```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{
    char name[20];
    int code;
} BinTreeElementType;

typedef struct BinTreeNode *BinTreePointer;
 struct BinTreeNode {
    BinTreeElementType Data;
    BinTreePointer LChild, RChild;
} ;

typedef enum {
    FALSE, TRUE
} boolean;


void CreateBST(BinTreePointer *Root);
boolean EmptyBST(BinTreePointer Root);
void BSTInsert(BinTreePointer *Root, BinTreeElementType Item);
void BSTSearch(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr);
void BSTSearch2(BinTreePointer Root, BinTreeElementType Item, boolean *Found,BinTreePointer *LocPtr,
BinTreePointer *Parent);
void BSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue);
void InorderTraversal(BinTreePointer Root);
void BuildBST(BinTreePointer *Root);
void InorderTraversalPart(BinTreePointer Root, int code);

int main()
{
    BinTreePointer ARoot,LocPtr;
    BinTreeElementType EmpRec;
    boolean found;
    int i;

    //1. Create BST
    BuildBST(&ARoot);

    //2. Insert employees
    for(i = 0; i < 9; i++)
    {
       if(i == 0)
       {
            printf("Give the data for Office Employees.\n");
            printf("-----------------------------------\n");
       }
       else if(i == 3)
       {
            printf("\nGive the data for the Factory Employees.\n");
            printf("----------------------------------------\n");
       }
       else if(i == 6)
       {
            printf("\nGive the data for the Sales Representatives.\n");
            printf("-------------------------------------------\n");
       }

       printf("\nGive the name:");
       fgets(EmpRec.name,20,stdin);
       EmpRec.name[strlen(EmpRec.name)-1]='\0';

       printf("Give the code:");
```

```
66          scanf("%d",&EmpRec.code);
67          getchar();
68
69          BSTInsert(&ARoot,EmpRec);
70      }
71
72      //3. Search for an employee by name.
73      printf("\n----------------------\n");
74      printf("Give a name for search:");
75      fgets(EmpRec.name,20,stdin);
76      EmpRec.name[strlen(EmpRec.name)-1]='\0';
77
78      BSTSearch(ARoot,EmpRec,&found,&LocPtr);
79      printf("\n-----Search Result-----\n");
80
81      if(found)
82      {
83          printf("Name:%s\n",LocPtr->Data.name);
84          printf("Code:%d\n",LocPtr->Data.code);
85      }
86      else
87          printf("Employee not Found\n");
88
89      //4. List all employees.
90      printf("\n----------------------\n");
91      printf("List of all Empolyees.\n");
92      printf("----------------------\n");
93      InorderTraversal(ARoot);
94
95      //5. List office employees.
96      printf("\n-------------------------\n");
97      printf("List of office Empolyees.\n");
98      printf("-------------------------\n");
99
100     InorderTraversalPart(ARoot,1);
101
102     //6. List factory employees.
103     printf("\n-------------------------\n");
104     printf("List of factory Empolyees.\n");
105     printf("-------------------------\n");
106
107     InorderTraversalPart(ARoot,2);
108
109     //7. List sale representatives.
110     printf("\n----------------------------\n");
111     printf("List of Sale Representatives.\n");
112     printf("----------------------------\n");
113
114     InorderTraversalPart(ARoot,3);
115
116     //8. Delete an employee record.
