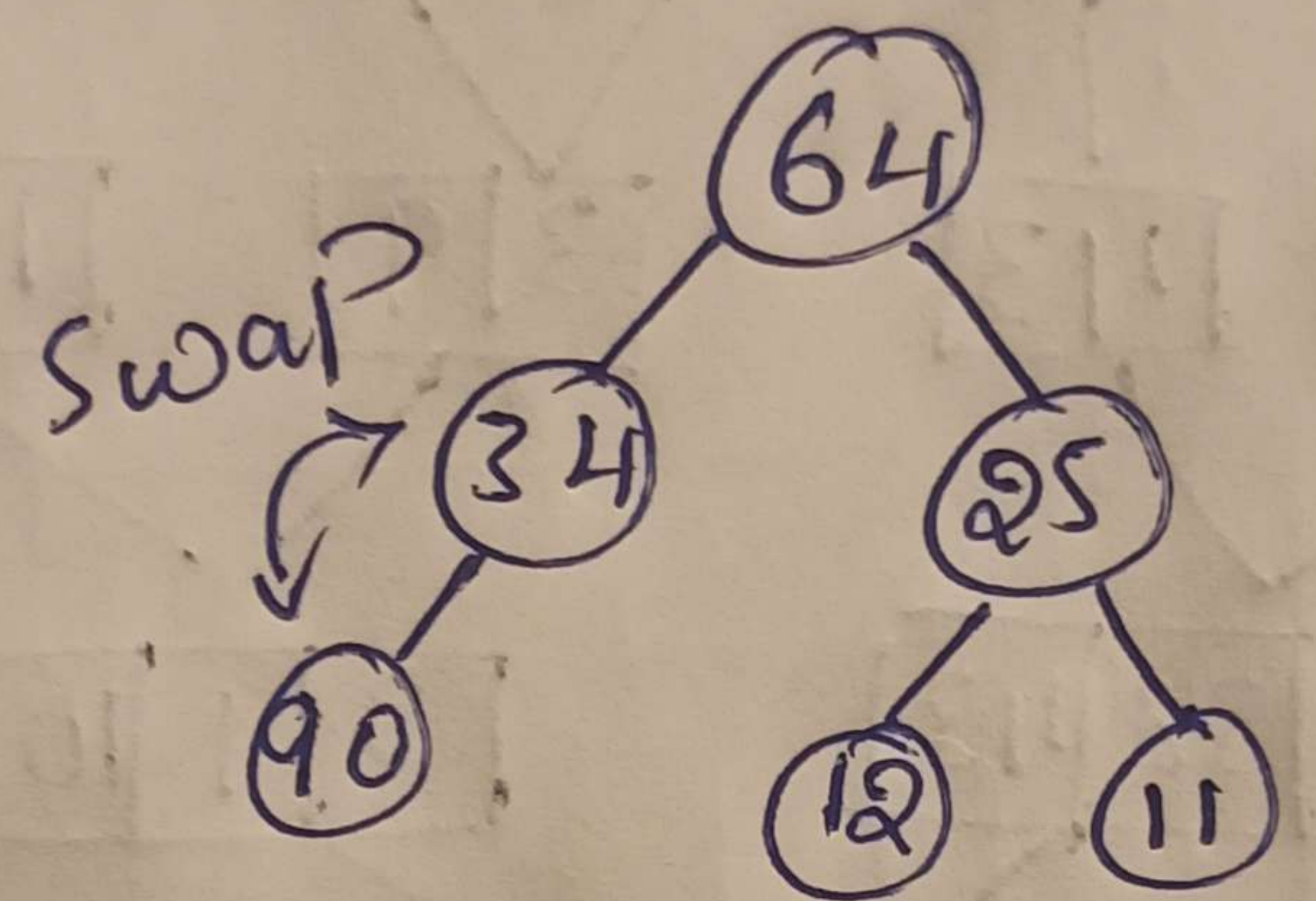
Insert '12'

