Date
Page

Arpana M R
1BM18CS147
Arpana

PROGRAM 06

11/11/20

Q) Write a program to implement insertion operation on a B-tree

```
class BTreeN{
    int *values;
    int t, n;
    BTreeN **C;
    bool leaf;
public:
    BTreeN(int _t, bool _leaf);
    void insertNonFull(int k);
    void splitChild(int i, BTreeN *y);
    void traverse();
    BTreeN *search(int k);
friend class BTree;
};

class BTree{
    BTree(int t=0)
    BTreeN *root;
    int t;
public:
    BTree(int _t)
    { root= NULL;
      t = _t; }
```

Date
Page

Arjana MR
1BM18CS044

```
void traverse()
{ if (root != NULL) root->traverse(); }

BTreeN* search(int k)
{ return (root == NULL)? NULL : root->search(k); }
  void insert(int k);
};

BTreeN::BTreeN(int t1, bool leaf1)
{
  t = t1; leaf = leaf1;
  n = 0;
  values = new int[2*t-1];
  C = new BTreeN *[2*t]; }

void BTreeN::traverse()
{
  for (i=0; i<n; i++)
  {
    if (leaf == false) C[i]->traverse();
    cout << " " << values[i];
  }

  if (leaf == false) C[i]->traverse();
}
```

Aayush MR
1BM18CS147

```
BTreeN *BTreeN::search(int k)
{
  while(i<n && k>values[i]) i++;
  if (values[i]==k) return this;
  if (leaf==true) return NULL;

  return C[i]->search(k); }

void BTree::insert(int k)
{
  if (root==NULL)
  { root=new BTreeN(t,true);
    root->values[0]=k;
    root->n=1; }
  else
  {
    if(root->n==2*t-1) {
    BTreeN *S = new BTreeN(t,false);
    S->C[0]=root; S->splitchild(0,root);
    if (S->values[0]<k) i++;
    S->C[i]->insertNonFull(k);
    root=S;
    else
      root->insertNonFull(k); }}
```

Arpana M R
1BM18CS147
Arpana

```
void BTree::insertNonFull(int k)
{
  i = n-1;
  if (leaf == true)
  { while (i>=0 && values[i] >k)
    {
      values[i+1] = values[i];
      i--; }

    values[i+1] = k;
    n = n+1; }
  else {
    while (i>=0 && values[i]>k)
    { i--; }
    if (C[i+1]->n == 2*t-1)
    { splitChild(i+1, C[i+1]);
      if (values[i+1]<k) i++;
    }
    C[i+1]->insertNonFull(k);
  }
}

void BTreeN::splitChild(int i, BTreeN *y)
{ BTreeN *z = new BTreeN(y->t, y->leaf);
  z->n = t-1;
```

```
for(j=0; j<t-1; j++) z->values[j]=y->keys[j];
if(y->leaf==false)
{
  for(j=0; j<t; j++) z->C[j]=y->C[j+t];
}
y->n=t-1;

for(j=h; j>=i+1; j--) C[j+1]=C[j];
C[i+1]=z;
for(j=n-1; j>=i; j--) values[j+1]=values[j];

values[i]=y->values[t-1];
n=n+1;}
```