ODD SEMESTER 2016 – 2017

MODULE TEST – II

Roll No.....

Programme	: B.E	Date	: 27.09.2016
Branch	: CSE	Time	: 2.30pm to 4.00pm
Semester	: III		
Course Code	: 14CST31	Duration	: 1 ½ Hours
Course Name	: Data Structures	Max. Marks	: 50

PART - A (10 X 2 = 20 Marks)

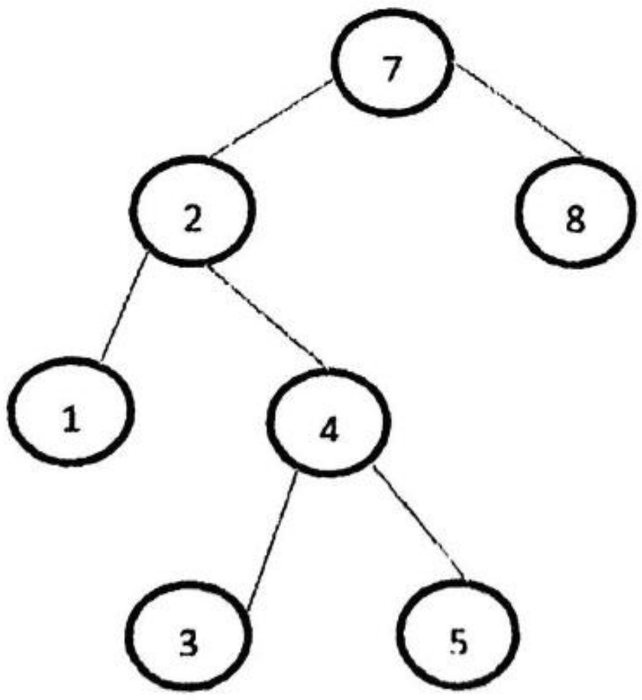
ANSWER ALL THE QUESTIONS

1. Write the node declaration for doubly linked list.
2. Compare array with linked list.
3. Define the following terminologies in a tree i) Depth ii) Sibling
4. Construct an expression tree for the following expression.
   A B C ∧ * D+
5. Define AVL tree.
6. Write the routine for post order tree traversal.
7. With an example, draw the pictorial representation of right –left double rotation in a AVL tree.
8. Write the procedure to perform bubble sort.
9. Mention the ways to pick up a pivot element in quick sort.
10. Give an example scenario for merge sort

PART - B (3 X 10 = 30 Marks)

ANSWER ANY THREE QUESTIONS

11. i) Perform the binary tree traversal for the following tree (6)



ii) Write the routine to delete a node from a singly linked list. (4)

12. Write a C program to perform insert and delete operations in a Binary Search Tree (10)

13. Show the result of inserting 3,1,4,6,9,2,5,7 into an initially empty AVL tree. (10)

14. i) Sort the following data in ascending order using insertion sort (5)
    8 32 34 51 64 21

    ii) Write the routine to perform Insertion sort. (5)

