Program:

```
#include <stdio.h>
#include <stdlib.h>
#define M 4
struct BTreeNode
{
   int num_keys;
   int keys[M-1];
   struct BTreeNode *children[M];
   int is_leaf;
};

struct BTreeNode *createNode(int is_leaf)
 {
   int i;
   struct BTreeNode *newNode = (struct BTreeNode *)malloc(sizeof(struct BTreeNode));
   if (newNode == NULL) {
          perror("Memory allocation failed");
          exit(EXIT_FAILURE);
   }
   newNode->num_keys = 0;
   newNode->is_leaf = is_leaf;
   for (i = 0; i < M; i++) {
          newNode->children[i] = NULL;
   }
   return newNode;
}

void splitChild(struct BTreeNode *parent, int index)
{
   int i;
   struct BTreeNode *child = parent->children[index];
   struct BTreeNode *newNode = createNode(child->is_leaf);
   newNode->num_keys = M/2 - 1;
   for (i = 0; i < M/2 - 1; i++)
   {
          newNode->keys[i] = child->keys[i + M/2];
   }
   if (!child->is_leaf)
   {
          for (i = 0; i < M/2; i++)
          {
             newNode->children[i] = child->children[i + M/2];
          }
   }
   child->num_keys = M/2 - 1;
   for (i = parent->num_keys; i > index; i--) {
          parent->children[i + 1] = parent->children[i];
```

```
    }
    parent->children[index + 1] = newNode;
    for (i = parent->num_keys - 1; i >= index; i--) {
            parent->keys[i + 1] = parent->keys[i];
    }
    parent->keys[index] = child->keys[M/2 - 1];
    parent->num_keys++;
}

void insertNonFull(struct BTreeNode *node, int key)
{
    int i = node->num_keys - 1;
    if (node->is_leaf)
    {
            while (i >= 0 && node->keys[i] > key)
            {
               node->keys[i + 1] = node->keys[i];
               i--;
            }
            node->keys[i + 1] = key;
            node->num_keys++;
    }
    else
    {
            while (i >= 0 && node->keys[i] > key)
            {
               i--;
            }
            i++;
            if (node->children[i]->num_keys == M - 1)
            {
               splitChild(node, i);
               if (node->keys[i] < key)
               {
                       i++;
               }
            }
            insertNonFull(node->children[i], key);
    }
}

void insert(struct BTreeNode **root, int key)
{
    struct BTreeNode *node = *root;

    if (node == NULL)
    {
            *root = createNode(1);
            (*root)->keys[0] = key;
            (*root)->num_keys = 1;
    }
```

```
    else
    {
        if (node->num_keys == M - 1)
        {
            struct BTreeNode *new_root = createNode(0);
            new_root->children[0] = node;
            splitChild(new_root, 0);
            *root = new_root;
        }
        insertNonFull(*root, key);
    }
}

void traverse(struct BTreeNode *root)
{
    if (root != NULL) {
        int i;
        for (i = 0; i < root->num_keys; i++)
        {
            traverse(root->children[i]);
            printf("%d ", root->keys[i]);
        }
        traverse(root->children[i]);
    }
}

void main()
{
    struct BTreeNode *root = NULL;
    insert(&root, 10);
    insert(&root, 20);
    insert(&root, 5);
    insert(&root, 6);
    insert(&root, 12);
    insert(&root, 30);
    printf("In-order traversal of the B-tree: ");
    traverse(root);
    printf("\n");
    getch();
}
```