Using version 2 :-

2. Input String:-

E A S Y Q U E S T I O N.

Quicksort:-

E A S Y Q U E S T I O N

E A I E N U Y S T S O Q

E A E I

A E

E

O Q S T S U Y

O

S T S U Y

S T S U

S T S

S S T

S T

∴ The sorting result is :-

A E E I N O Q S S T U Y

1. Version-1:-

Input string:-

E A S Y Q U E S T I O N

U S S T Y O Q

Step-1: E A E I (N) U S S T Y O Q

2: E A E (I)

3: E A (E)

4: (A) E

5: (E)

6:

U S S T Y O Q

Q O (Q) S T Y U S

(O) S T Y U S

S (S) Y U T

(S) Y U T

(T) U Y

U Q

(Y)

∴ The final o/p is:-

A E E I N O Q S S T U Y

7.6. Example of 6 files in the case where Quicksort gives worst-case performance.

EX-1:-

9 8 7 6 5 4 3 2 1 10

9 (Ans).

EX2: 8 7 6 5 4 3 2 1 9 10

EX3: 7 6 5 4 3 2 1 8 9 10

EX4: 6 5 4 3 2 1 7 8 9 10

EX5: 5 4 3 2 1 6 7 8 9 10

EX6:- 4 3 2 1 5 6 7 8 9 10

(Quicksort shows worst case behavior when the partitioning element resides on the extreme sides of the list. (Ans)

6.74.

0	1	2	3	4	5	6	7	8	9	10
A	B	R	A	C	A	D	A	B	R	A

	A	B	C	D	R
A → 5	0	5	2	1	1
B → 2					
C → 1				8	9
D → 1	0	5	7		
R → 2					

	0	1	2	3	4	5	6	7	8	9	10 (Location)
A	A					B					
B	A					B				R	
R	A					B				R	
A	A	A				B				R	
C	A	A						C		R	
A	A	A	A			B		C		R	
D	A	A	A						D	R	
A	A	A	A	A		B		C	D	R	
B	A	A	A	A		B	B	C	D	R	
R	A	A	A	A		B	B	C	D	R	R
A	A	A	A	A	A	B	B	C	D	R	R

(SORTED)

Q.41. LSD Radix Sort

Keywords	1st leading place	2nd leading place
now	all	party
is	aid	people
the	come	of
time	for	the
for	good	the
all	is	their
good	now	aid
people	of	time.
to	people	all
come	party	now
the	the	good
aid	time	to
of	to	come
their	the	is.
party	their.	

(Ans).

3.25. Returns no of nodes in a circular linked list.

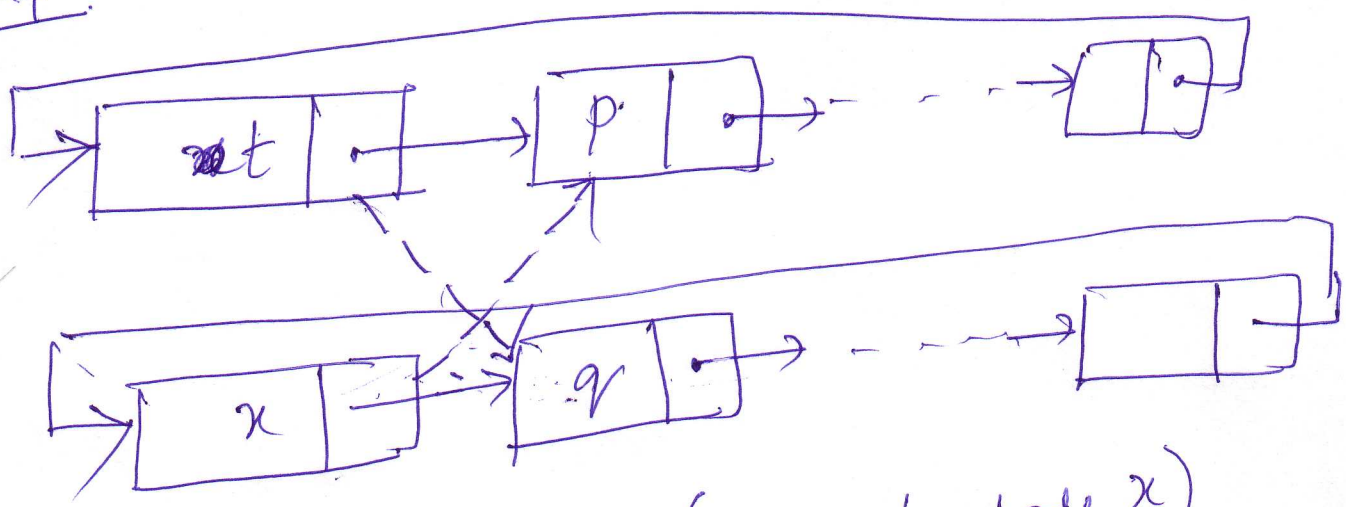
Method:

```
int count_node (Node p). */ p is a
                            Node type variable
{ int count = 0;            having two fields,
    Node start = p;         val & next */
    while (p.next != start)
      { count ++;
        p = p.next;
      }
    return count;
}.
```