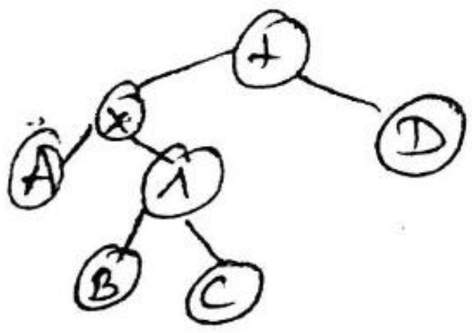
module - II

Part - A

1. Struct node
   { int data;
     Struct node *next, *prev;
   } *head;

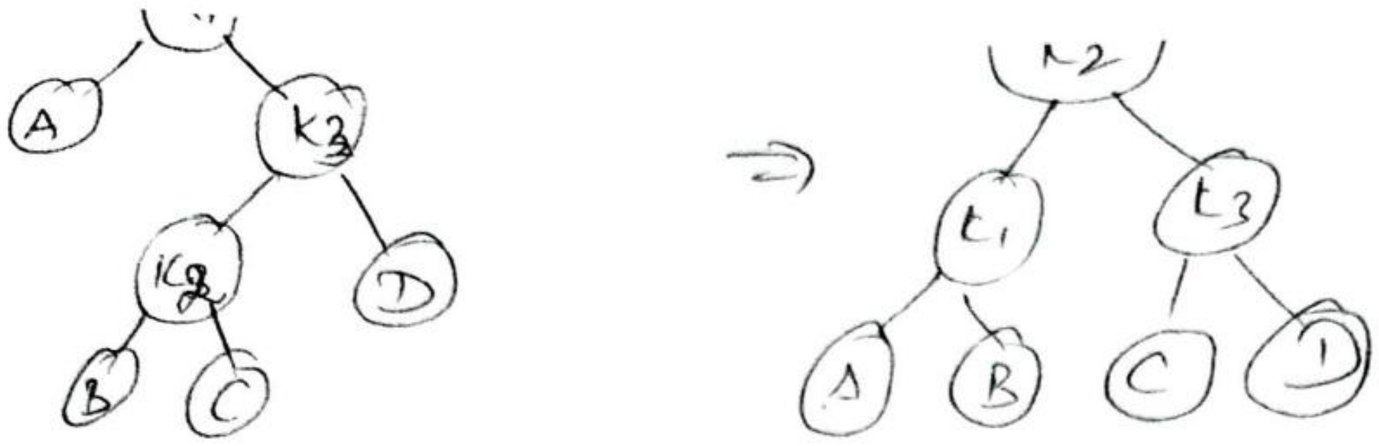
2. Array → Insertion & Deletion tough
   → Contiguous memory allocation
   Linked list → Insertion & Deletion easy
   → Dynamic memory allocation.

3. Depth → Length of the path from root to node n;
   Sibling → Nodes with same parent

4. 

5. It is a binary search tree with balancing condition. i.e the difference b/w height of left & right subtree should be equal to 0, 1, -1.

6. void Postorder (*temp)
```
{
   if (temp != NULL)
   {
      Postorder (temp->left);
      Postorder (temp->right);
       Printf("%d", temp->data);
} }
```



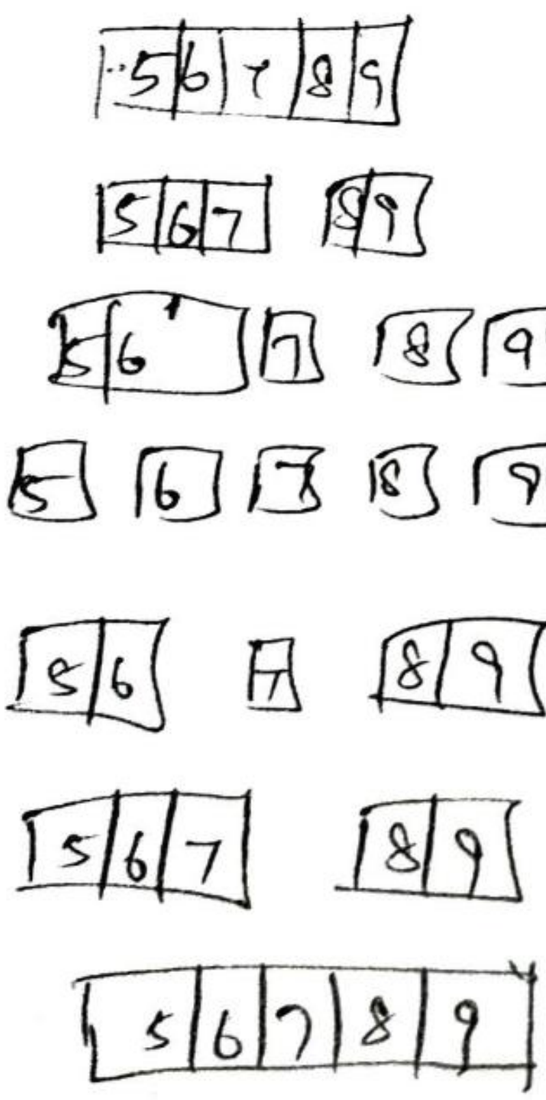
→ Example.

8.

```
for(c=0; c<n; c++)
{
  for(d=0; d<c-n-1; d++)
  {
     swap();
  }
}
```

9.
* First element
* Random element
* median of 3 partitioning.

10.



11. (i) Preorder - 7214358
Inorder - 1234578
Postorder - 1354287 —(6)

(ii) delete a node.

```
void delete()
{
    temp1 = head->next;
if (head->next == NULL)
    Printf("List empty");
else if (temp1->next == NULL)
    head->next = NULL;
else
  {
    head->next= temp1->next
  }
    free(temp1);
```

—(4)

(or Any one delete.

