

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  //?????? ??? ????? ?? char.
5  typedef char BinTreeElementType;
6
7  typedef struct BinTreeNode *BinTreePointer;
8  struct BinTreeNode {
9      BinTreeElementType Data;
10     BinTreePointer LChild, RChild;
11 } ;
12
13 typedef enum {
14     FALSE, TRUE
15 } boolean;
16
17
18 void CreateBST(BinTreePointer *Root);
19 boolean EmptyBST(BinTreePointer Root);
20 void BSTInsert(BinTreePointer *Root, BinTreeElementType Item);
21 void BSTSearch(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr);
22 void BSTSearch2(BinTreePointer Root, BinTreeElementType Item, boolean *Found, BinTreePointer *LocPtr,
BinTreePointer *Parent);
23 void BSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue);
24 void InorderTraversal(BinTreePointer Root);
25 int BSTLevel(BinTreeElementType Item, BinTreePointer Root);
26
27 int main()
28 {
29     //?????? ??????????.
30     BinTreePointer ARoot;
31     int i, j, level;
32     boolean found;
33     char str[] = "PROCEDURE";
34
35     CreateBST(&ARoot);
36
37     //???????? ??? ??? ??? ?????????? ??? ????.
38     for(i=0; str[i] != '\0'; i++)
39         BSTInsert(&ARoot, str[i]);
40
41     //???????? ??? level ??? ?????????? ??? ?????? ??? ???.
42     for(i=0; str[i] != '\0'; i++)
43     {
44         //???????? ??? ?? ?????????? ??? ??? ??? ??? ??????.
45         found = FALSE;
46         for(j=0; j < i; j++)
47         {
48             if(str[i] == str[j])
49             {
50                 found = TRUE;
51                 break;
52             }
53         }
54         if(!found)
55         {
56             level = BSTLevel(str[i], ARoot);
57             printf("Level of %c: %d\n", str[i], level);
58         }
59     }
60
61     return 0;
62 }
63
64
65 void CreateBST(BinTreePointer *Root)

```

```

66 {
67     *Root = NULL;
68 }
69
70 boolean EmptyBST(BinTreePointer Root)
71 {
72     return (Root==NULL);
73 }
74
75 void BSTInsert(BinTreePointer *Root, BinTreeElementType Item)
76 {
77     BinTreePointer LocPtr, Parent;
78     boolean Found;
79
80     LocPtr = *Root;
81     Parent = NULL;
82     Found = FALSE;
83     while (!Found && LocPtr != NULL) {
84         Parent = LocPtr;
85         if (Item < LocPtr->Data)
86             LocPtr = LocPtr ->LChild;
87         else if (Item > LocPtr ->Data)
88             LocPtr = LocPtr ->RChild;
89         else
90             Found = TRUE;
91     }
92     if (Found)
93         printf("To %c EINAI HDH STO DDA\n", Item);
94     else {
95         LocPtr = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
96         LocPtr ->Data = Item;
97         LocPtr ->LChild = NULL;
98         LocPtr ->RChild = NULL;
99         if (Parent == NULL)
100             *Root = LocPtr;
101         else if (Item < Parent ->Data)
102             Parent ->LChild = LocPtr;
103         else
104             Parent ->RChild = LocPtr;
105     }
106 }
107
108 void BSTSearch(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr)
109 {
110
111     (*LocPtr) = Root;
112     (*Found) = FALSE;
113     while (!(*Found) && (*LocPtr) != NULL)
114     {
115         if (KeyValue < (*LocPtr)->Data)
116             (*LocPtr) = (*LocPtr)->LChild;
117         else
118             if (KeyValue > (*LocPtr)->Data)
119                 (*LocPtr) = (*LocPtr)->RChild;
120             else (*Found) = TRUE;
121     }
122 }
123
124 void BSTSearch2(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr,
BinTreePointer *Parent)
125 {
126     *LocPtr = Root;
127     *Parent=NULL;
128     *Found = FALSE;
129     while (!(*Found) && *LocPtr != NULL)
130     {

```

```

131         if (KeyValue < (*LocPtr)->Data) {
132             *Parent=*LocPtr;
133             *LocPtr = (*LocPtr)->LChild;
134         }
135         else
136             if (KeyValue > (*LocPtr)->Data) {
137                 *Parent=*LocPtr;
138                 *LocPtr = (*LocPtr)->RChild;
139             }
140             else *Found = TRUE;
141     }
142
143 }
144
145 void BSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
146 {
147
148     BinTreePointer
149     n,
150     Parent,
151     nNext,
152     SubTree;
153     boolean Found;
154
155     BSTSearch2(*Root, KeyValue, &Found , &n, &Parent);
156     if (!Found)
157         printf("TO STOIXEIO %d DEN EINAI STO DDA\n", KeyValue);
158     else {
159         if (n->LChild != NULL && n->RChild != NULL)
160             {
161                 nNext = n->RChild;
162                 Parent = n;
163                 while (nNext->LChild !=NULL)
164                     {
165                         Parent = nNext;
166                         nNext = nNext->LChild;
167                     }
168                 n->Data = nNext->Data;
169                 n = nNext;
170             }
171         SubTree = n->LChild;
172         if (SubTree == NULL)
173             SubTree = n->RChild;
174         if (Parent == NULL)
175             *Root = SubTree;
176         else if (Parent->LChild == n)
177             Parent->LChild = SubTree;
178         else
179             Parent->RChild = SubTree;
180         free(n);
181     }
182 }
183
184 void InorderTraversal(BinTreePointer Root)
185 {
186     if (Root!=NULL) {
187         InorderTraversal(Root->LChild);
188         printf("%c ",Root->Data);
189         InorderTraversal(Root->RChild);
190     }
191 }
192
193 int BSTLevel(BinTreeElementType Item,BinTreePointer Root)
194 {
195     //?????? ??????????.
196     boolean Found = FALSE;

```

```

197     int level = 1;
198
199     //???? ?????????? ?????? ?? ?????? ?? ????????? ? ?????? ?? "?????????" ?? ???
200     while(!(Found) && (Root != NULL))
201     {
202         /*????????? ??? ?????????? ?? ??? ???? ?????? ASCII ?????????????????? ??? ???
203         ?????????? ??? ???? BSTSearch.*/
204         if(Root->Data == Item)
205             Found=TRUE;
206         else
207         {
208             if(Item < Root->Data)
209                 Root = Root->LChild;
210             else
211                 Root = Root->RChild;
212
213             level++;
214         }
215     }
216
217     //????????? ?????? ?????????? ?? ????????? ? ??? ?? ?????????? ??? ???
218     if(Found)
219         return level;
220     else
221         return -1;
222 }

```