Program Name: Master of Computer Applications
Semester: 2
Paper Title: Data Structures

Submitted by: Anurag Joshi
Examination roll number: 19234757007
Department roll Number: 07

Data Structures Assignments

Assignments :

1. Linked List (Assignment-1)
2. Heap Questions (Assignment-2)
3. Hash Maps and Binary Search Tree (Assignment-3)
4. AVL Tree (Assignment -4)
5. Programming BST, Heap, AVL Tree, etc (Assignment-5)

Assignment-1

Q1. Single Linked List

Output:

```
MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 1

Enter the info: 3

Element added.
Linked list: 3 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 2

Enter the info: 8
```

```
Element added.
Linked list: 3 -> 8 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 3

Enter the index of the new element: 1

Enter the info: 4

Linked list: 3 -> 4 -> 8 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 7

List is reversed.
```

```
Linked list: 8 -> 4 -> 3 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 8

Enter ele to search: 4

Element found at index 1 is swapped with its previous element.

Linked list: 4 -> 8 -> 3 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 6

Enter the index of the element to be deleted: 1

Element deleted.
```

```
Linked list: 4 -> 3 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 4

Linked list: 3 ->  NULL

MENU::
1)Add ele to head.
2)Add ele to tail.
3)Add ele in a position.
4)Delete ele from head.
5)Delete ele from tail.
6)Delete ele from a position.
7)Reverse list.
8)Search element and Swap.
9)Exit.

Enter your choice: 5

Linked list: Empty
```

Q2. Doubly Linked List

Output:

```
MAIN MENU
1. Add to node to head                  2. Add a node to tail
3. Print the list                       4. Print the list in reverse
5. Search for an element                6. Delete a node from head
7. Delete a node from tail              8. Delete a node from a position
9. Delete a node with a given value     10. Add a node at a given position
11. Exit
Enter your choice: 4

Reversed list: 3 5 7
Program for Unordered Double Linked List
MAIN MENU
1. Add to node to head                  2. Add a node to tail
3. Print the list                       4. Print the list in reverse
5. Search for an element                6. Delete a node from head
7. Delete a node from tail              8. Delete a node from a position
9. Delete a node with a given value     10. Add a node at a given position
11. Exit
Enter your choice: 5

Enter the element to search: 7
Element found

Program for Unordered Double Linked List
MAIN MENU
1. Add to node to head                  2. Add a node to tail
3. Print the list                       4. Print the list in reverse
5. Search for an element                6. Delete a node from head
7. Delete a node from tail              8. Delete a node from a position
9. Delete a node with a given value     10. Add a node at a given position
11. Exit
Enter your choice: 9
```

```
Enter the value of the node to be deleted: 5
List: 7 3

Program for Unordered Double Linked List
MAIN MENU
1. Add to node to head                  2. Add a node to tail
3. Print the list                       4. Print the list in reverse
5. Search for an element                6. Delete a node from head
7. Delete a node from tail              8. Delete a node from a position
9. Delete a node with a given value     10. Add a node at a given position
11. Exit
Enter your choice: 6

Element deleted from head
List: 3

Program for Unordered Double Linked List
MAIN MENU
1. Add to node to head                  2. Add a node to tail
3. Print the list                       4. Print the list in reverse
5. Search for an element                6. Delete a node from head
7. Delete a node from tail              8. Delete a node from a position
9. Delete a node with a given value     10. Add a node at a given position
11. Exit
Enter your choice: 7

Element deleted from tail
List: List is empty.
```

Q3. Circular Linked List

Output:

```
MENU::
1)Add node
2)Remove node
3)Print Front
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 1

Enter the info: 4

Circular List-> 4

MENU::
1)Add node
2)Remove node
3)Print Front
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 1

Enter the info: 6

Circular List-> 6  4

MENU::
1)Add node
2)Remove node
3)Print Front
```

```
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 1

Enter the info: 8

Circular List-> 8  6  4

MENU::
1)Add node
2)Remove node
3)Print Front
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 3

Front element is 8

MENU::
1)Add node
2)Remove node
3)Print Front
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 4

Back element is 4
```

```
MENU::
1)Add node
2)Remove node
3)Print Front
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 2

Circular List-> 6  4

MENU::
1)Add node
2)Remove node
3)Print Front
4)Print Back
5)Advance
6)Traverse list
7)Exit.

Enter your choice: 5

Circular List-> 4  6
```

Q4. Rearrange the given array such that even numbers come before odd numbers.

Output:

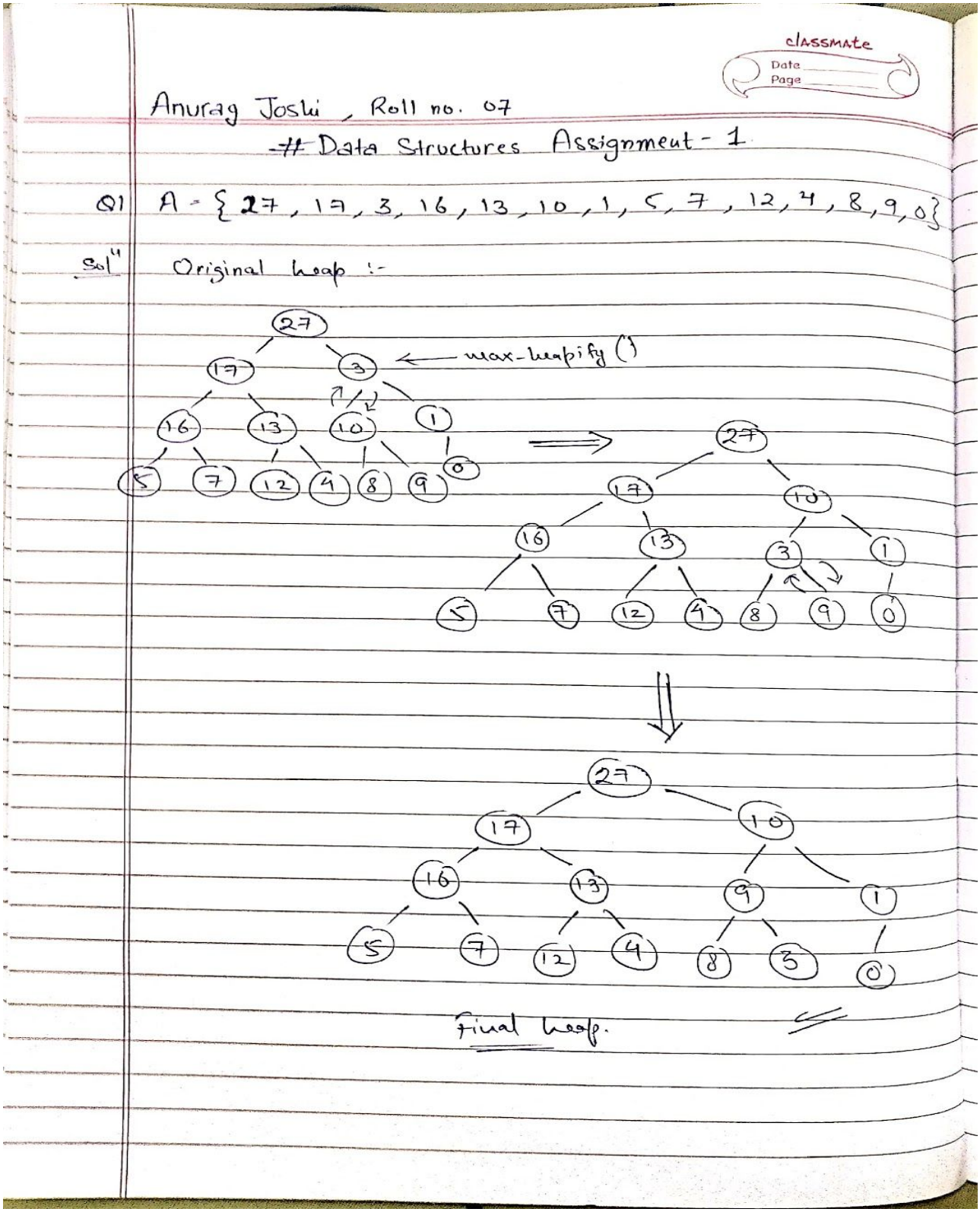
```
Enter Size of the array: 10

Enter the elements of the array: 6 3 9 7 1 2 4 0 5 8

Rearranged array: 6 8 0 4 2 1 7 5 9 3
```

Q5. Reverse String Using Recursion.