117     printf("----------------------\n");
118     printf("Give a name to delete:");
119     fgets(EmpRec.name,20,stdin);
120     EmpRec.name[strlen(EmpRec.name)-1]='\0';
121
122     BSTDelete(&ARoot,EmpRec);
123
124     return 0;
125 }
126
127 void CreateBST(BinTreePointer *Root)
128 {
129     *Root = NULL;
130 }
131
```

```
boolean EmptyBST(BinTreePointer Root)
{
    return (Root==NULL);
}

void BSTInsert(BinTreePointer *Root, BinTreeElementType Item)
{
    BinTreePointer LocPtr, Parent;
    boolean Found;

    LocPtr = *Root;
    Parent = NULL;
    Found = FALSE;
    while (!Found && LocPtr != NULL) {
        Parent = LocPtr;
        if (strcmp(Item.name,LocPtr->Data.name) < 0)
            LocPtr = LocPtr ->LChild;
        else if (strcmp(Item.name,LocPtr->Data.name) > 0)
            LocPtr = LocPtr ->RChild;
        else
            Found = TRUE;
    }
    if (Found)
        printf("To %s EINAI HDH STO DDA\n", Item.name);
    else {
        LocPtr = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
        LocPtr ->Data = Item;
        LocPtr ->LChild = NULL;
        LocPtr ->RChild = NULL;
        if (Parent == NULL)
            *Root = LocPtr;
        else if (strcmp(Item.name,Parent->Data.name) < 0)
            Parent ->LChild = LocPtr;
        else
            Parent ->RChild = LocPtr;
    }
}

void BSTSearch(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr)

{

    (*LocPtr) = Root;
    (*Found) = FALSE;
    while (!(*Found) && (*LocPtr) != NULL)
    {
        if (strcmp(KeyValue.name,(*LocPtr)->Data.name) < 0)
            (*LocPtr) = (*LocPtr)->LChild;
        else
            if (strcmp(KeyValue.name,(*LocPtr)->Data.name) > 0)
                (*LocPtr) = (*LocPtr)->RChild;
            else (*Found) = TRUE;
    }
}

void BSTSearch2(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr,
BinTreePointer *Parent)
{
    *LocPtr = Root;
    *Parent=NULL;
    *Found = FALSE;
    while (!(*Found) && *LocPtr != NULL)
    {
        if (strcmp(KeyValue.name,(*LocPtr)->Data.name) < 0)
        {
            *Parent=*LocPtr;
```

```
            *LocPtr = (*LocPtr)->LChild;
        }
        else
            if (strcmp(KeyValue.name,(*LocPtr)->Data.name) > 0)
            {
                *Parent=*LocPtr;
                *LocPtr = (*LocPtr)->RChild;
            }
            else *Found = TRUE;
    }

}

void BSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
{

   BinTreePointer
   n,                  //??????? ???? ????? ??? ???????? ??? ???? KeyValue *)
   Parent,             // ??????? ??? n ? ??? nNext
   nNext,              // ???????????????? ???????? ??? n
   SubTree;            // ??????? ???? ????????? ??? n
   boolean Found;      // TRUE AN TO ???????? KeyValue EINAI ??????? ??? ??? *)

     BSTSearch2(*Root, KeyValue, &Found , &n, &Parent);
     if (!Found)
        printf("TO STOIXEIO %s DEN EINAI STO DDA\n", KeyValue.name);
     else {
          if (n->LChild != NULL && n->RChild != NULL)
          {  // ?????? ???? ???????? ?? ??? ??????
                 //???? ??? ???????????????? ??????? ??? ??? ?????? ???
                 nNext = n->RChild;
                 Parent = n;
                 while (nNext->LChild !=NULL)  //* DIASXISH PROS TA ARISTERA *)
                 {
                      Parent = nNext;
                      nNext = nNext->LChild;
                 }
                /* ????????? ??? ???????????? ??? nNext ???? n ???
                ?????? ??? n ???? ?? ??????? ???? ??????? */
                n->Data = nNext->Data;
                n = nNext;
          } //??????????? ?? ??? ????????? ??? ? ?????? ???? ?? ???? 1 ?????
          SubTree = n->LChild;
          if (SubTree == NULL)
             SubTree = n->RChild;
          if (Parent == NULL)                //* 8A DIAGRAFEI H RIZA *)
             *Root = SubTree;
          else if (Parent->LChild == n)
                  Parent->LChild = SubTree;
               else
                   Parent->RChild = SubTree;
          free(n);
     }
}

void InorderTraversal(BinTreePointer Root)
{
    if (Root!=NULL) {
        InorderTraversal(Root->LChild);
        printf("%s \n",Root->Data.name);
        InorderTraversal(Root->RChild);
    }
}

void BuildBST(BinTreePointer *Root)
{
```

```
263     FILE *input;
264     int nscan;
265     BinTreeElementType EmpRec;
266
267
268     CreateBST(&(*Root));
269     input = fopen("I12F5.TXT","r");
270
271     if(input == NULL)
272     {
273         printf("Problem opening file.\n");
274     }
275     else
276         while(TRUE)
277         {
278             nscan = fscanf(input,"%20[^,],%d\n",EmpRec.name,&EmpRec.code);
279             if(nscan == EOF) break;
280             if(nscan != 2) printf("Error\n");
281
282
283             BSTInsert(&(*Root),EmpRec);
284         }
285
286     fclose(input);
287 }
288
289 void InorderTraversalPart(BinTreePointer Root, int code)
290 {
291     if(!EmptyBST(Root))
292     {
293         InorderTraversalPart(Root->LChild,code);
294
295         if(Root->Data.code == code)
296         {
297             printf("Name:%s\n",Root->Data.name);
298             printf("Code:%d\n\n",Root->Data.code);
299         }
300
301         InorderTraversalPart(Root->RChild,code);
302     }
303 }
304
305
```