3.26. Code fragment that determines the no of nodes that are between two given references x & t.

```
int Count-node(Node x, node t)
{ int count = 0;
  while ( x.next != t )
    { count++;
      x = x.next;
    }
  return count;
}
```

3.27

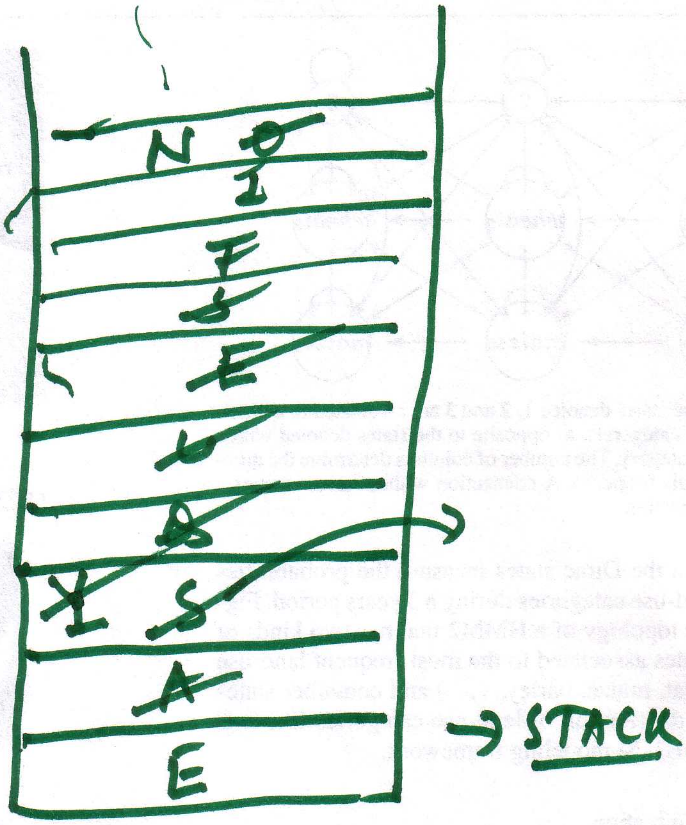



```
void list-ins (Node t, Node x)
{  Node q = x.next;
   Node p = t.next;
   x.next = p;
   t.next = q;
}
```

This entire insertion takes $\Theta(1)$ time.

4·5.

E
A
S
pop-1 *
Y
(2) *
Q
U
E
(3) *
(4) *
(5) *
S
T
(6) *
(7) *
(8) *
I
O
(9) *
N
(10) *
(11) *
(12) *



pop (1) → Returns → S
(2) Y
(3) E
(4) U
(5) Q
(6) T
(7) S
(8) A
(9) O
(10) N
(11) I
(12) E

Array Representation

4.18. class intStack

```
{ private int[] s;
  private int N;
  intStack(int maxN)
  { s = new int[maxN]; N = 0; }
  int count()
  { return N; }
  void push(int item)
  { s[N++] = item; }
  int pop()
  { return s[--N]; }
}
```

(Ans).

