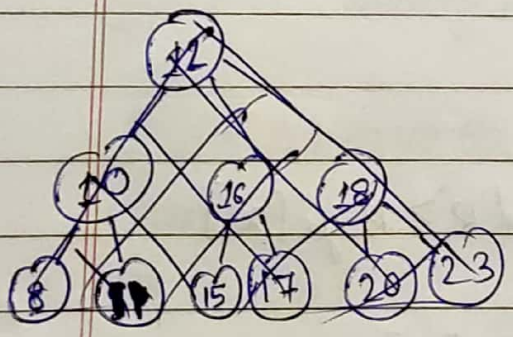
Experiment 10

1) B T search

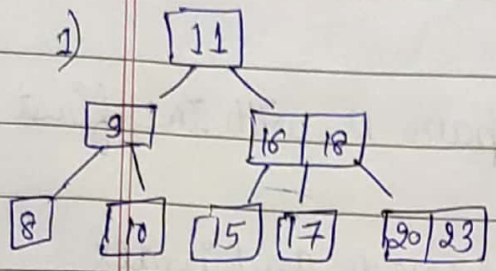
Steps (i) Starting from the root node, compare k with the first key of the node.
If k = first key of node, return the node and index
(ii) If k.leaf = true, return NULL (ie., not found)
(iii) If k < first key of root node, search the left child of this key recursively.
(iv) If there is more than one key in the current node and k > the first key, compare k with the next key in the node.
If k < next key, search the left child of this key (ie., k lies in b/w first and the second keys)
Else, search the right child of the key.
(v) Repeat steps (i) to (iv) until the leaf is reached.

Example:- Search key k = 17 in the tree of degree 3

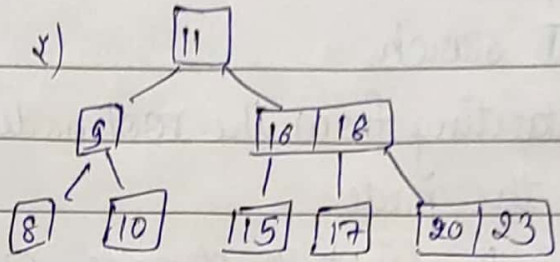


→ k is not found at root, compare it with root key.
→ since k > 12, go to right child of root
→ Compare k with 16. Since k > 16, compare k with next key 18.
→ Since k < 18, k lies b/w 16 and 18. Search in the right child of 16 or the left child of 18
→ k is found

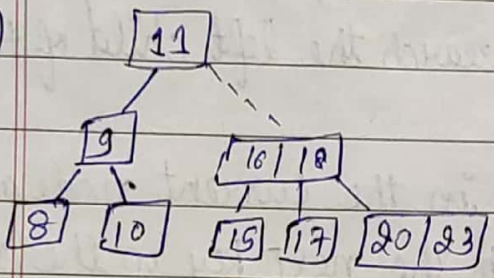
1)



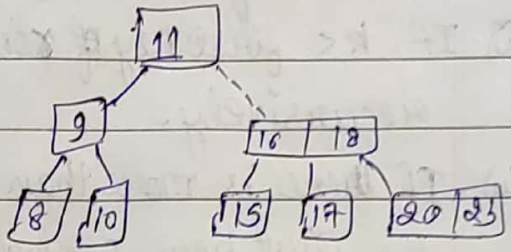
2)



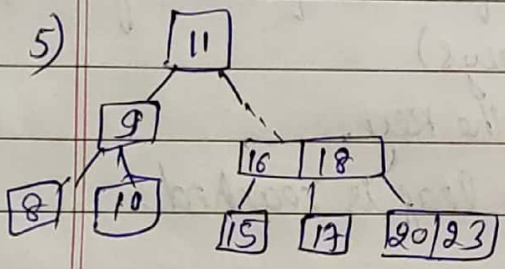
3)



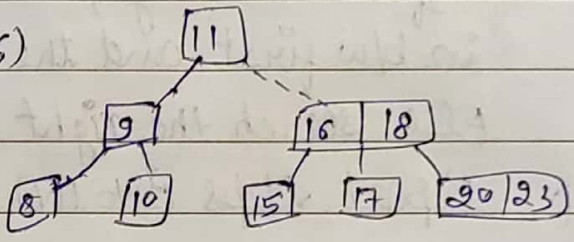
4)



5)



6)



Pseudocode

```
BTreeSearch (x, k)
  i = 1
while i ≤ n[x] and k ≥ keyi[x]
      do i = i+1
if i=n[x] and k = keyi[x]
    then return (x, i)
if leaf [x]
    then return NIL
else
    return BTreeSearch(ci[x], k)
```

① Insertion

Steps | 1) Initialize x as root

2) While x is not leaf, do following

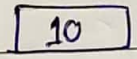
3) (a) Find the child of x that is going to be traversed next. Let the child be y.

(b) If y is not full, change x to point to y

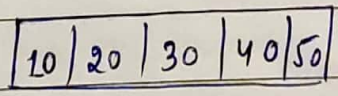
(c) If y is full, split it and change x to point to one of the two parts of y. If k is smaller than mid key in y, then set x as the first part of y. Else second part of y. When we split y, we move a key from y to its parent x.

3) The loop in (2) stops when x is leaf. x must have space for 1 extra key as we have been splitting all nodes in advance. So simply insert k to x.

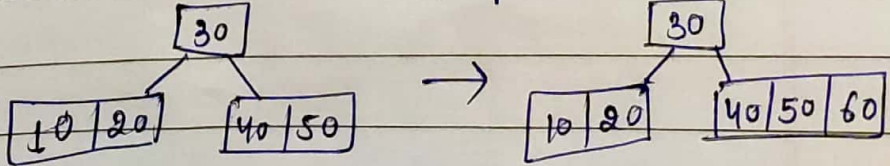
Example:- Initially root is NULL. First insert 10



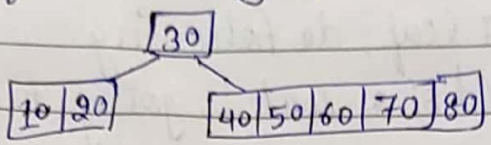
Insert 20, 30, 40, 50. They all will be inserted in root bcz. the maximum no. of keys a node can accommodate is 2*t-1 which is 5. (as t=3)



Insert 60. Since root node is full, it will first split into two then 60 will be inserted into the appropriate child.



Insert 70 and 80. These new keys will be inserted into the appropriate leaf without any split.



Insert 90. This insertion will cause a split. The middle key will go up to the parent