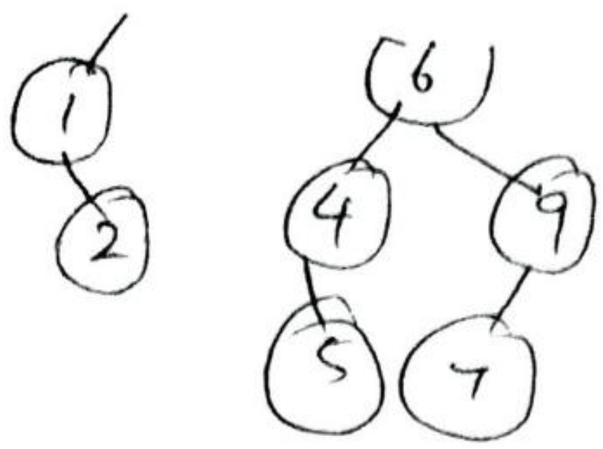
```
Insert()
{
 if (root != NULL)
   {
    temp = temp1;
 }
 else if ( )

temp1   temp1->left = temp1;
  else if()
    temp1->right = temp1;
}
```

—(5)

```
delete()
{
 if(root == NULL)
        Tree empty.
  else if
     Searching on left
else if
     Searching on right
  findmin(temp1);
```

+(5)



-(10)

14. (i) 8 21 32 34 51 64 -(5)

(ii)

```
for (j=1; i<5; j++)
{
temp = a[i];
j = i-1;
while ((temp < a[j] && (j >=0))
{
  a[j+1] = a[j];
  j = j-1;
}
a[j+1] = temp;
```

-(5)

COMPUTER SCIENCE ENGINEERING KEC

A.P RADHA MADHAVAN	A.P [illegible]
Name and Signature of Hall Supdt. with Date	

KONGU ENGINEERING COLLEGE

PERUNDURAI, ERODE - 638 052.

(Autonomous)

ESTD 1984

TÜV SÜD ISO 9001

Name of the Student	S. RADHIKA DEVI	Register No.	1	5	C	S	R	1	5	9
Programme	B.E	Branch & Semester	COMPUTER SCIENCE & ENGINEERING SEMESTER - III							
Course Code and Name	14CST31 and DATA STRUCTURES	Date	27.09.16			No.of Pages Used		10		

MARKS TO BE FILLED IN BY THE EXAMINER

PART - A		PART - B			Grand Total Max. Marks : 50
Question No.	Max Marks : 2	Question No.		Max Marks : 10	
1	2	11	i)	6	45 S.Radhika Devi 90
2	2		ii)	3	
3	2	12	i)	—	
4	2		ii)		
5	2				
6	2	13	i)	9	
7	1		ii)		
8	2	14	i)	4	
9	2		ii)	4	
10	2				
TOTAL	19	TOTAL		26	

Total Marks in words : FOUR FIVE

INSTRUCTION TO THE CANDIDATE

1. Check the Question Paper, Programme ,Course Code, Branch Name etc., before answering the questions.
2. Use both sides of the paper for answering questions.
3. POSSESSION OF ANY INCRIMINATING MATERIAL AND MALPRACTICE OF ANY NATURE IS PUNISHABLE AS PER RULES.

S.V.KOGILAVANI
Name of the Examiner

S.Kogilavani 28/9/16
Signature of the Examiner with Date

PART - A.

1. Node declaration for doubly linked list:

```
struct node
{
  int data;
  struct node *next;
  struct node *prev;
}*head = NULL;
```

2. Array with linked list comparison:

* In array implementation, size of array should be known inadvance.

* If we want to insert an element into the array, we must move the next elements towards right.

* If we want to delete an element from the array, we must move the next elements towards left.

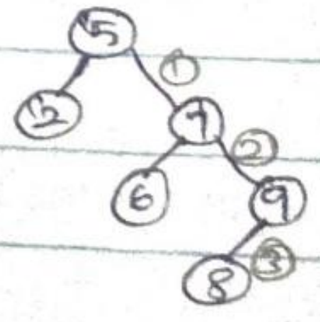
So, It consumes long time

But in linked list, never worry about size, time consumption is less compared to array implementation.

3. Tree terminology:

(1) depth:

* Length of the unique path from root node to node.
* depth of root node is always 0. Eg)
* depth for the given example is 3.