Linked list :-

```
class LinkedStack
{ private node head;
  private class node
  { - - -
  }
  - - - - - -
  int count()
  { private count;
    while (head.next != null
    {
```

Linked list Representation:-

```
class intStack
{ private Node head;
  private class Node
    {
      - - - -
      - - -
    }
  intStack(int maxN)
  . - - - - -

  int count ( )                    */ isEmpty() is
{ private int count = 0;              replaced
  while (head.next != null)           by count
    { head = head.next;               method
      count++;                        */
    }
  return count;
}
  void Push (int item)
  { - - - -
  }
  int pop( )                                (Ans)
  { -
  }
```

4.9. Convert to Postfix Expression:

(5 * ((9 * 8) + (7 * (4 + 6))))

Input	Output	Stack
(		
5	5	
*		*
(		*
(		*
9	9	*
*		* *
8	8	*
)	*	*
+		* +
(		* +
7	7	
*		* + *
(		* + *
4	4	
+		* + * +
6	6	* + * +
)	+	
)	*	
)	+	
)	*	

∴ The postfix expression is

5 9 8 * 7 4 6 + * + *

4.10.

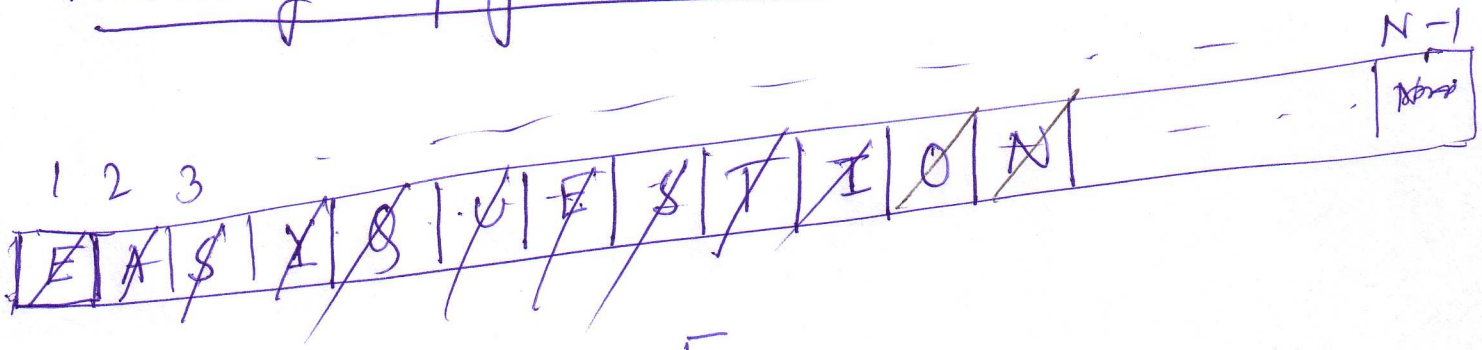
Postfix Evaluation. (showing stack content).

taken	stack
5	5
9	5 9
*	45
8	45 8
7	45 8 7
4	45 8 7 4
6	45 8 7 4 6
+	45 8 7 10
*	45 8 70
2	45 8 70 2
1	45 8 70 2 1
3	45 8 70 2 1 3
*	45 8 70 2 3
+	45 8 70 5
*	45 8 350
+	45 358
*	16110 (Answer)

4.36. E A S * Y * Q U E * * *

S T * * * I O * N * * *.

Initially empty queue:-



1st get (*) → Returns E

2nd " (*) → " A

3rd " (*) → " S

4th " (*) → " Y

5th " (*) → " Q

6th " (*) → " U

7th " (*) → " E

8th " (*) → " S

9th " (*) → " T

10th " (*) → " I

12th " (*) → " O

12th (*) → " N. (Ans).

14.31. EASY QUTION

M = 16.

Hashfunction 1 = 11 K mod M

" 2 = (K mod 3) + 1.

Hash Table

0	
1	S
2	
3	Y
4	I
5	O
6	
7	E
8	U
9	
10	N
11	A
12	T
13	U
14	Q.
15	

input symbol scanned

E = Hf1(E)
= 11K mod 16
= 11X5 mod 16
= 55 mod 16. = 7

A = Hf1(A) = 11 mod 16 = 11

S = Hf1(S) = 11X19 mod 16 = 1

Y = Hf1(Y) = 25X11 mod 16

Q = Hf1(Q) = 17X11 mod 16
= 11 (collision).