Output:

```
Enter the string: We shall overcome one day

Reversed String: yad eno emocrevo llahs eW
```

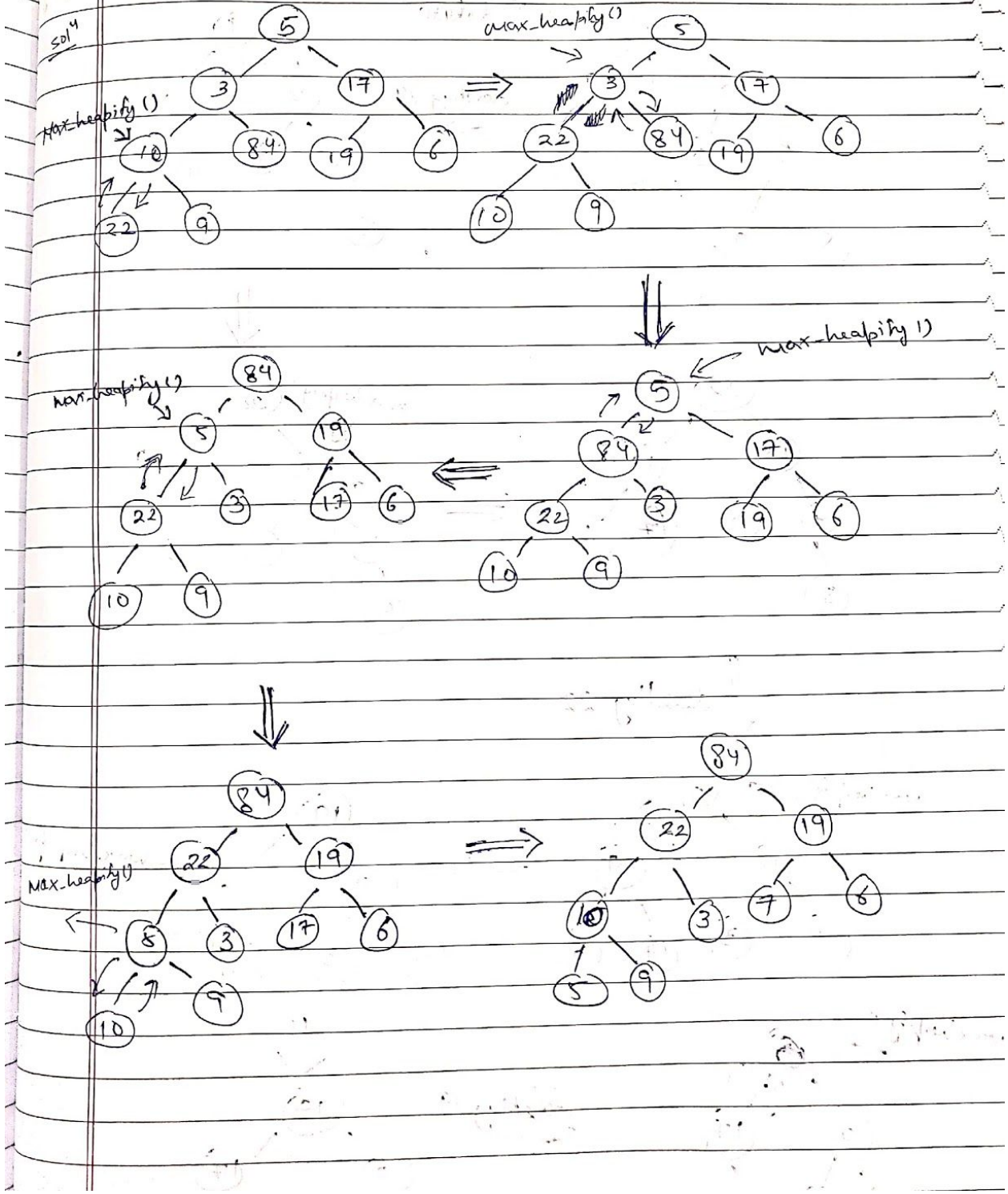
Assignment-2



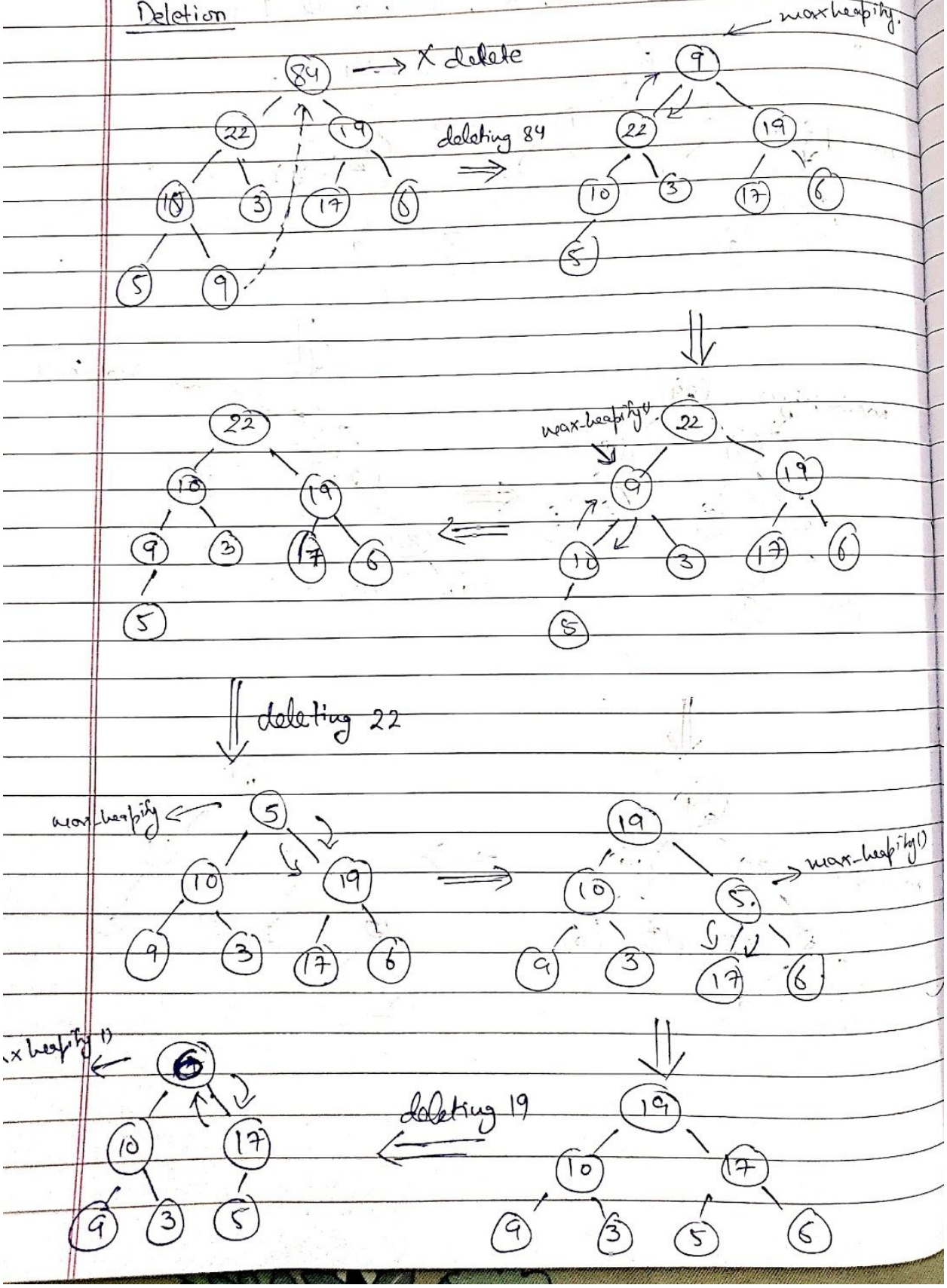
Page

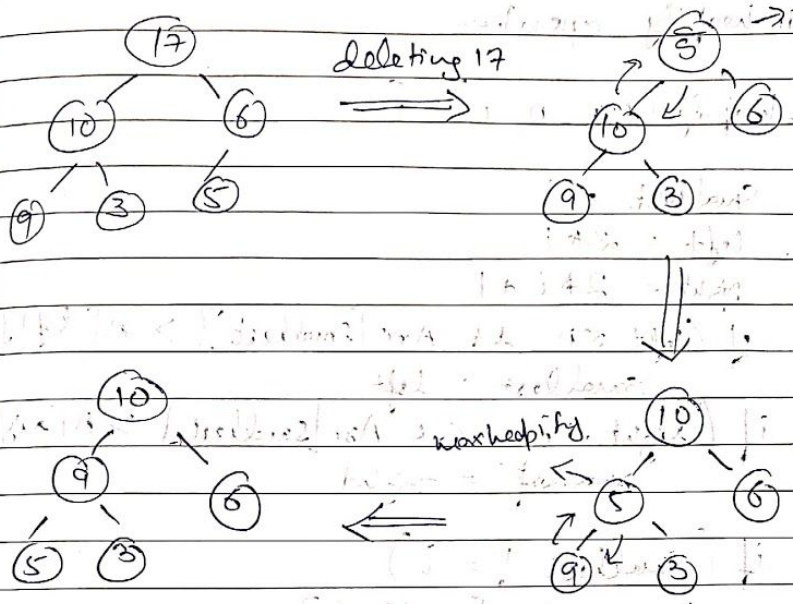
