```
//Created By Ritwik Chandra Pandey on 3 Nov' 2021
//B-tree: Insertion, Traversal

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 4
#define MIN 2


struct BTreeNode {
    int val[MAX + 1], count;
    struct BTreeNode *link[MAX + 1];
};
typedef struct BTreeNode* BTNODE;
BTNODE root;
/* creating new node */
BTNODE createNode(int val,BTNODE child) {
    BTNODE newNode = (BTNODE)malloc(sizeof(struct BTreeNode));
    newNode->val[1] = val;
    newNode->count = 1;
    newNode->link[0] = root;
    newNode->link[1] = child;
    return newNode;

}
//Fill the code in below functions. Please refer to the psuedo code.
void addValToNode(int val, int pos, BTNODE node, BTNODE 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, BTNODE node,BTNODE child, BTNODE *newNode) {
        int median, j;
```

```
        if(pos>MIN)
         median = MIN + 1;
        else
         median = MIN;
        *newNode = (BTNODE)malloc(sizeof(struct 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) {
                addValToNode(val, pos, node, child);
        }else{
                addValToNode(val,pos - median, *newNode,child);
        }
        *pval = node->val[node->count];
        (*newNode)->link[0] = node->link[node->count];
        node->count--;
}
int setValueInNode(int val, int *pval,BTNODE node,BTNODE* 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("Duplicates not allowed.\n");
                        return 0;
                }

        }
        if(setValueInNode(val,pval,node->link[pos],child)){
```

```
                if(node->count < MAX){
                        addValToNode(*pval,pos,node,*child);
                }
                else{
                        splitNode(*pval, pval,pos, node, *child, child);
                        return 1;
                }
        }
        return 0;
}
void insertNodeInBTree(int val) {
        int flag, i;
        BTNODE child;
        flag = setValueInNode(val, &i, root, &child);
        if(flag){
                root = createNode(i, child);
        }
}
void traverseBTree(BTNODE myNode) {
        int i;
        if(myNode){
                for(i=0; i< myNode->count; i++){
                        traverseBTree(myNode->link[i]);
                        printf("%d ", myNode->val[i+1]);
                }
                traverseBTree(myNode->link[i]);

        }

}



int main() {
        int ele, op, pos;
        while(1)
        {
                printf("1.Insert 2.Traversal 3.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
```

```
		switch(op) {
			case 1:printf("Enter an element to be inserted : ");
				scanf("%d", &ele);
				insertNodeInBTree(ele);
				break;
			case 2:
				if(root == NULL) {
					printf("B-Tree is empty.\n");
				}
				else {
					printf("Elements of the B-Tree : ");
					traverseBTree(root);
					printf("\n");
				}
				break;
			case 3:exit(0);
		}
	}
}
```