2. B-Tree (order 5) with insertion, search (basic)

Description: Simplified B-Tree of minimum degree t=3 gives order up to 5 (max keys = 2*t-1 = 5). Includes insert and search. Deletion is complex; omitted for brevity.

```c
// btree.c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define T 3 // minimum degree -> max keys = 2*T-1 = 5

typedef struct BNode {
    int keys[2*T-1];
    struct BNode *C[2*T];
    int n;
    bool leaf;
} BNode;

BNode* createNode(bool leaf){
    BNode* node = (BNode*)malloc(sizeof(BNode));
    node->leaf = leaf; node->n = 0;
    for(int i=0;i<2*T;i++) node->C[i]=NULL;
    return node;
}

void traverse(BNode* root){
    if(!root) return;
    int i;
    for(i=0;i<root->n;i++){
        if(!root->leaf) traverse(root->C[i]);
        printf("%d ", root->keys[i]);
    }
    if(!root->leaf) traverse(root->C[i]);
}

BNode* search(BNode* root, int k){
    int i=0; while(i<root->n && k>root->keys[i]) i++;
    if(i<root->n && root->keys[i]==k) return root;
    if(root->leaf) return NULL;
    return search(root->C[i], k);
}

void splitChild(BNode* x, int i){
    BNode* y = x->C[i];
    BNode* z = createNode(y->leaf);
    z->n = T-1;
    for(int j=0;j<T-1;j++) z->keys[j] = y->keys[j+T];
    if(!y->leaf) for(int j=0;j<T;j++) z->C[j] = y->C[j+T];
    y->n = T-1;
    for(int j=x->n;j>=i+1;j--) x->C[j+1] = x->C[j];
    x->C[i+1] = z;
    for(int j=x->n-1;j>=i;j--) x->keys[j+1] = x->keys[j];
    x->keys[i] = y->keys[T-1];
    x->n += 1;
}

void insertNonFull(BNode* x, int k){
    int i = x->n - 1;
    if(x->leaf){
        while(i>=0 && x->keys[i]>k){ x->keys[i+1] = x->keys[i]; i--; }
        x->keys[i+1] = k; x->n += 1;
    } else {
        while(i>=0 && x->keys[i]>k) i--;
        i++;
        if(x->C[i]->n == 2*T-1){ splitChild(x,i); if(k > x->keys[i]) i++; }
        insertNonFull(x->C[i], k);
    }
}

BNode* insert(BNode* root, int k){
    if(!root){ root = createNode(true); root->keys[0]=k; root->n=1; return
root; }
    if(root->n == 2*T-1){
        BNode* s = createNode(false); s->C[0]=root; splitChild(s,0);
        int i=0; if(s->keys[0]<k) i++; insertNonFull(s->C[i], k); return s;
    } else { insertNonFull(root, k); return root; }
}

int main(){
    BNode* root = NULL; int n, x;
    printf("Number of keys to insert: "); scanf("%d", &n);
    for(int i=0;i<n;i++){ scanf("%d", &x); root = insert(root,x); }
    printf("Traversal of B-Tree: "); traverse(root); printf("\n");
    printf("Search key: "); scanf("%d", &x);
    BNode* res = root? search(root,x):NULL;
    if(res) printf("Key found.\n"); else printf("Not found.\n");
    return 0;
}
```