Q2. A = { 5, 3, 17, 10, 84, 19, 6, 22, 9 }

Sol^n

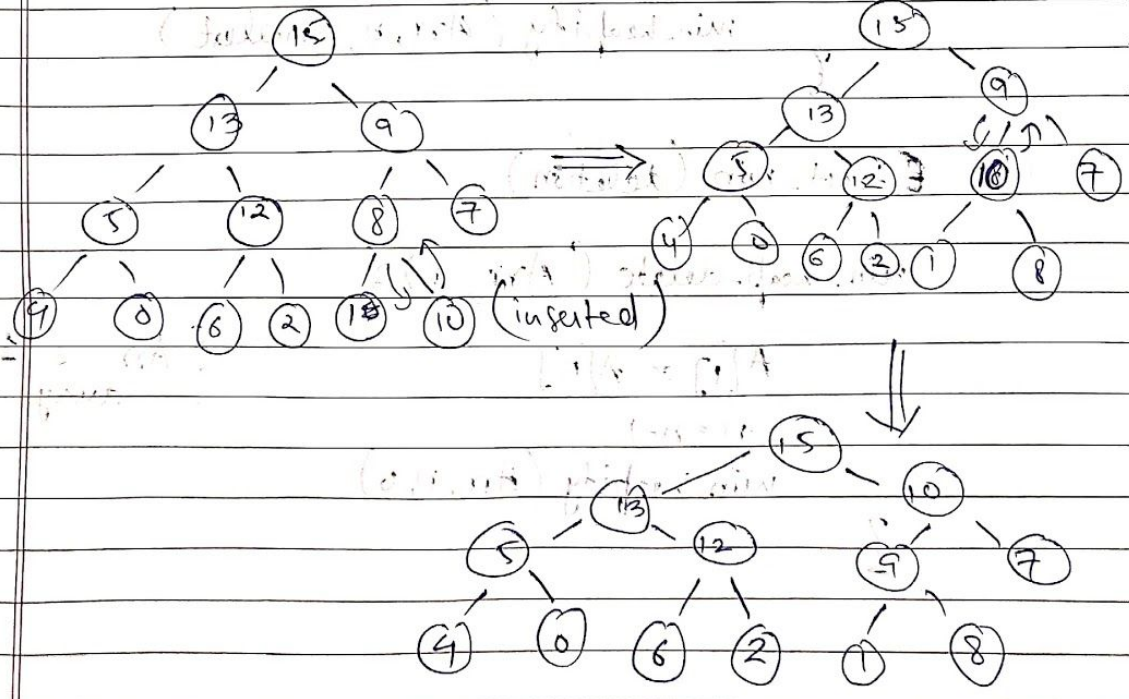


Deletion


max heapify
84 → X delete
deleting 84
22
19
10
3
17
6
5
9
max-heapify
deleting 22
max_heapify
max_heapify()
deleting 19