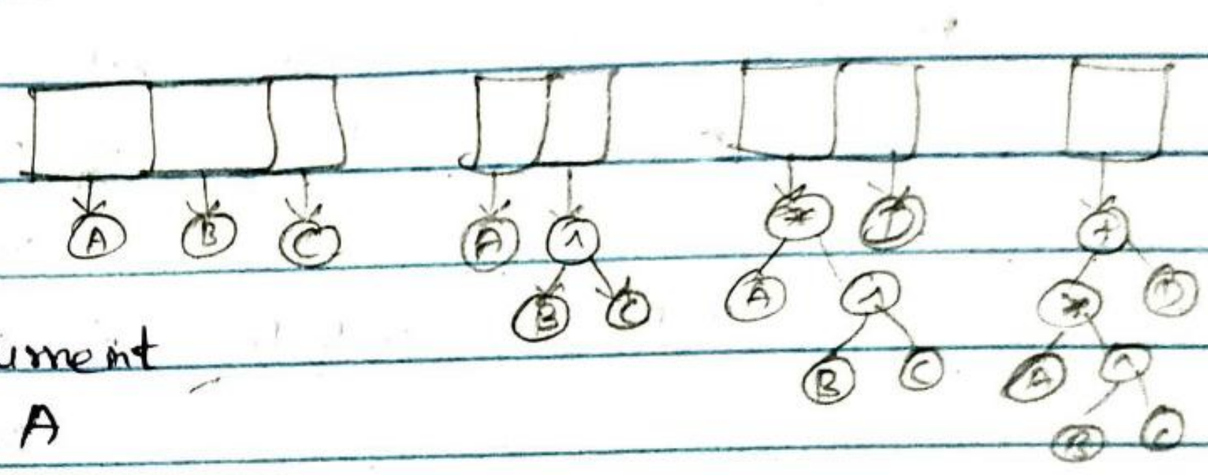
ii) Sibling:

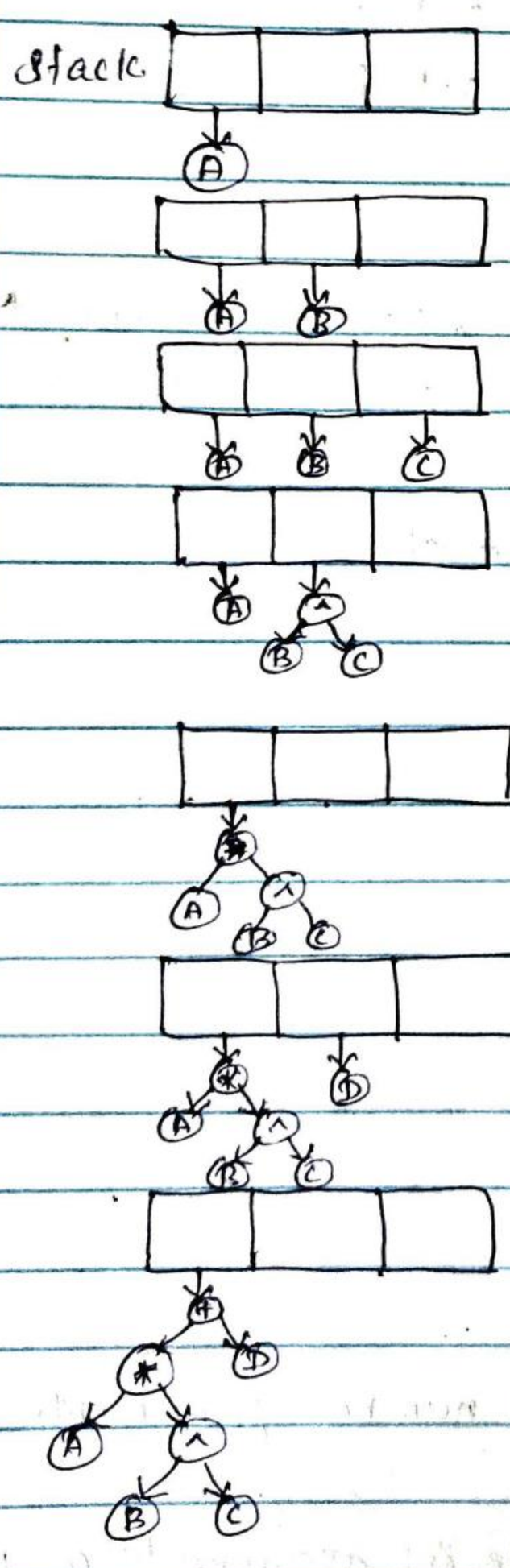
The children which are having the same parent is known as sibling.

For the below example, siblings are (i) 2 and 7
(ii) 6 and 9.