∴ Q = Hf2(Q) = (17 mod 3) + 1
= 3. (probe increment)

∴ Q is inserted in = 14.

U = Hf1(U) = 7 (collision)

U = Hf2(U) = (21 mod 3) + 1 = 1

∴ U goes into 7+1 = 8

T = Hf1(T) = 12

I = Hf1(I) = 3 collision
Hf2(I) = 1
∴ I goes to 3+1 = 4th location

O = Hf1(O) = 5

N = Hf1(ON) = 10

14.32

$Hf_1 = 11K \bmod M$ (initial probe)

$Hf_2 = (K \bmod 3) + 1$ (increment).

$M = 10$.

Hash Table - M

0	Q
1	A
2	U
3	T
4	I
5	E
6	O
7	Y
8	N
9	S

EASY QUTION

$Hf_1(E) = 11 \times 5 \bmod 10$
$= 5$

$Hf_1(A) = 11 \bmod 10 = 1$

$Hf_1(S) = 19 \times 11 \bmod 10 = 9$

$Hf_1(Y) = 25 \times 11 \bmod 10 = 5$ (collision)

$Hf_2(Y) = (25 \bmod 3) + 1 = 2$.

∴ Y goes to $5 + 2 = 7$ th location

$Hf_1(Q) = 7$ (collision)

$Hf_2(Q) = 3$

∴ Q goes to $= 0$ th location)

$Hf_1(U) = 1$ (collision)

$Hf_2(U) = 1$

∴ U goes to $1 + 1 = 2$

$Hf_1(T) = 6$ (collision)

$Hf_2(T) = (20 \bmod 3) + 1 = 3$

~~$= 0$ (collision).~~

$Hf_1(I) = 99 \bmod 10 = 9$ (collision)

$Hf_2(I) = 1$.

goes finally to location 4.

$Hf_1(O) = 15 \times 11 \bmod 10$
$= 5$ (collision)

$Hf_2(O) = 1$

∴ O goes to 6th location

$Hf_1(N) = 4$ (collision)

$Hf_2(N) = 3$

finally gets inserted to location 8 after trying 7, 0, 3, 6, 9, 2, 5 positions

5.59

```
Node remove (Node h, Item v)
{
    if (h==null) return null;
    if (equals(h.item, v))
    {
        h.l = null;
        h.r = null;
    }
    if (h.l != null) remove(h.l, v);
    if (h.r != null) remove(h.r, v);
}
```

5.68 Total no. of nodes = NM + 1

∴ No. of external nodes = MN + 1 − N.

5.79

preorder :- DBACFEG, CBADE, ECBADHFGI

inorder :- ABCDEFG, ABCDE, ABCDEFGHI

postorder :- ACBEGFD, ABEDC, ABDCGFIHE

level order :- DBFACEG, CBDAE, ECHBDFIAG

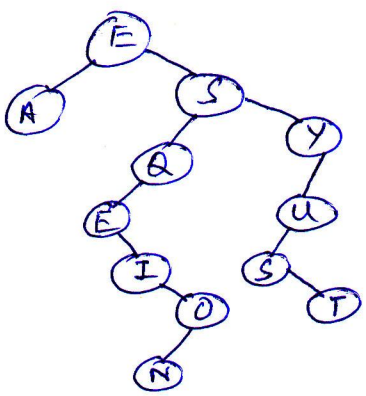
5.86

```
static int countLeaves (TreeNode node)
{
    if (node == null) return 0;
    else if ((node.left == null) && (node.right == null))
            return 1;
        else
            return countLeaves (node.left) + countLeaves
                                              (node.right);
}
```

5.87

```
static int countChild (Treenode node)
{
    if (node == null) return 0;
    else if (((node.left == null) && (node.right != null))
    ||((node.left != null) && (node.right == null)))
    return 1;
    else
        return countChild (node.left) + countChild(node.right);
}.
```

