class BTree

{

BTreeNode \*root;

int internalSize;

int leafSize;

public:

BTree(int ISize, int LSize);

void insert(int value);

void print();

}; // BTree class

class BTreeNode

{

protected:

int count;

int leafSize;

InternalNode \*parent;

BTreeNode \*leftSibling;

BTreeNode \*rightSibling;

public:

BTreeNode(int LSize, InternalNode \*p, BTreeNode \*left, BTreeNode \*right);

virtual ~BTreeNode() {}

int getCount() const;

BTreeNode\* getLeftSibling();

virtual int getMinimum()const = 0;

BTreeNode\* getRightSibling();

virtual BTreeNode\* insert(int value) = 0;

virtual void print(Queue <BTreeNode\*> &queue) = 0;

void setLeftSibling(BTreeNode \*left);

void setParent(InternalNode \*p);

void setRightSibling(BTreeNode \*right);

}; //BTreeNode class

class InternalNode:public BTreeNode

{

int internalSize;

BTreeNode \*\*children;

int \*keys;

public:

InternalNode(int ISize, int LSize, InternalNode \*p,

BTreeNode \*left, BTreeNode \*right);

int getMinimum()const;

InternalNode\* insert(int value); // returns pointer to new InternalNode

// if it splits else NULL

void insert(BTreeNode \*oldRoot, BTreeNode \*node2); // if root splits use this

void insert(BTreeNode \*newNode); // from a sibling

void print(Queue <BTreeNode\*> &queue);

}; // InternalNode class

class LeafNode:public BTreeNode

{

int \*values;

public:

LeafNode(int LSize, InternalNode \*p, BTreeNode \*left,

BTreeNode \*right);

bool isFull() const;

int getMinimum() const;

LeafNode\* insert(int value); // returns pointer to new Leaf if splits

// else NULL

void print(Queue <BTreeNode\*> &queue);

}; //LeafNode class