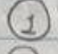
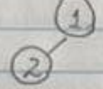
Lab 7

problem1: 1, 2, 4, 4, 5, 6, 9, 11, 12, 12, 17

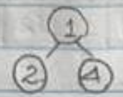
Insert 1.



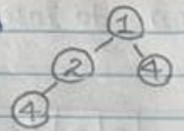
Insert 2



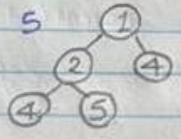
insert 4



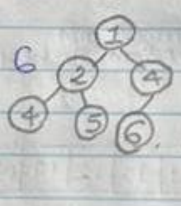
insert 4



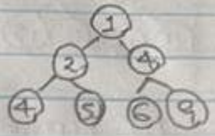
Insert 5



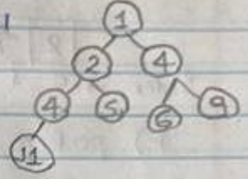
Insert 6



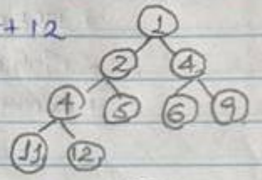
Insert 9



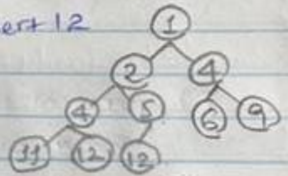
Insert 11



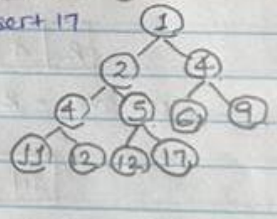
Insert 12



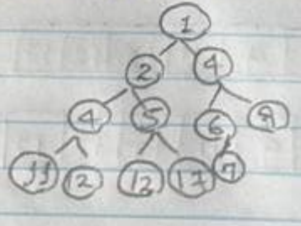
Insert 12



Insert 17

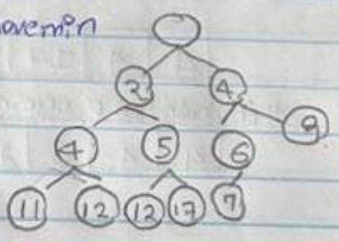


B) insert 7

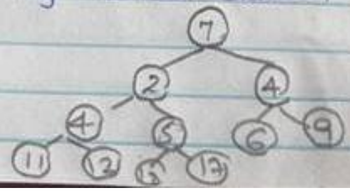


c) Show remove min

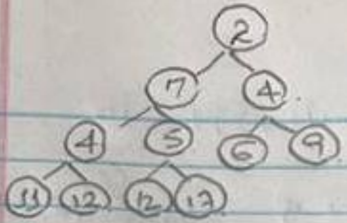
- removemin



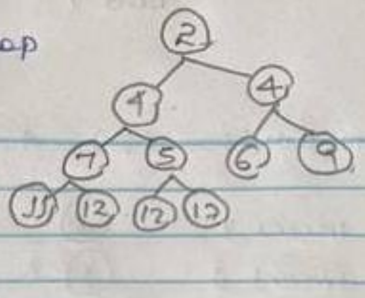
- bring the last element to pos min



- do down heap.



→ downheap



D) Represent in an Array

	2	4	4	7	5	6	9	11	12	12	17
index. →	0	1	2	3	4	5	6	7	8	9	10

E) perform Array based insertion to insert 14

- left child → $2i+1$
- righ child → $2i+2$
- parent node → $\frac{(i-1)}{2}$

0	1	2	3	4	5	6	7	8	9	10	11
2	4	4	7	5	6	9	11	12	12	17	14

parent(14) = $\frac{(11-1)}{2} = 5$

6 is smaller so ✓

F) perform Array-based ~~version of~~ remove min heapSort on the ~~input~~ array A