4. Expression tree:

ABC ^ * D +





L : R Root

postorder => ABC^*D+

preorder => +*A^BCD (Root L R)

Inorder => A*B^C+D (L Root R)

5. AVL tree:

(Adelson Velski Landis)

It is similar to Binary search tree with balanced condition. Same as Binary search tree except every node of AVL tree is in balanced condition of (0, -1, 1) If it is not balanced, then do the following rotations.

* Single rotation ⇒ left left rotation
  right right rotation
* Double rotation ⇒ left right rotation
  right left rotation

6. Routine for post order tree traversal:

Traverse through left subtree, ~~visit or process~~ traverse through right subtree and visit or process the root node.

```
void postorder (Binary tree node *t)
{
 if (t != NULL)
   {
    postorder (t.leftchild);
    postorder (t.rightchild);
    visit (root);
   }
}
```

8. Bubble Sort procedure:

Step 1: Get the array elements and number of elements as argument.

Step 2: Using for loop, temporary variable is assigned to 0, this loop

gets increamented till temporary variable value is less than number of elements subtracted from 1.

Step3: Inside this, introduced another for loop, another variable to 0, it gets increamented till variable value is less than number of elements subtracted from 1 and also from previous temporary value.

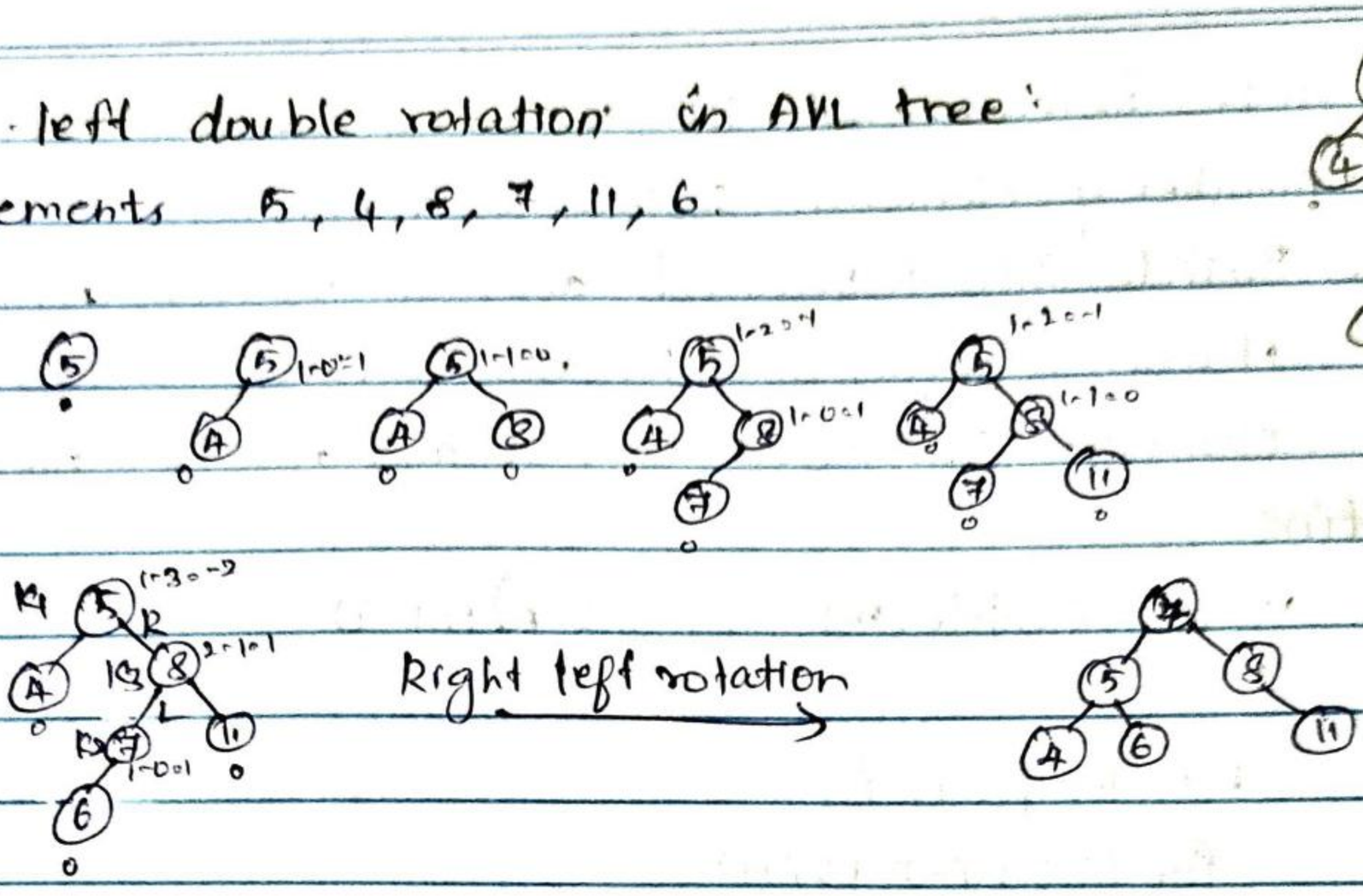
Routine:

```
void bubble(int a[], int n)
{
 int temp, i, j;
 for (i=0; i<n-1; i++)
    {
     for (j=0; j<n-i-1; j++)
     {
      if(a[j]>a[j+1])
       {
        temp = a[j];
        a[j] = a[j+1];
        a[j+1] = temp;
       }
     }
}
```

