PRANAV JAGADEESH 1BM18CS071

PROGRAM 7 5B WRITE-UP BATCH-4

```
class BTreeNode
{ int *keys
  int t
  BTreeNode **C
  int n
  bool leaf
  public: BTreeNode(int _t, bool _leaf):
          void insertNonFull(int K)
          void splitChild(int i, BTreeNode *y)
          void traverse()
          BTreeNode *search(int K)
          friend class BTree
        }
class BTree
{ BTreeNode *root
  int t
  public: BTree(int _t)
    { root <- NULL
      t <- _t
    }
    void traverse()
  { if (root != NULL)
      root -> traverse()
  }
    BTreeNode * search(int K)
  { return (root <-> NULL) ? NULL : root -> search(K)
  }
    void insert(int K)
}
BTreeNode :: BTreeNode(int t1, bool leaf1)
{ t <- t1
  leaf <- leaf1
```

```
    Keys ← new int [2*t - 1]
    C ← new BTreeNode *[2*t]
    n ← 0
}
void BTreeNode :: traverse()
{ int i
  for (i ← 0; i < n; i++)
    { if (leaf ⇔ false)
        C[i] → traverse()
      print Keys[i]
    }
  if (leaf ⇔ false)
    C[i] → traverse()
}
void BTree :: insert(int K)
{ if (root ⇔ NULL)
    { root ← new BTreeNode(t, true)
      root → Keys[0] ← K
      root → n ← 1
    }
  else
  { if (root → n ⇔ 2*t - 1)
    { BTreeNode *S ← new BTreeNode(t, false)
      S → C[0] ← root
      S → splitChild(0, root)
      int i ← 0
      if (S → Keys[0] < K)
        i++
      S → C[i] → insertNonFull(K)
      root ← S
    }
    else root → insertNonFull(K)
```

```
    }
  }
  void BTreeNode :: insertNonFull(int k)
{ int i <- n-1
  if( leaf <-> true)
  { while( i >=0 && keys[i] >k)
    { keys[i+1] <- keys[i]
      i--
    }
    keys[i+1] <- k
      n <- n+1
  }
  else
  { while( i >=0 && k[i] >k)
      i--
    if ( C[i+1] -> n <-> 2*t-1)
    { splitChild(i+1, c[i+1])
      if ( keys[i+1] < k)
          i++
    }
    c[i+1] -> insertNonFull(k)
  }
}
void BTreeNode :: splitChild(int i, BTreeNode *y)
{ BTreeNode * z <- new BTreeNode(y->t, y->leaf)
  z-> n <- t-1
  for (int j <-0, j<t-1, j++)
    z-> keys[j] <- y->keys[j+t]
  if(y -> leaf <-> false)
    for (int j<-0; j<t, j++)
      z -> C[j] <- y->C[j+t]
}
```

```
    y -> n <- t-1
    for (int j <- n, j >= i+1, j--)
      c[j+1] <- c[j]
    c[i+1] <- z
    for (int j <- n-1, j >= i, j--)
      keys[j+1] <- keys[j]
      keys[i] <- y -> keys[t-1]
      n <- n+1
}

  int main()
{
  int n <- 0, int xx <- 0
    int k <- 0
  BTree t(3)
  print "Nodes to be inserted:"
  xx <- value
  print "Enter value:"
  for (int i <- 0, i < n, i++)
  {
    xx <- value
      t.insert(n)
  }
    print: Traversal of tree constructed
    t.traverse()
  return 0
}
```