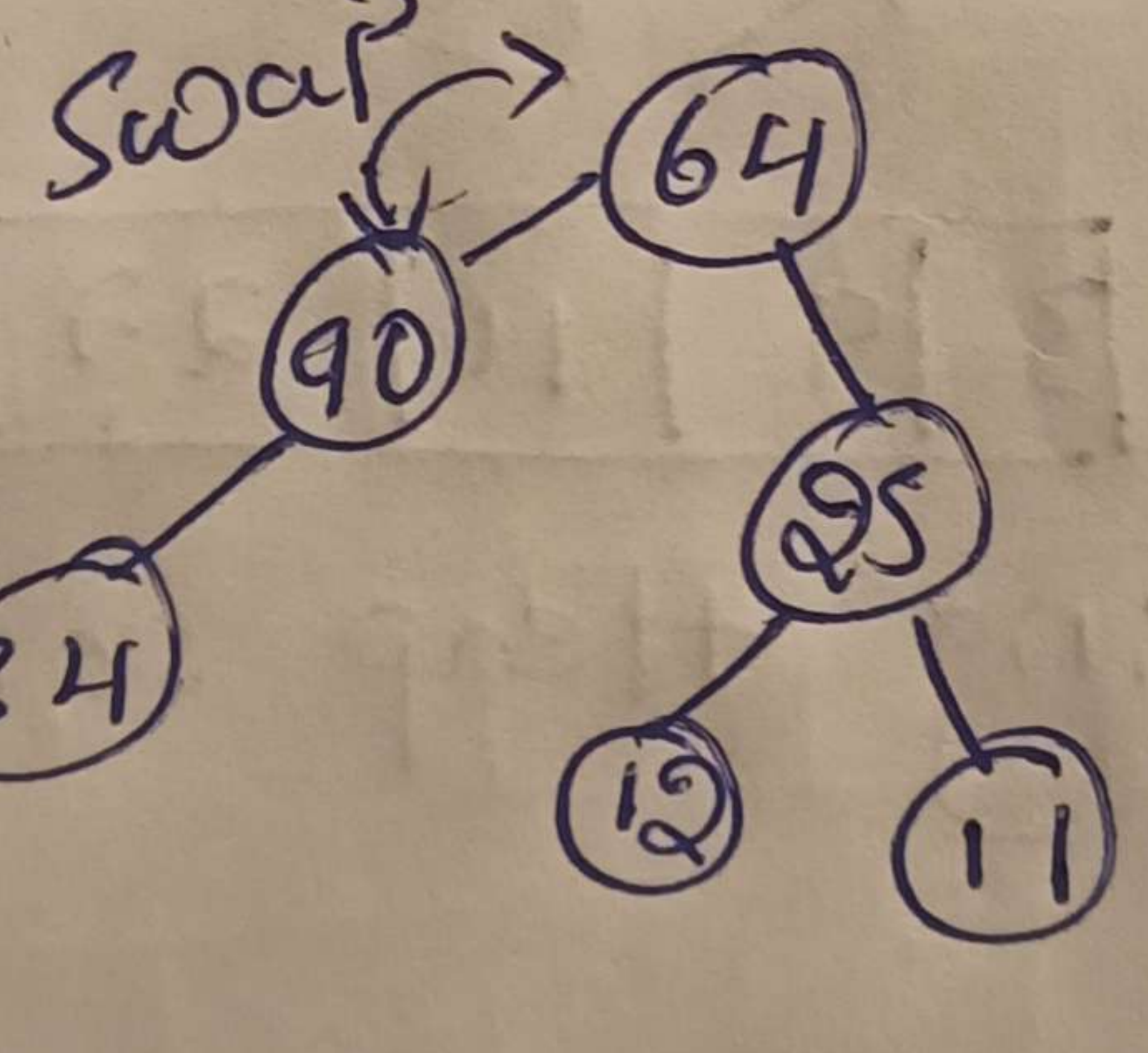


Insert '11'