Step4: If array first value greater than second value, then swap. for till the loop gets terminated.

7. Right left double rotation in AVL tree:

elements 5, 4, 8, 7, 11, 6.



9. The ways to pick up a pivot element in quick sort:

* Pick the first element as pivot element
  It is right choice but not applicable for presorted order or in reverse order.
* pick randomly a pivot element.
  It is safe idea but random generation is difficult process.
* Median of three partitioning.
  It consumes long time. calculation is must.

10 Merge sort:

1 2 4 5. i=0 j=3

i<j mid (i+j)/2 mergesort call

0<3(T) 0+3/2 = 1 1 2 i=0 mid=1

mid (i+j)/2

0+1/2 = 0

mergesort

4 5 mid+1=2 j=3

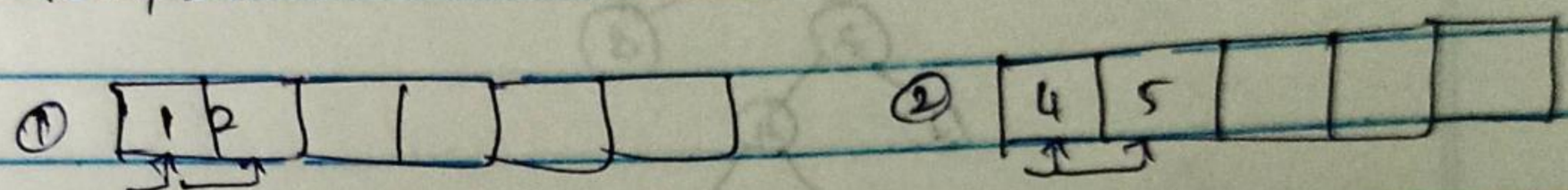
mid 0 merge sort

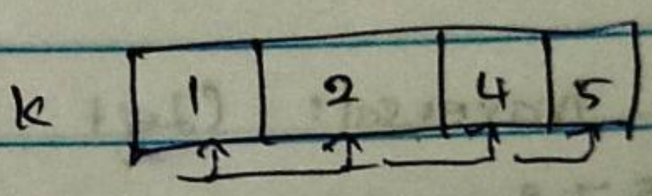
4 5

merge call

0<=1 && 2<=3

temp[k] = 1



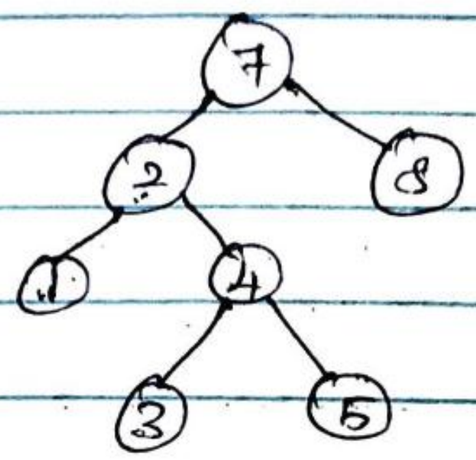



```
void merge(int a[], int i, int j)
{
int mid;
if (i<j)
  {
   mid=(i+j)/2;
   merge(a,i,mid);
   merge(a,mid+1,j);
  mergesort(a,i,mid,mid+1,j)
}
void mergesort(int a[], int i, int k, int j, int j)
 {
while (i<=k && j<=j)
   temp[k++]=a[i++]
else
   temp[k++]=a[j++]
while (i<=k)
   temp[k++]=a[i++]
while (j<=j)
   temp[k++]=a[j++]
```

```
for(i=0,j=0; i<=j1,i++,j++)
a[i]=temp[i]
}
```

PART - B

11. (i) Binary tree traversal:



Inorder: tree traversal: (left root right)

