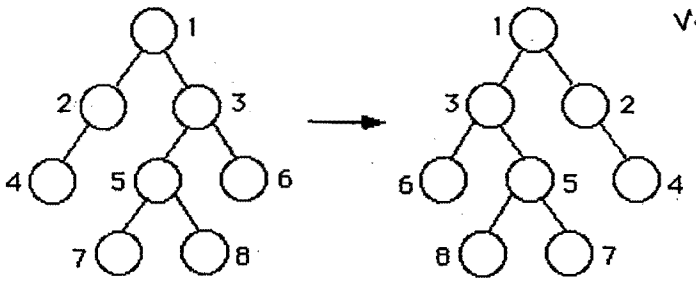
(b) Write a recursive function that will interchange all left and right subtrees of a binary tree. The function should take one parameter, a pointer to the root (or subroot) of the tree (or subtree) to be interchanged, for example see the diagram shown below.




```
void reverse(Node
{
if(T != NULL)
{
reverse(T->leaft
reverse(T-> right
Node* temp = T->lea
T->leaft = T->right;
T->right = temp;
}
}
```

Question 4. (12+8 marks)

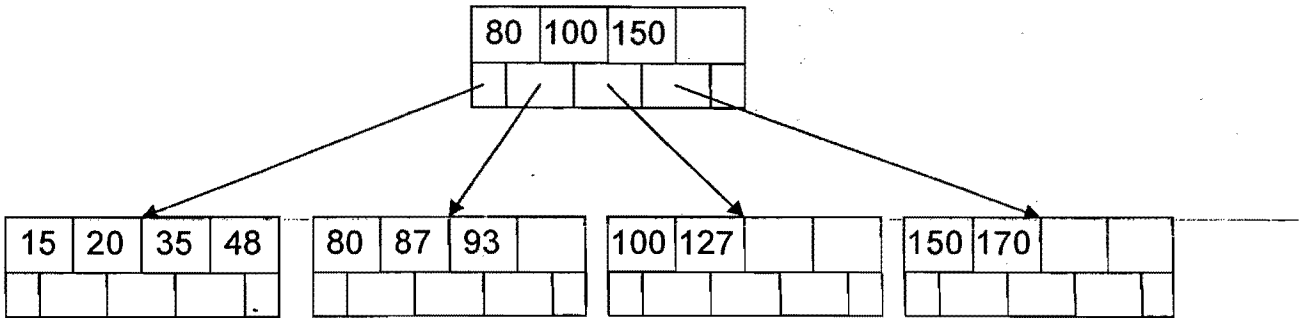
A B+-Tree of order 5 has the following characteristics

(a) root node can have minimum one and maximum 4 keys .

(b) all other nodes including leaf nodes can have minimum 2 and maximum 4 keys

(c) the data is stored only in leaf nodes i.e. leaf nodes are data nodes and intermediate nodes are index nodes.

Consider the following B+-tree



(i) Insert 128, 135, 165, and 167 into the B+-Tree and show the result on a tree diagram. Then insert 25 and 140 one at a time in the given order and show the result of each insertion on a separate tree diagram. Also, state if a merge operation is needed during insertion of a key.

(ii) Delete 135 and 150 one at a time in the given order from the B+-tree which you get in part (i) and show the result of each deletion on a separate tree diagram.

Question 5. (15 marks)

(a) Insert the following keys into a hash table of TableSize equal to 7: 811, 711, 633, 355, 635, 423, and 458. The hashing function is *H(key) = key mod TableSize*. The collisions are resolved through linear rehashing with of step equal to 1 i.e. (*rehash = (H(key) + 1) mod TableSize*). After the insertions determine how many probes are required to find the keys 423 and 511.

(b) Repeat the above insertions with external chaining used as the collision resolution strategy. And after the insertions determine the number of search steps required to find the keys 423 and 511.