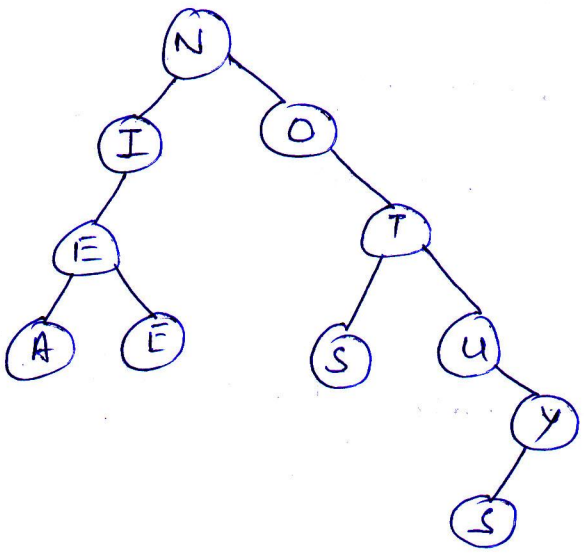
12.58



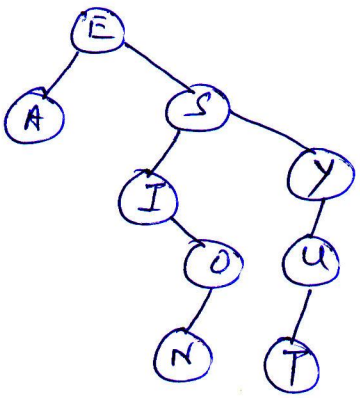
12.70

```
static int height(Treenode node)
{
    if (node == null) return 0;
    else if ((node.left == null) && (node.right == null))
        return 0;
    else
        return max(height(node.left), height(node.right))
                + 1;
}
```

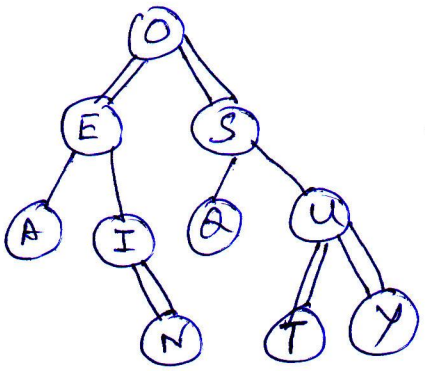
12.84



12.90



13.48



double edges are the 'red' edges.

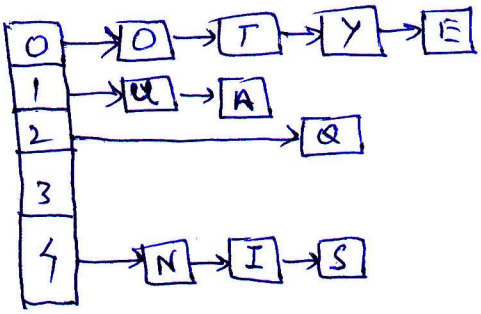
13.53

Because every 3-node can give 2 possible orientations, we can have a total of 2^t combinations.

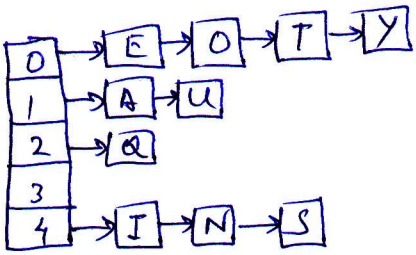
14.17

E A S Y Q U T I O N

0 1 4 0 2 1 0 4 0 4



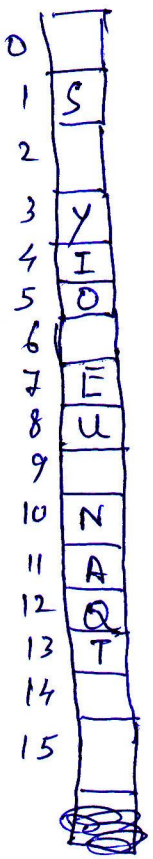
14.18



Answer does not depend on the order of inserting items.

14.25

E	A	S	Y	Q	U	T	I	O	N
7	11	1	3	11	7	12	3	5	10



14.26

E	A	S	Y	Q	U	T	I	O	N
5	1	9	5	7	1	0	9	5	4