Q3. A = {15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1}



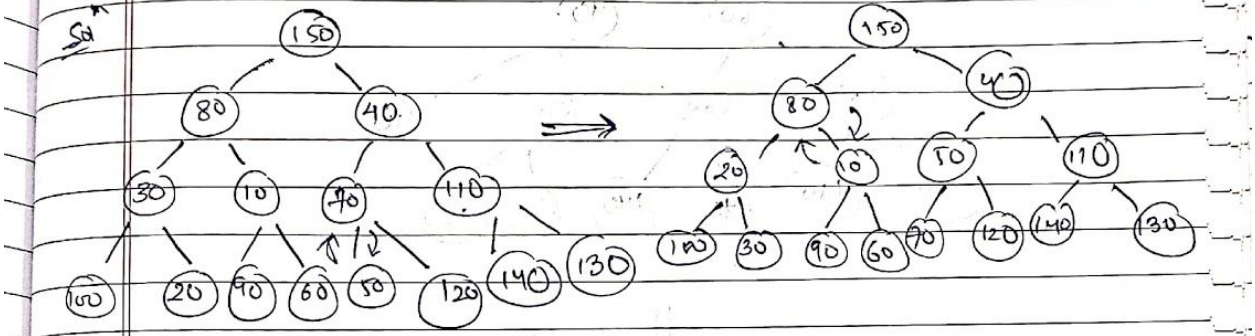
Q4. (a) Min heapify operation

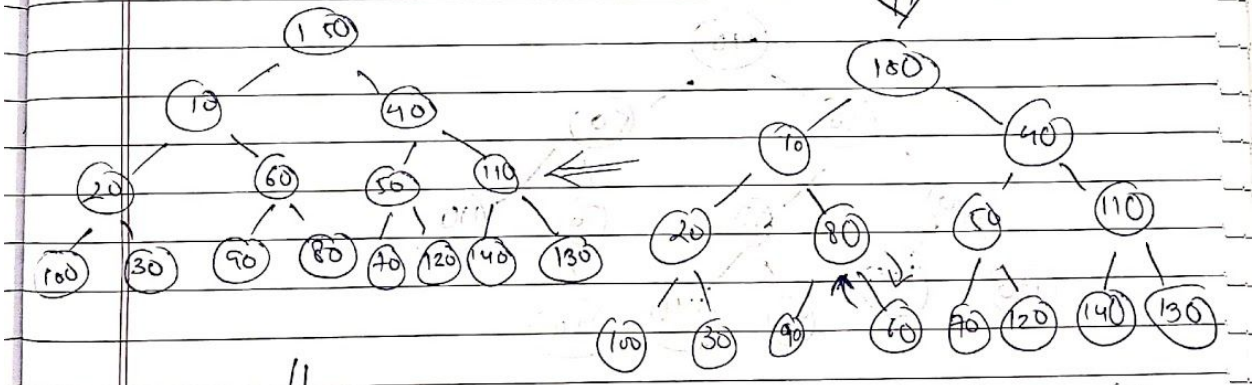
```
min_heapify (Arr, n, i) {

    Smallest = i
    left = 2*i
    right = 2*i + 1
    if (left <= n && Arr[smallest] > A[left])
        smallest = left
    if (right <= n && Arr[smallest] > A[right])
        smallest = right

    if (Smallest != i)
        temp = Arr[smallest]
        Arr[i] = temp
        Arr[smallest] = Arr[i]
        Arr[i] = temp

        min_heapify (Arr, n, smallest)
    }
```

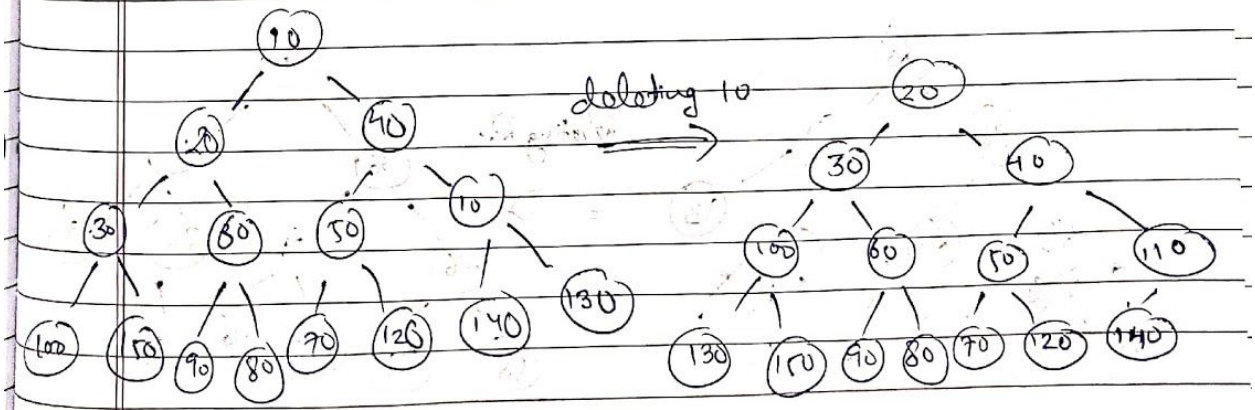
(b) Extract_min (deletion)

```
min_heap_delete (Arr, n) {

    A[1] = A[n]          { Arr is 1 index away }
    n = n-1
    min_heapify (Arr, n, 0)
}
```

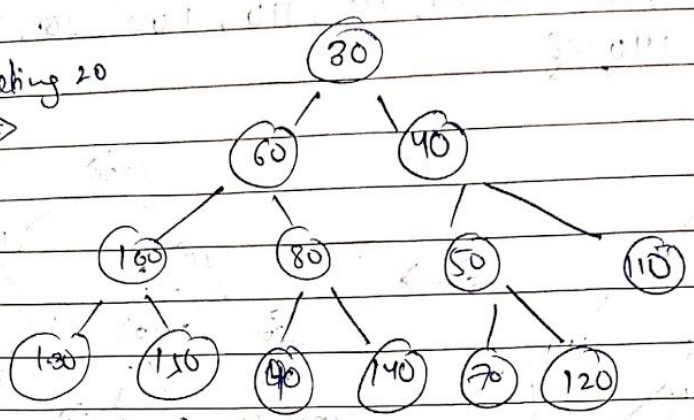
Q: A = {150, 80, 40, 30, 10, 70, 110, 100, 20, 90, 60, 50, 120, 140, 130}