1 2 3 4 5 7 8

preorder tree traversal: (root left right)

7 2 1 4 3 5 8

postorder tree traversal: (left right root)

1 3 5 4 2 8 7

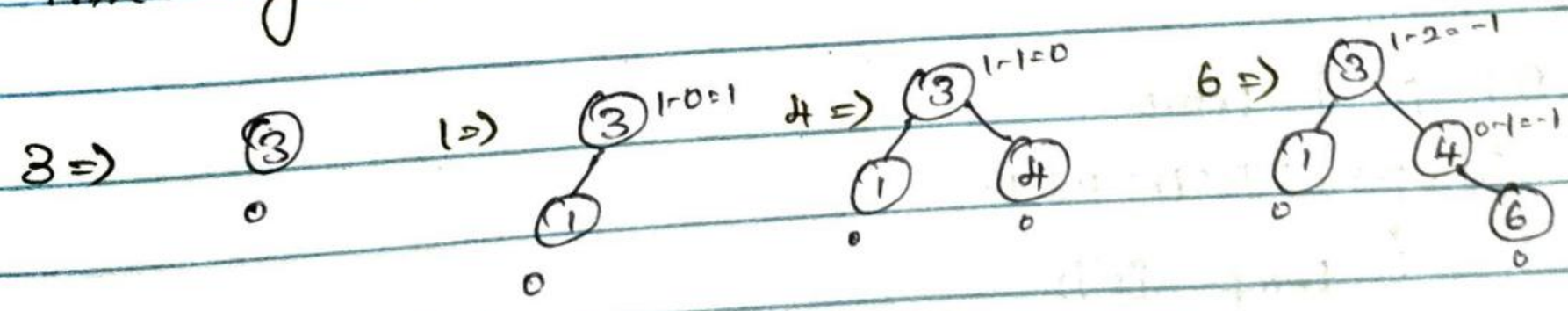
(ii) Routine to delete a node from a singly linked list:

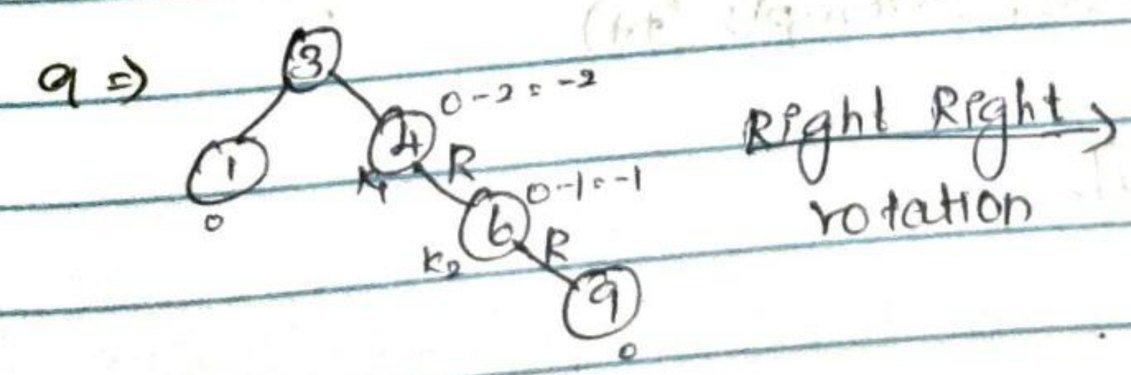
```
int delete(int num)
{
 struct node *cur, *prev; cur = head;
 while (cur != head)
 {
  if (cur -> data = num)
    {
  if (cur = head)
  {
    head = cur -> next;
    free (cur);
    return 0;
```