2	4	4	7	5	6	9	11	12	12	17	14

remove - 2

14	4	4	7	5	6	9	11	12	12	17

then do downheap

4	4	14	7	5	6	9	11	12	12	17

then down heap again

4	4	6	7	5	14	9	11	12	12	17

problem 2: sort. [1, 4, 3, 9, 12, 2, 4]

phase I : Heapification

1) 1 | 4 3 9 12 2 4

2) 1 4 | 3 9 12 2 4 → upheap 4 1 | 3 9 12 2 4

3) 4 1 3 | 9 12 2 4

4) 4 1 3 9 | 12 2 4 → upheap x 2 9 4 3 1 | 12 2 4

5) 9 4 3 1 12 | 2 4 → upheap x 2 12 9 3 1 4 | 2 4

6) 12 9 3 1 4 2 | 4 → No upheap

7) 12 9 3 1 4 2 4 → up heap 12 9 4 1 4 2 3

phase II : remove max

12

□ 9 4 1 4 2 3 → 3 9 4 1 4 2 | _

down heap

9 3 4 1 4 2 | _ $\xrightarrow{\text{downheap}}$ 9 4 4 1 3 2 | 12

9

□ 4 4 1 3 2 | 12 → 2 4 4 1 3 _ | 12

down heap

4 2 4 1 3 _ | 12 $\xrightarrow{\text{downheap}}$ 4 3 4 1 2 | 9 12

4 □ 3 4 1 2 | 9 12 → 2 3 4 1 _ | 9 12

down heap

4 3 2 1 _ | 9 12 → 4 3 2 1 | 4 9 12

4

□ 3 2 1 | 4 9 12 → 1 3 2 _ | 4 9 12

down heap

3 1 2 _ | 4 9 12 → 3 1 2 | 4 4 9 12

3

□ 1 2 | 4 4 9 12 → 2 1 _ | 4 4 9 12

2 1 | 3 4 4 9 12

2

□ 1 | 3 4 4 9 12 → 1 _ | 3 4 4 9 12

1 | 2 3 4 4 9 12

1

□ | 2 3 4 4 9 12 → No element to swap with

1 2 3 4 4 9 12 → sorted

problem 3 carry out BUH on the input sequence

11, 5, 2, 3, 17, 24, 1.

k = 11

A_1 = [5 2 3] A_2 = [17 24 11]

BUH (A_1 = [5, 2, 3])

k = 5

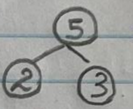
A_1 = [2] A_2 = [3]

BUH (A_1 = [2])

k = 2 A_1 = ∅   ②

BUH (A_2 = [3])   → take the 2 nodes and the key

k = 3 A_2 = ∅   ③



= T_1

Now BUH (A_2 = [17, 24, 11])

k = 17

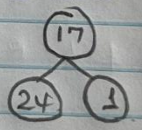
A_1 = 24 A_2 = 1

BUH (A_1 = [24])

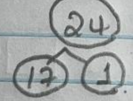
k = 24 A_1 = ∅   ㉔

BUH (A_2 = [1])   → take 2 nodes and the key

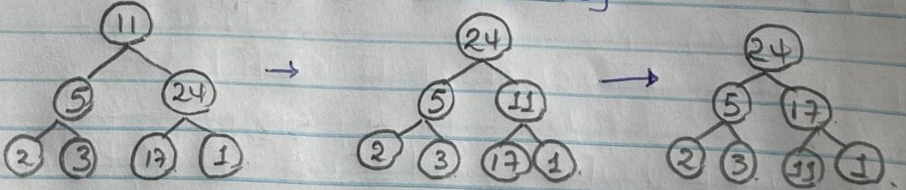
K = 1 A_2 = ∅   ①



→ down heap



take 2 trees and the key

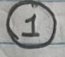


problem 4: draw Max heap Using keys [1,21]