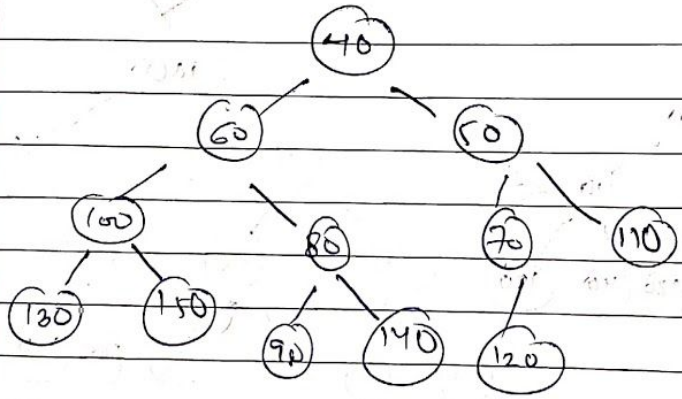




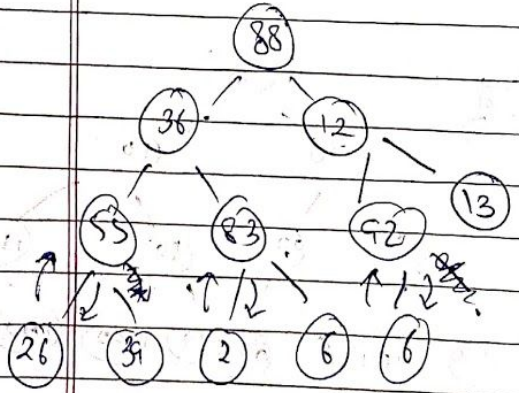
deleting 20



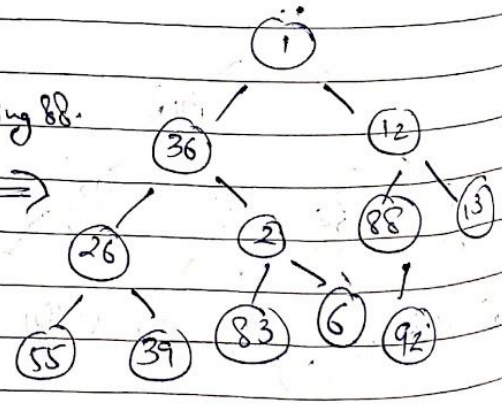
deleting 30

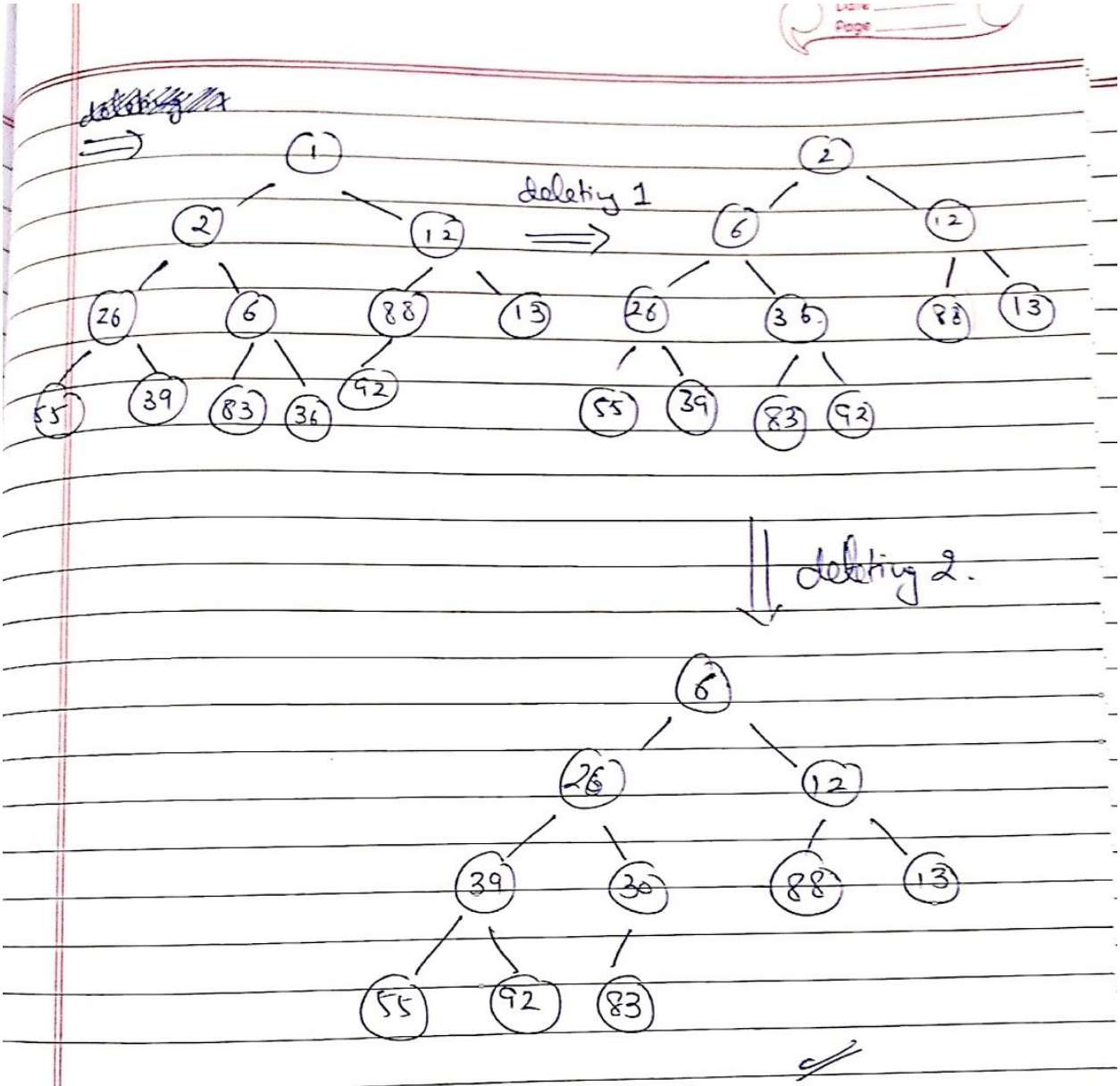


Q6. A = { 88, 36, 12, 55, 83, 92, 13, 26, 39, 2, 6, 1 }



deleting 88.




deleting 1
deleting 2.

Assignment-3

Submitted by : Anurag Joshi

Data Structure Assignment – 3

Q1. Sol^n

20	1	8		12	25	4	6	18	3
0	1	2	3	4	5	6	7	8	9

Length = 10

keys	H (i)
1	1 % 10 = 1
3	9 % 10 = 9
12	144 % 10 = 4
4	16 % 10 = 6
25	625 % 10 = 5
6	36 % 10 = 6 (36+1) % 10 = 7
18	324 % 10 = 4 ⇒ 4, 5, 6, 7 occupied ⇒ 8
20	400 % 10 = 0
8	64 % = 4 ⇒ (4+8) mod 10 = 2

For 8 upto k=7 no place is empty that is why k=8

$(H(i) + k) \% 10 = (4+8) \% 10$

$= 12 \% 10 = 2$

Maximum probe value ⇒ 8.

Q2.

Sol^n Hash table given below :-

40	91	52	23		60	50			62
0	1	2	3	4	5	6	7	8	9

K	$H_1(k)$	$H_2(k)$	$DH(k)$ if needed.			
			i=0	i=1	i=2	i=3
23	3	8	3			
91	1	4	1			
52	2	5	2			
40	0	1	0			
50	0	3	0	3	6	
60	0	5	0	5		
63	2	7	2	9		

No. of collision ⇒ 3 (i=0), 1 (i=1)

for 50, i=0 creates collision.

i=1 DH(k) = (0+1*3) %. 10 = 3

i=2 DH(k) = (0+2*3) %. 10 = 6

For 60, i=0 (creates Collision

i=1, (6+1×5) %10 = 5

for 62, i=0 (collision)

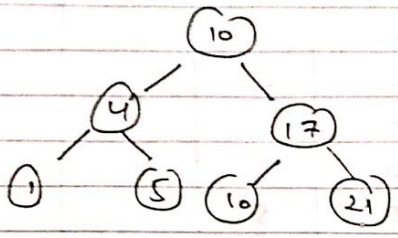
i=1 DH(k) = (2+7) % 10 = 9 %. 10 = 9

No. of collisions = 4

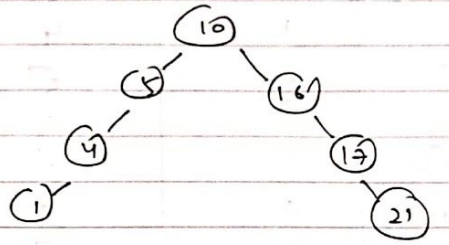
Q3

Sol:

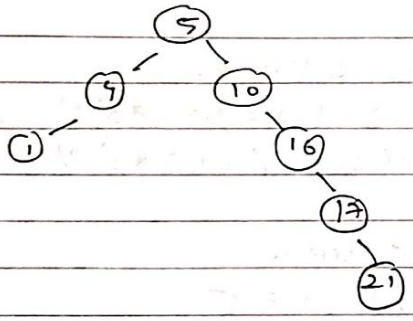
BST of Height = 2 :-



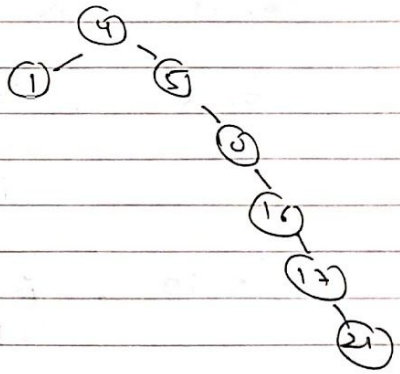
BST Height = 3.



