```
#include<stdio.h>
#include<stdlib.h>
#define MAX 3
#define MIN 2

struct BTreeNode
{
    int val[MAX + 1], count;
    struct BTreeNode *link[MAX + 1];
};

typedef struct BTreeNode BTreeNode;

BTreeNode *createNode(BTreeNode *root, int val, BTreeNode *child)
{
    BTreeNode *newNode;
    newNode = (BTreeNode *)malloc(sizeof(BTreeNode));
    newNode -> val[1] = val;
    newNode -> count = 1;
    newNode -> link[0] = root;
    newNode -> link[1] = child;
    return newNode;
}

void insertNode(int val, int pos, BTreeNode *node, BTreeNode *child)
{
    int j = node -> count;
    while (j > pos)
    {
        node -> val[j + 1] = node -> val[j];
        node -> link[j + 1] = node -> link[j];
        j--;
    }

    node -> val[j + 1] = val;
    node -> link[j + 1] = child;
    node -> count++;
}

void splitNode(int val, int *pval, int pos, BTreeNode *node, BTreeNode *child,
BTreeNode **newNode)
{
    int median, j;

    if(pos > MIN)
        median = MIN + 1;
    else
        median = MIN;

    *newNode = (BTreeNode *)malloc(sizeof(BTreeNode));
```

```
    j = median + 1;

    while(j <= MAX)
    {
        (*newNode) -> val[j - median] = node -> val[j];
        (*newNode) -> link[j - median] = node -> link[j];
        j++;
    }

    node -> count = median;
    (*newNode) -> count = MAX - median;

    if(pos <= MIN)
    {
        insertNode(val, pos, node, child);
    }
    else
    {
        insertNode(val, pos - median, *newNode, child);
    }

    *pval = node -> val[node -> count];
    (*newNode) -> link[0] = node -> link[node -> count];
    node -> count--;
}

int setValue(int val, int *pval, BTreeNode *node, BTreeNode **child)
{
    int pos;

    if(!node)
    {
        *pval = val;
        *child = NULL;
        return 1;
    }

    if(val < node -> val[1])
    {
        pos = 0;
    }
    else
    {
        for(pos = node -> count; (val < node -> val[pos] && pos > 1); pos--);

        if(val == node -> val[pos])
        {
            printf("\nSorry, duplicates are not permitted\n");
            return 0;
        }
```

```
    }

    if(setValue(val, pval, node -> link[pos], child))
    {
        if(node -> count < MAX)
        {
            insertNode(*pval, pos, node, *child);
        }
        else
        {
            splitNode(*pval, pval, pos, node, *child, child);
            return 1;
        }
    }
    return 0;
}

BTreeNode  *Insert(BTreeNode *root, int val)
{
    int flag, i;
    BTreeNode *child;

    flag = setValue(val, &i, root, &child);
    if(flag)
        root = createNode(root, i, child);

    return root;
}

BTreeNode *Create(BTreeNode *root)
{
    int num, i, ele;
    printf("\n Enter the number of elements:");
    scanf("%d", &num );
    printf("\n Enter elements:");
    for( i = 0; i < num; i++)
    {
        scanf("%d", &ele);
        root = Insert(root, ele);
    }

    return root;
}

void search(int val, int *pos, BTreeNode *myNode)
{
    if(!myNode)
    {
        return;
    }
```

```
    if(val < myNode -> val[1])
    {
        *pos = 0;
    }
    else
    {
        for(*pos = myNode -> count; (val < myNode -> val[*pos] && *pos > 1);
(*pos)--);

        if(val == myNode -> val[*pos])
        {
            printf("\nThe element %d is present in the B - Tree\n", val);
            return;
        }
    }

    search(val, pos, myNode -> link[*pos]);

    return;
}

void displayTree(BTreeNode *myNode)
{
    int i;

    if(myNode)
    {
        for(i = 0; i < myNode -> count; i++)
        {
            displayTree(myNode -> link[i]);
            printf("%d ", myNode -> val[i + 1]);
        }
        displayTree(myNode -> link[i]);
    }
}

int main()
{
        BTreeNode *root = NULL;
        int pos;
        int ele;
        int e = 1, ch;

        while( e )
        {
                printf( "\n--------------MENU--------------\n" );
                printf( "\n\t1. Create\n\t2. Insert\n\t3. Display\n\t4.
Search\n\t5. Exit\n" );
                printf( "\n--------------------------------\n" );
```

```
                printf( "\n Enter your choice:" );
                scanf( "%d", &ch );

                switch( ch )
                {
                        case 1: root = Create( root );
                             break;
                        case 2:
                                printf("\n Enter the element to insert:");
                                scanf("%d", &ele);
                                root = Insert( root, ele);
                                 break;
                        case 3: displayTree( root );
                                 break;
                        case 4:
                                printf("\n Enter the element to search :");
                                scanf("%d", &ele);
                                search(ele, &pos, root);
                                 break;
                    case 5: e = 0;
                                 break;
                        default: printf( "\n Invalid choice \n" );
                }

        }
        return 0;
}
```