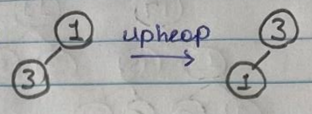
our keys will be

1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21

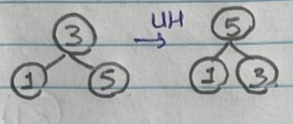
Insert 1



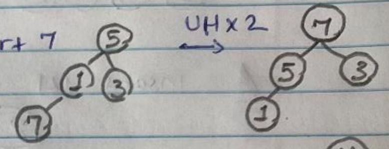
Insert 3



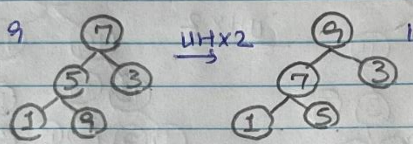
Insert 5



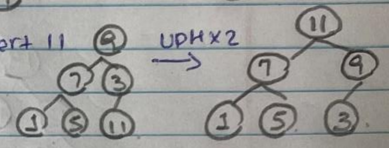
Insert 7



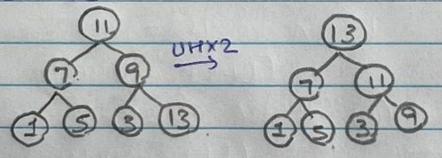
insert 9



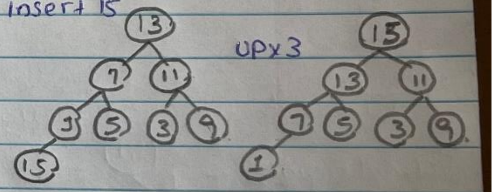
Insert 11



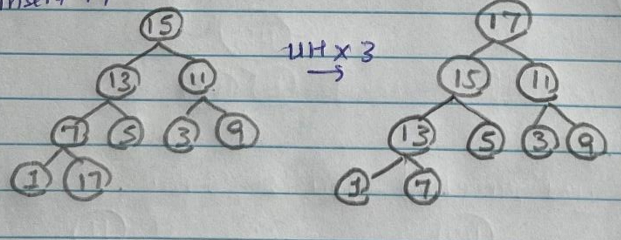
insert 13



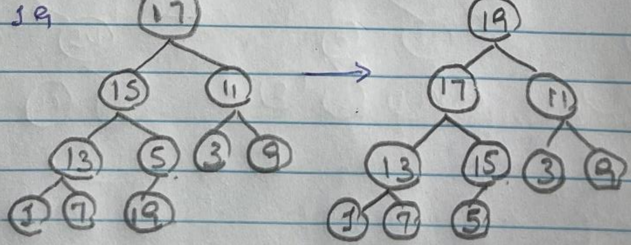
Insert 15



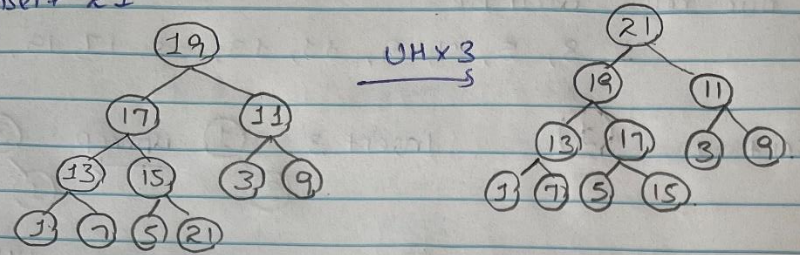
Insert 17



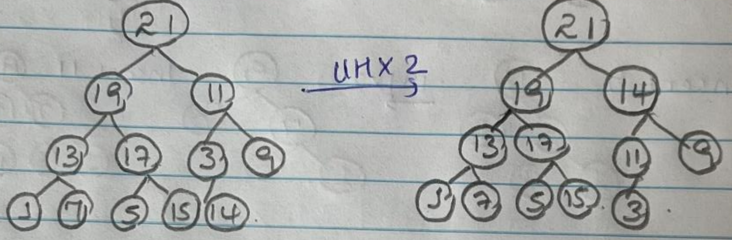
Insert 19



insert 21



Insert 14



problem 5 : Optimization BUH

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

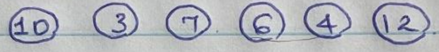
n = 11

steps: make $\frac{n+1}{2}$ single item heap tree

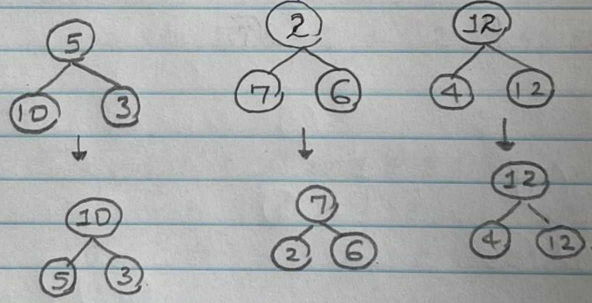
$\frac{11+1}{2} = \frac{12}{2} = 6$

10, 3, 7, 6, 4, 12

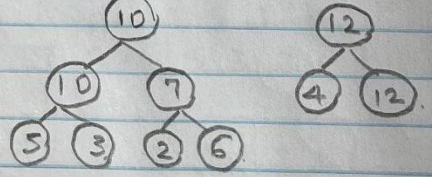
$\frac{n-1}{2}$ - reserve 5, 2, 12, 10, 1



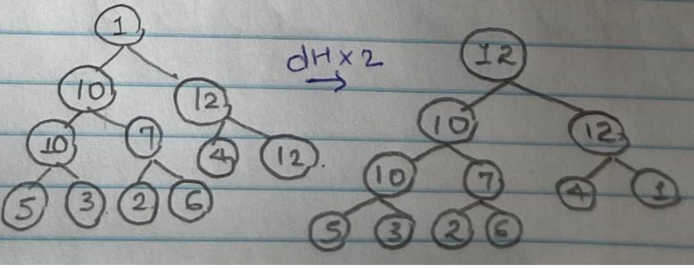
pull next $\frac{n+1}{4} = \frac{11+1}{4} = 3$