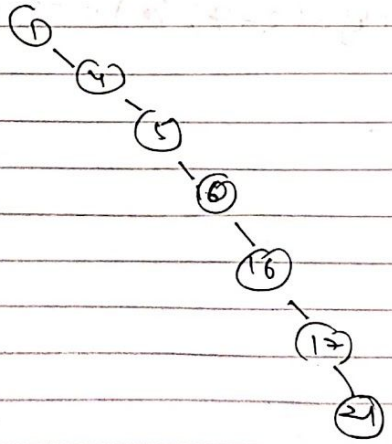
BST of Height = 4 :-



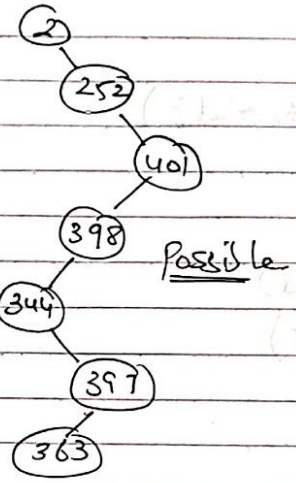
BST Height = 5



BST of Height = 6 :

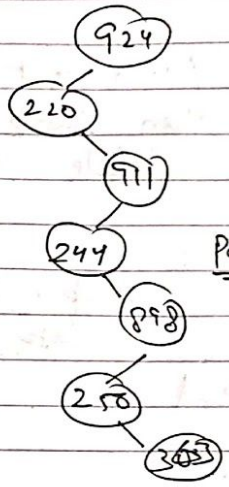


Q4. Soln a)



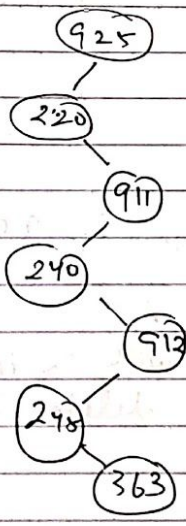
Possible

b)



Possible.

c)



This sequence can't be possible

912 > 911 So

It cannot be at left of 911

=

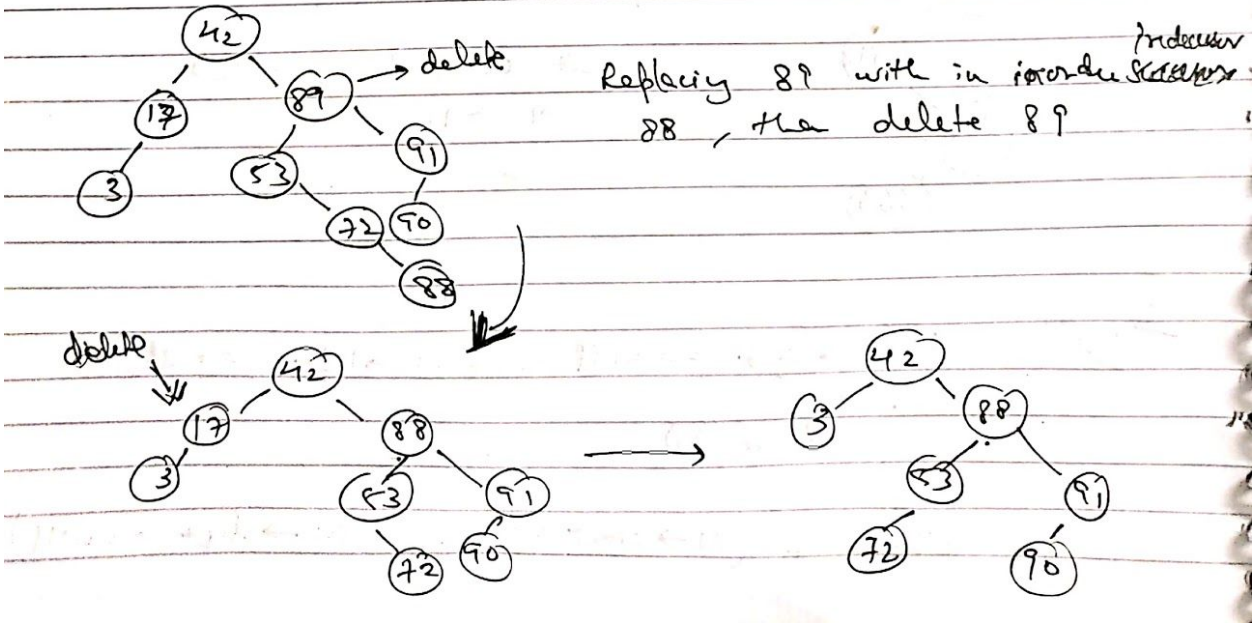
Q5.

Soln

```
if (N -> right == null && N -> left == null)
then
    delete(N)
end
else if (N -> right == null || N -> left == null)
```

```
then
    if (N -> right = null)
      then Swap (N, N -> left)
        delete N -> left

    else
        Swap (N, N -> right)
        delete (N -> right)
    end

else
    S(T) = Inorder_successor (N);
    Swap (N, S(T));
    delete (S(T));
end
```