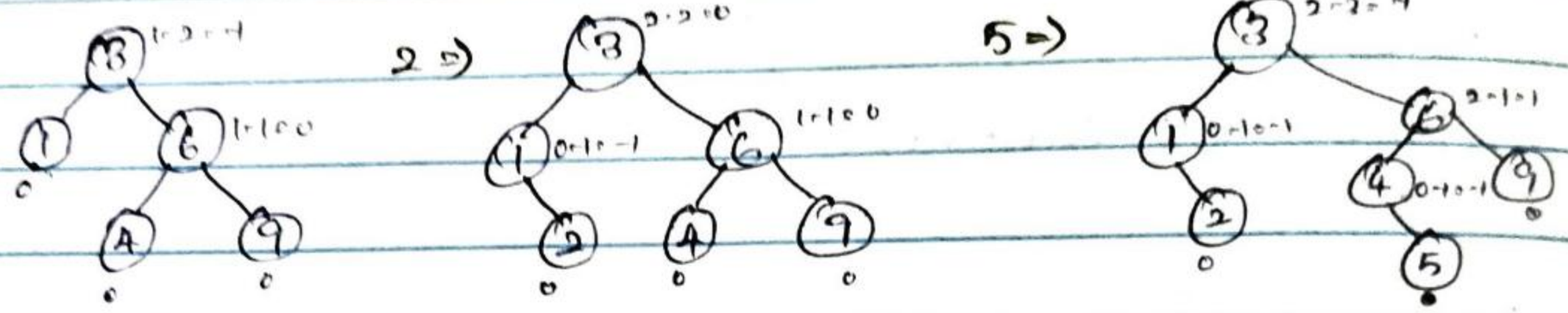
```
else
prev -> next = cur -> next;
 free (cur);
 return 0;
 }
else
 {
  prev = cur;
  cur = cur -> next;
  }
}
printf("%d is not found", num);
return 1;
}
}
```

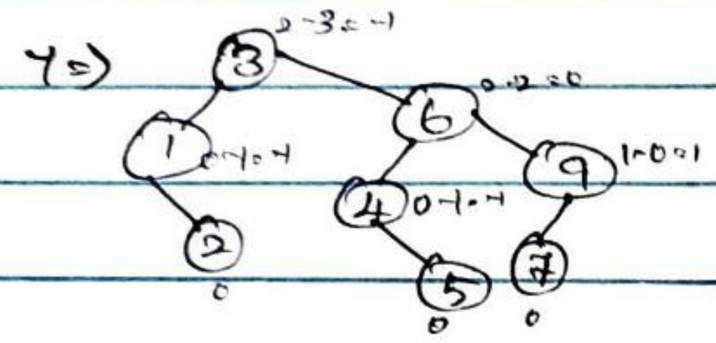
12. (Incorrect)

13. Inserting 3, 1, 4, 6, 9, 2, 5, 7

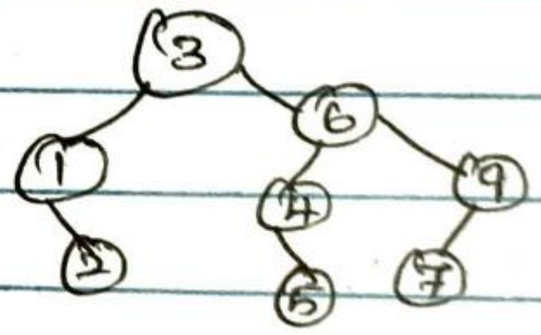








∴ Resultant AVL tree is



14. (ii) Routine to perform insertion sort:

```
void insert(int a[], int n)
{
 int p, j, temp;
 for(p=1; p<n; p++)
     { temp=a[p];
     for(j=p; (j>0)&&(a[j-1]>temp); j++)
         {
          a[j] = a[j-1];
         }
     a[j] = temp;
     }
}
```

14. (i) using insertion sort

8 32 84 51 64 21 n=6

p	p<n	temp	j=p	(j>0)&&(a[j-1]>temp)	a[j]
1	1<6(T)	32	1	1>0(T) 8>32(F)	~~a[1]=32~~
			~~0~~	~~0>0(F)~~	a[1]=32
2	2<6(T)	34	2	2>0(T) 32>34(F)	
			~~1~~	~~1>0(T) 8>34(F)~~	a[2]=34
			~~0~~	~~0>0(F)~~	
3	3<6(T)	51	3	3>0(T) 34>51(F)	
			~~2~~	~~2>0(T) 32>51(F)~~	
			~~1~~	~~1>0(T) 8>51(F)~~	a[3]=51
			~~0~~	~~0>0(F)~~	
4	4<6(T)	64	4	4>0(T) 51>64(F)	
			~~3~~	~~3>0(T) 34>64(F)~~	
			~~2~~	~~2>0(T) 32>64(F)~~	
			~~1~~	~~1>0(T) 8>64(F)~~	
			~~0~~	~~0>0(F)~~	a[4]=64.
5	5<6(T)	21	5	5>0(T) 64>21(T)	a[5]=64
			4	4>0(T) 51>21(T)	a[4]=51
			3	3>0(T) 34>21(T)	a[3]=34
			2	2>0(T) 32>21(T)	a[2]=32
			1	1>0(T) 8>21(F)	a[1]=21
			0	0>0(F)	

The sorted ordered ascendering order list is

8 21 32 34 51 64