Table size = 7

H(key) = key mod table Size

Rehash = (H(Key) + 1) mod table Size

Insert 811, 711, 633, 355, 635, 423, 458

811 : H(811) = 811 mod 7 = 6 , we store 811 in index 6 with 1 prob.

0	1	2	3	4	5	6
emp	emp	emp	emp	emp	emp	811

711 : H(711) = 711 mod 7 = 4 , we store 711 in index 4 with 1 prob.

0	1	2	3	4	5	6
emp	emp	emp	emp	711	emp	811

633 : H(633) = 633 mod 7 = 3 , we store 633 in index 3 with 1 prob.

0	1	2	3	4	5	6
emp	emp	emp	633	711	emp	811

355 : H(355) = 355 mod 7 = 5 , we store 355 in index 5 with 1 prob.

0	1	2	3	4	5	6
emp	emp	emp	633	711	355	811

635 : H(635) = 635 mod 7 = 5, Now we do rehash :

5 + 1 mod 7 = 6 mod 7 = 6, rehash again

6 + 1 mod 7 = 7 mod 7 = 0, we store 635 in index 0 with 3 prob.

0	1	2	3	4	5	6
635	emp	emp	633	711	355	811

423 : H(423) = 423 mod 7 = 3 , Now we do rehash :

3 + 1 mod 7 = 4 mod 7 = 4, rehash again

4 + 1 mod 7 = 5 mod 7 = 5, rehash again

5 + 1 mod 7 = 6 mod 7 = 6, rehash again

6 + 1 mod 7 = 7 mod 7 = 0, rehash again

7 + 1 mod 7 = 8 mod 7 = 1, we store 423 in index 1 with 6 prob.

0	1	2	3	4	5	6
635	423	emp	633	711	355	811

458 : H(458) = 458 mod 7 = 3 , Now we do rehash :

3 + 1 mod 7 = 4 mod 7 = 4, rehash again

4 + 1 mod 7 = 5 mod 7 = 5, rehash again

5 + 1 mod 7 = 6 mod 7 = 6, rehash again

6 + 1 mod 7 = 7 mod 7 = 0, rehash again

7 + 1 mod 7 = 8 mod 7 = 1, rehash again

8 + 1 mod 7 = 9 mod 7 = 2, we store 458 in index 0 with 7 prob.

0	1	2	3	4	5	6
635	423	458	633	711	355	811
Prob. 3	6	7	1	1	1	1

Find Key : 423, we apply H(key) = key mod 7

423 : H(423) = 423 mod 7 = 3 , but it is not in index 3, Now we do rehash :

3 + 1 mod 7 = 4 mod 7 = 4, but it is not in index 4, rehash again

4 + 1 mod 7 = 5 mod 7 = 5, but it is not in index 5, rehash again

5 + 1 mod 7 = 6 mod 7 = 6, but it is not in index 6, rehash again

6 + 1 mod 7 = 7 mod 7 = 0, but it is not in index 0, rehash again

7 + 1 mod 7 = 8 mod 7 = 1, we found 423 in index 1 with 6 prob.

Find Key : 511, we apply H(key) = key mod 7

511 : H(511) = 511 mod 7 = 0 , but it is not in index 0, Now we do rehash :

0 + 1 mod 7 = 1 mod 7 = 1, but it is not in index 1, rehash again

1 + 1 mod 7 = 2 mod 7 = 2, but it is not in index 2, rehash again

2 + 1 mod 7 = 3 mod 7 = 3, but it is not in index 3, rehash again

3 + 1 mod 7 = 4 mod 7 = 4, but it is not in index 4, rehash again

4 + 1 mod 7 = 5 mod 7 = 5, but it is not in index 5, rehash again

5 + 1 mod 7 = 6 mod 7 = 6, but it is not in index 6, rehash again

6 + 1 mod 7 = 7 mod 7 = 0, but it is not in index 0, we have done 7 probs, all elements occupied, the find will fail.