Q6-sol^{n} 42, 17, 89, 53, 72, 91, 3, 88, 90



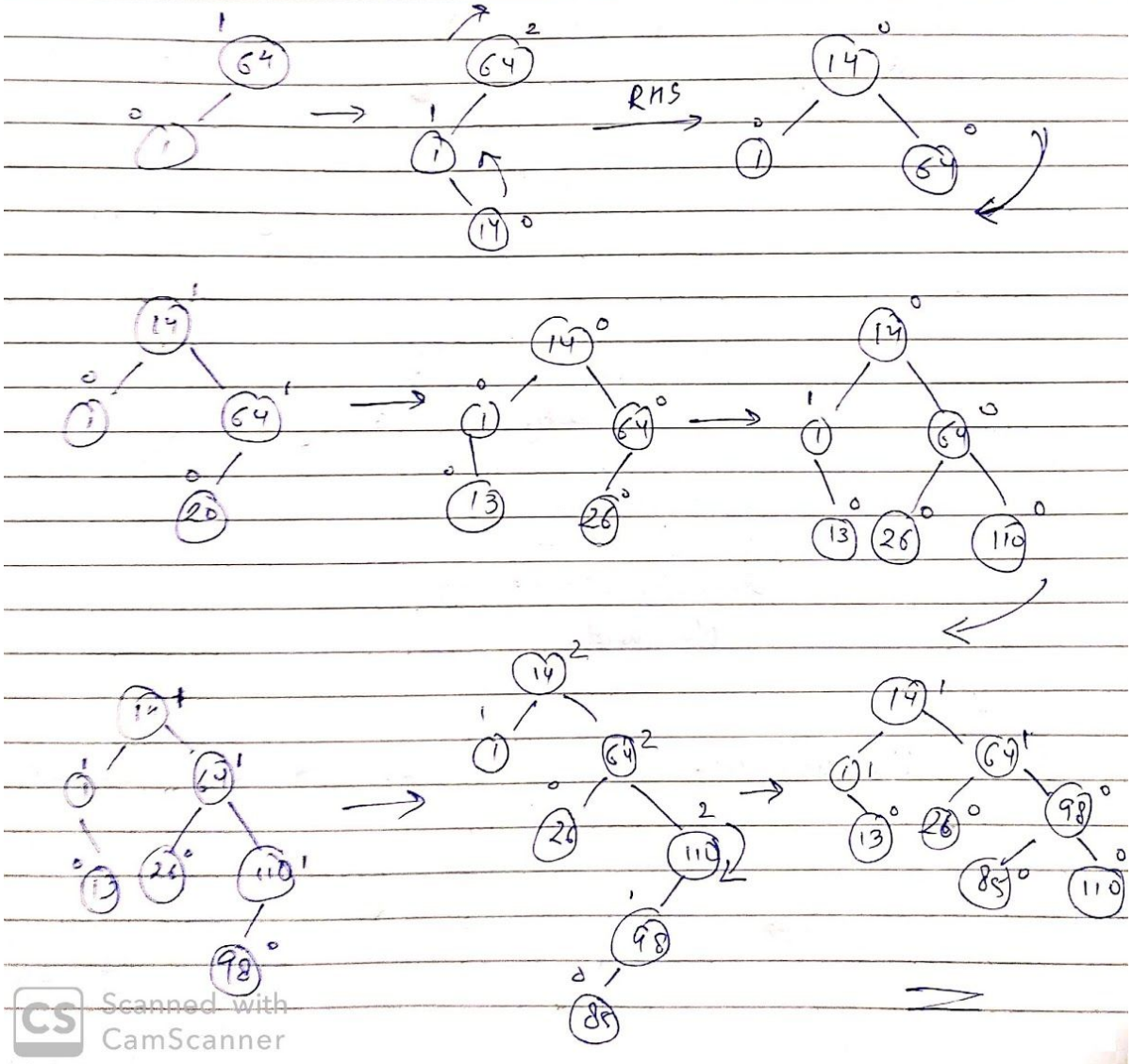
Assignment-4

Data Structures Assignment- 4

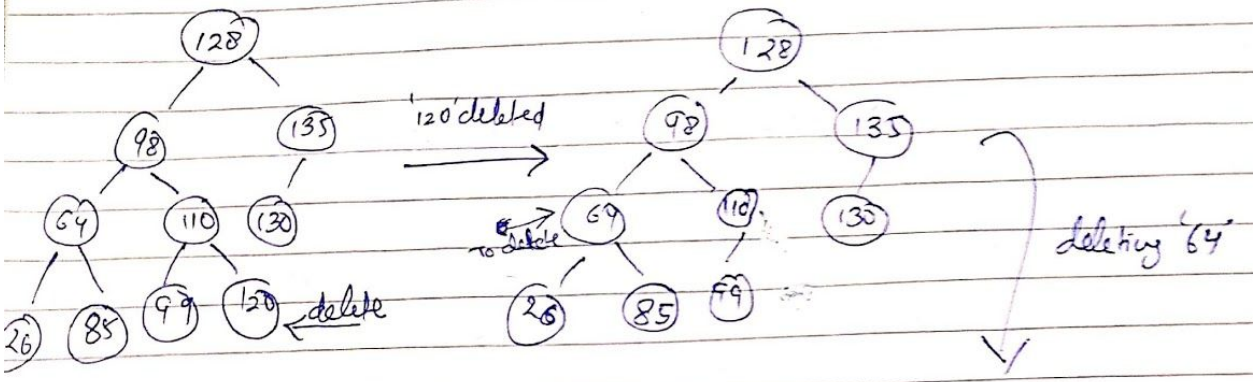
Submitted by: Anurag Joshi
Roll no: 07

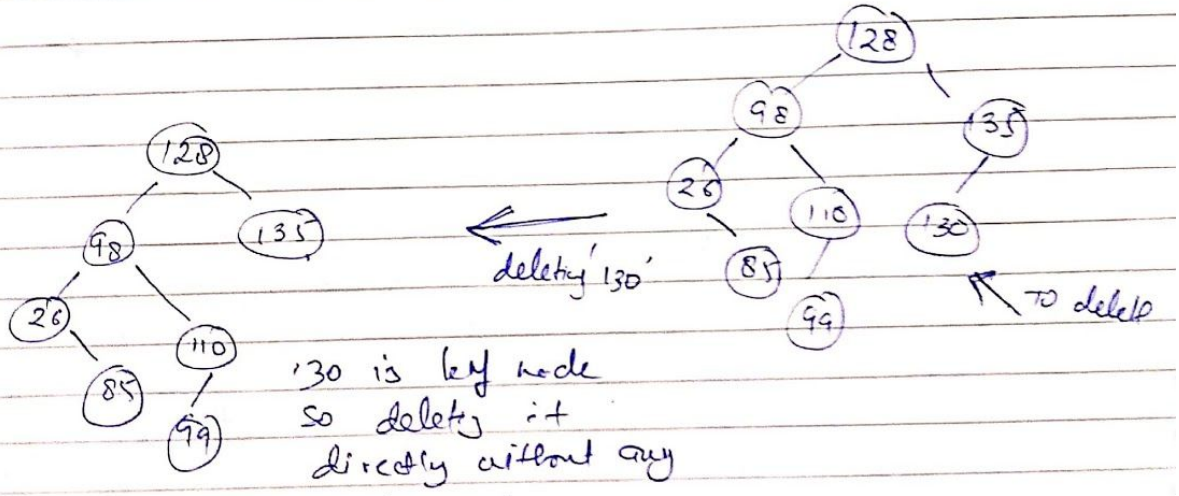
Q1.