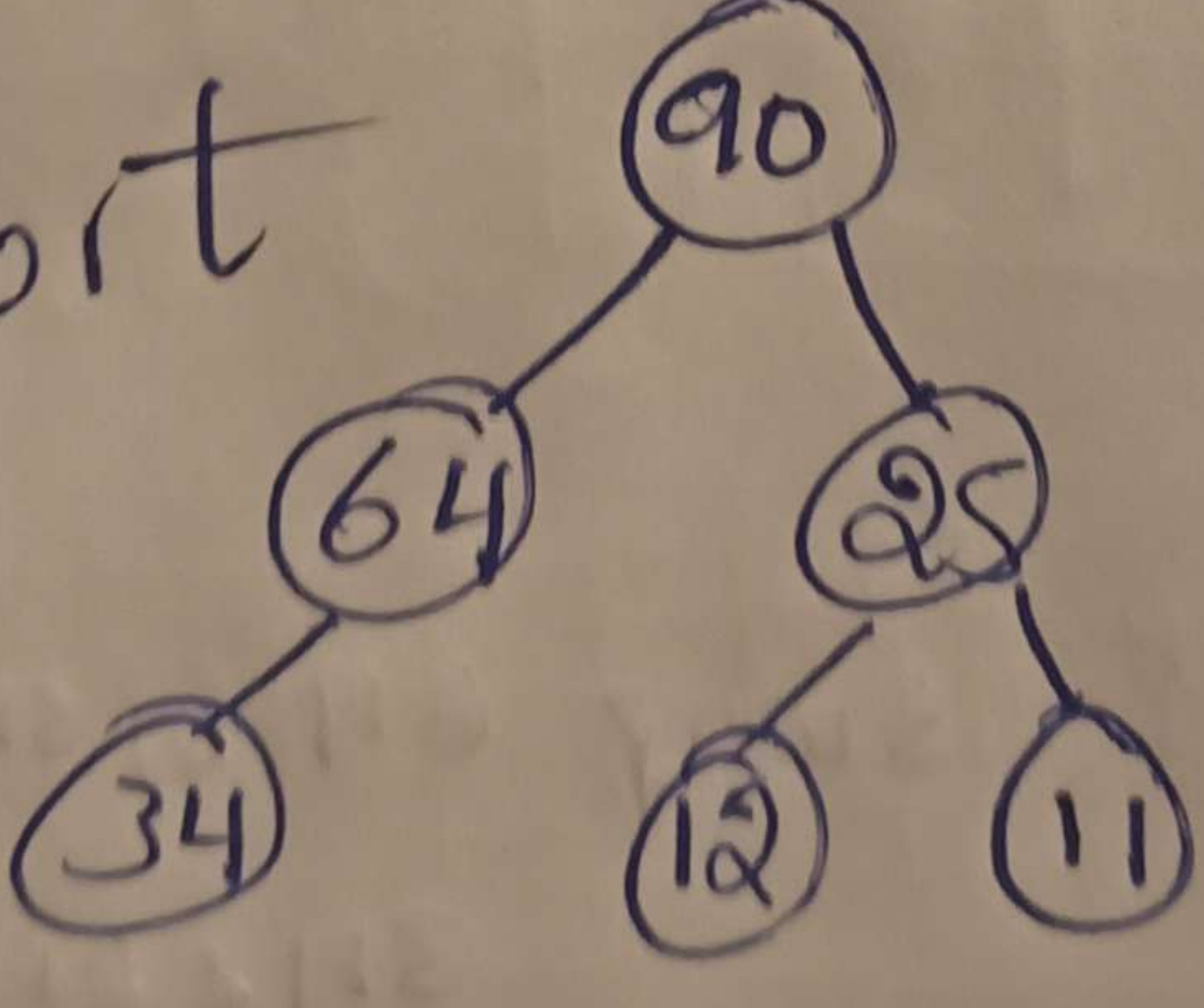


Insert '90'





Final Sort


90
64
25
34
12
11