

```
main.c
1 // Searching a key on a B-tree in C
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 #define MAX 3
7 #define MIN 2
8
9 * struct BTreenode {
10     int val[MAX - 1], count;
11     struct BTreenode *link[MAX - 1];
12 };
13
14 struct BTreenode *root;
15
16 // Create a node
17 * struct BTreenode *createNode( int val, struct BTreenode *child) {
18     struct BTreenode *newNode;
19     newNode = (struct BTreenode *)malloc(sizeof(struct BTreenode));
20     newNode->val[1] = val;
21     newNode->count = 1;
22     newNode->link[0] = root;
23     newNode->link[1] = child;
24     return newNode;
25 }
26
27 // Insert node
28 - void insertNode( int val, int pos, struct BTreenode *node,
```

Output

```
/tmp/o5fQ2d0Vix.o
8 9 10 11 15 16 17 18 20 23
11 is found
```

==== Code Execution Successful =====

Google Chrome isn't your default browser

Set as default



C Online Compiler

main.c

Run Share Output

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_KEYS 3
5 #define MIN_KEYS 1
6
7 // Define the structure for a 2-3-4 tree node
8
9     int keys[MAX_KEYS];
10
11 struct Node *children[MAX_KEYS + 1];
12
13 } Node;
```

```
/tmp/t42l9wZdQV.o
In-order traversal of the 2-3-4 tree:
5 6 10 15 20 25 30
== Code Execution Successful ==
```

```
14
15 // Function to create a new 2-3-4 tree node
16 Node* createnode(int isleaf) {
17     Node* node = (Node*)malloc(sizeof(Node));
18     node->nunKeys = 0;
19     node->isLeaf = isLeaf;
20
21     for (int i = 0; i < MAX_KEYS + 1; i++) {
22         node->children[i] = NULL;
23     }
24
25
26 // Function to traverse the 2-3-4 tree in order
27 void traverse(Node* root) {
28     if (root == NULL) return;
```

C Online Compiler

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_KEYS 4
5 #define MIN_KEYS 1
6
7 // Define the structure for a 2-3-4-5 tree node
8 typedef struct Node {
9     int keys[MAX_KEYS];
10    struct Node *children[MAX_KEYS - 1];
11    int numKeys;
12    int isLeaf;
13 } Node;
14
15 // Function to create a new 2-3-4-5 tree node
16 Node* createNode(int isLeaf) {
17     Node* node = (Node*)malloc(sizeof(Node));
18     node->numKeys = 0;
19     node->isLeaf = isLeaf;
20     for (int i = 0; i < MAX_KEYS - 1; i++) {
21         node->children[i] = NULL;
22     }
23     return node;
24 }
25
26 // Function to traverse the 2-3-4-5 tree in order
27 void traverse(Node* root) {
28     if (root == NULL) return;
29
30     // In-order traversal of the 2-3-4-5 tree:
31     // 5 6 10 15 20 25 30
32 }
```

Output

```
/tmp/I42L9wZdQV.o
In-order traversal of the 2-3-4-5 tree:
5 6 10 15 20 25 30
== Code Execution Successful ==
```

9:40 AM 7/31/2024

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_KEYS 2
5 #define MIN_KEYS 1
6
7 typedef struct Node {
8     int keys[MAX_KEYS - 1];
9     struct Node *children[MAX_KEYS - 2];
10    int numKeys;
11    int isLeaf;
12 } Node;
13
14 // Function to create a new node
15 Node* createNode(int isLeaf) {
16     Node* node = (Node*)malloc(sizeof(Node));
17     node->numKeys = 0;
18     node->isLeaf = isLeaf;
19     for (int i = 0; i < MAX_KEYS - 2; i++) {
20         node->children[i] = NULL;
21     }
22     return node;
23 }
24
25 // Function to traverse the B-tree in order
26 void traverse(Node* root) {
27     if (root == NULL) return;
28 }
```

Run Share

Run

Output

```
/tmp/nimKTd1oDt.o
Segmentation fault
```

== Code Exited With Errors ==

ENG IN