Sol: 64, 1, 14, 26, 13, 110, 98, 85



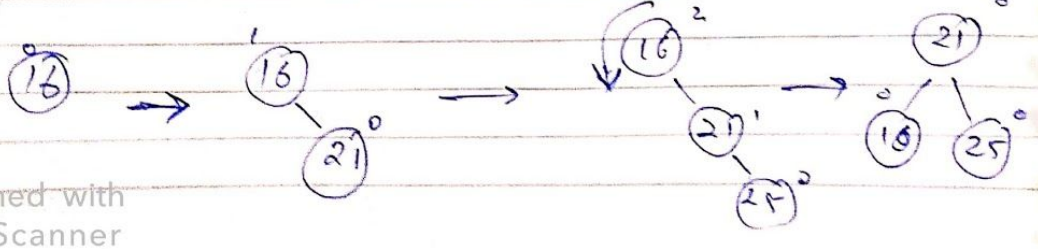
Q2. Soln To Delete 120, 64 & 130

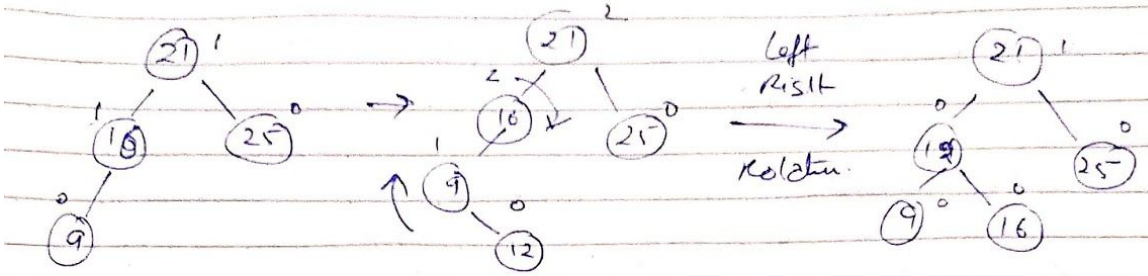


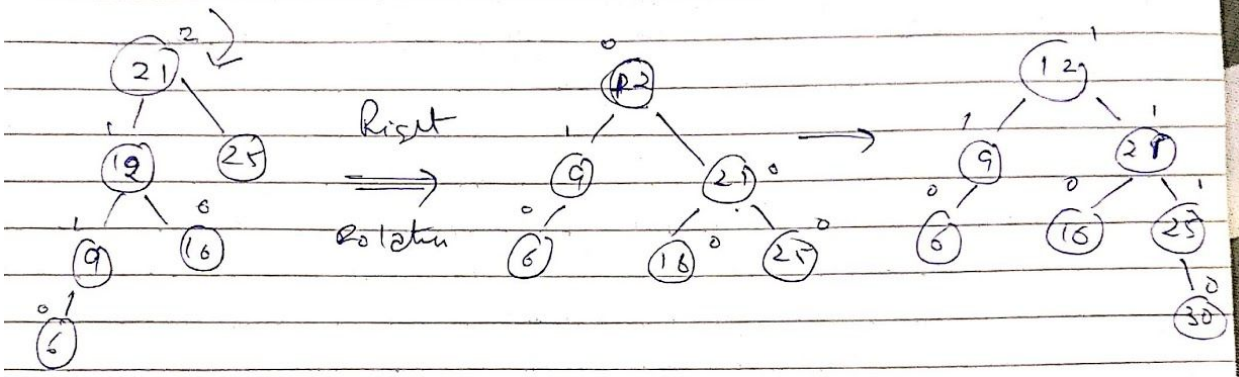


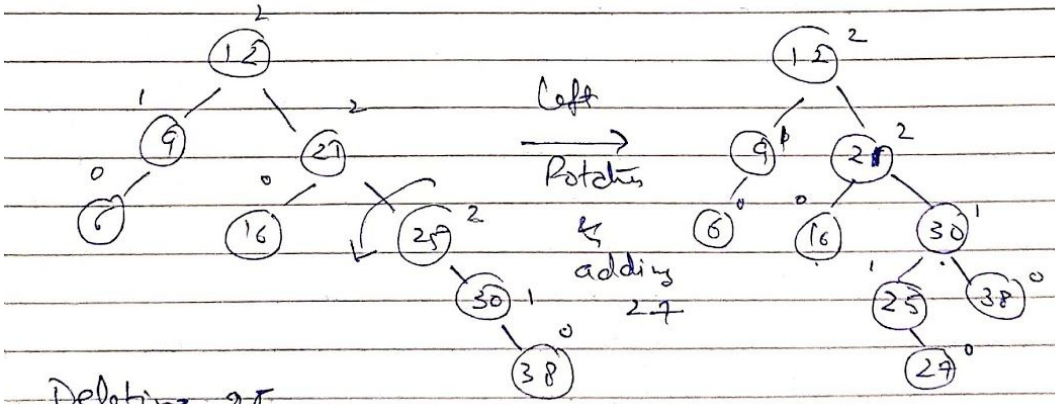
'130 is leaf node so delete it directly without any replacement.

Q3. Build AVL Tree : 16, 21, 25, 9, 12, 6, 30, 38, 29

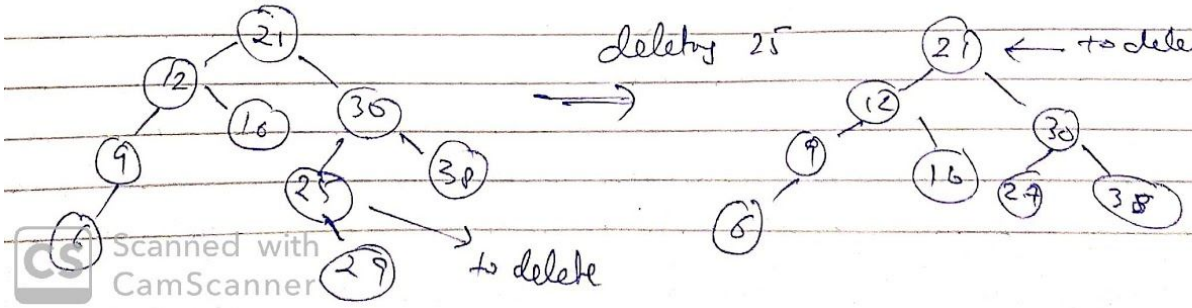




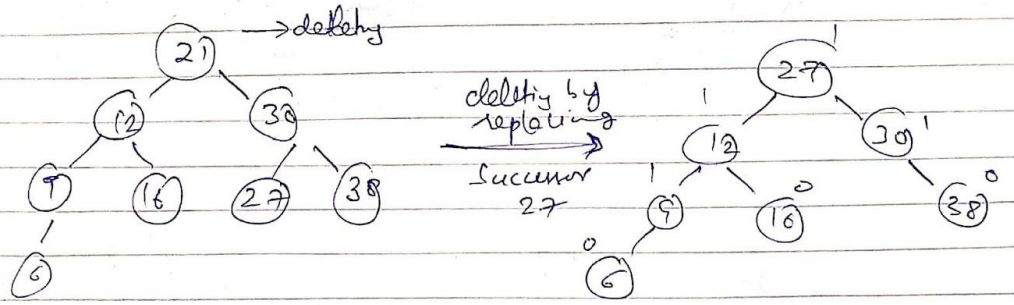




Deleting 25



Deleting 21 by replacing either by 16 or 27 then delete 21.



Assignment-5

Q1. WAP to print postorder traversal of a binary tree from given inorder and preorder traversals.

```
Enter number of nodes in the tree: 8
Enter in order traversal: 6 4 3 8 9 1 2 7
Enter pre order traversal: 1 7 3 8 9 6 4 2
Post Order Traversal: 4 6 9 8 3 2 7 1
```

Q2. WAP to print preorder traversal of a binary tree from given inorder and postorder traversals.

```
Enter number of nodes in the tree: 8
Enter in order traversal: 1 2 3 4 5 6 7 8
Enter post order traversal: 6 5 4 8 7 3 2 1
Pre Order Traversal: 1 2 3 7 4 5 6 8
```

Q3. WAP to create Binary Search Tree and perform insertion and deletion in it.

```
Menu::
1.Insert element in BST.
2.Delete element from BST.
3.Print InOrder traversal.
4.Exit.
```

```
Enter your Choice: 1

Enter value to be inserted: 12

Done.

Menu::
1.Insert element in BST.
2.Delete element from BST.
3.Print InOrder traversal.
4.Exit.

Enter your Choice: 1

Enter value to be inserted: 56

Done.

Menu::
1.Insert element in BST.
2.Delete element from BST.
3.Print InOrder traversal.
4.Exit.

Enter your Choice: 1

Enter value to be inserted: 43

Done.

Menu::
1.Insert element in BST.
2.Delete element from BST.
3.Print InOrder traversal.
4.Exit.

Enter your Choice: 3
```

```
In order traversal: 12 43 56

Menu::
1.Insert element in BST.
2.Delete element from BST.
3.Print InOrder traversal.
4.Exit.

Enter your Choice: 2

Enter value to be deleted: 43

Done.

Menu::
1.Insert element in BST.
2.Delete element from BST.
3.Print InOrder traversal.
4.Exit.

Enter your Choice: 3

In order traversal: 12 56
```

Q4. WAP to implement heapsort (max heap).

```
Enter the size of the array: 8
Input elements of the array: 6 3 7 2 1 9 4 10
Sorted array: 1 2 3 4 6 7 9 10
```

Q5. WAP to implement AVL tree.

```
Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 1

Enter value to be inserted: 6

Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 1

Enter value to be inserted: 4

Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 1

Enter value to be inserted: 89
```

```
Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 1

Enter value to be inserted: 99

Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 1

Enter value to be inserted: 2

Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 2

AVL Tree:
```

```
                        99
                89
Root -> 6
                4
                        2


Menu::
1.Insert Element.
2.Show AVL Tree.
3.Print InOrder traversal.
4.Print PreOrder traversal.
5.Print PostOrder traversal.
6.Exit

Enter your Choice: 3

Inorder Traversal:
2 4 6 89 99
```