pull next $\frac{n+1}{8} = \frac{11+1}{8} = 1$



finally



the array obtained from this is

12	10	12	10	7	4	1	5	3	2	6

phase II remove max

12

□ 10 12 10 7 4 1 5 3 2 6

6 10 12 10 7 4 1 5 3 2 | _

12 10 6 10 7 4 1 5 3 2 |

12 10 7 10 6 4 1 5 3 2 | 12

12

□ 10 7 10 6 4 1 5 3 2 | 12

down heap x3 2 10 7 10 6 4 1 5 3 | _ 12

10 10 7 5 6 4 1 2 3 | 12 12

10

□ 10 7 5 6 4 1 2 3 | 12 12

downheap x3 3 10 7 5 6 4 1 2 | _ 12 12

10 6 7 5 3 4 1 2 | 10 12 12

10

□ 6 7 5 3 4 1 2 | 10 12 12

2 6 7 5 3 4 1 | _ 10 12 12

7 6 4 5 3 2 1 | 10 10 12 12

7

□ 6 4 5 3 2 1 | 10 10 12 12

1 6 4 5 3 2 | _ 10 10 12 12

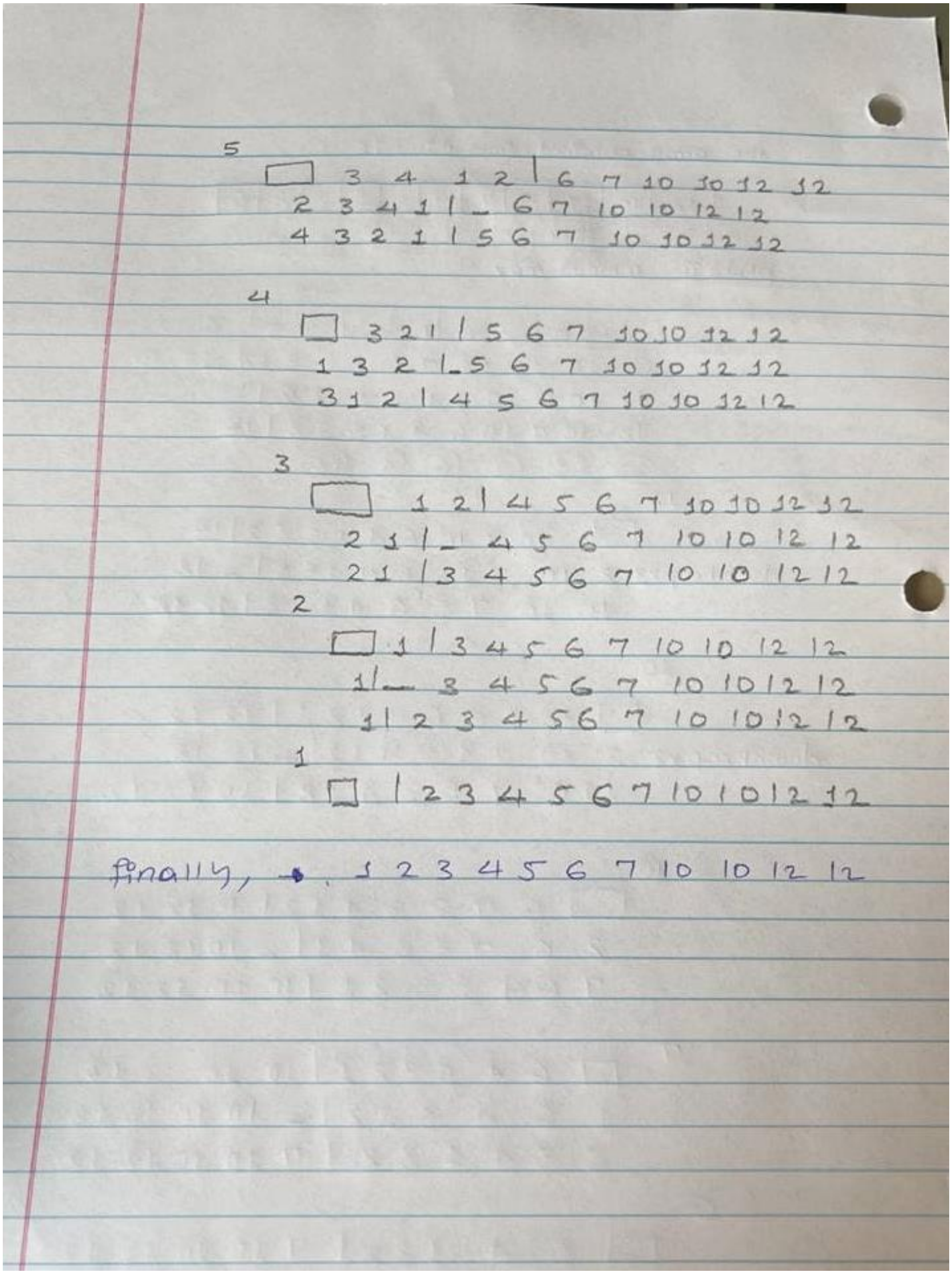
6 5 4 1 3 2 | 7 10 10 12 12

6

□ 5 4 1 3 2 | 7 10 10 12 12

2 5 4 1 3 | _ 7 10 10 12 12

5 3 4 1 2 | 6 7 10 10 12 12



5

□ 3 4 1 2 | 6 7 10 10 12 12

2 3 4 1 | _ 6 7 10 10 12 12

4 3 2 1 | 5 6 7 10 10 12 12

4

□ 3 2 1 | 5 6 7 10 10 12 12

1 3 2 | _ 5 6 7 10 10 12 12

3 1 2 | 4 5 6 7 10 10 12 12

3

□ 1 2 | 4 5 6 7 10 10 12 12

2 1 | _ 4 5 6 7 10 10 12 12

2 1 | 3 4 5 6 7 10 10 12 12

2

□ 1 | 3 4 5 6 7 10 10 12 12

1 | _ 3 4 5 6 7 10 10 12 12

1 | 2 3 4 5 6 7 10 10 12 12

1

□ | 2 3 4 5 6 7 10 10 12 12

finally, • 1 2 3 4 5 6 7 10 10 12 12