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

//?????? ??? ????? BinTreeElementType
typedef struct
{
    int code;
    int recNo;
} BinTreeElementType;

//?????????? ???????? student.
typedef struct
{
    int code;
    char surname[20];
    char name[20];
    char sex;
    int year;
    float grade;

}studentT;

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

typedef enum {
    FALSE, TRUE
} boolean;


void CreateBST(BinTreePointer *Root);
boolean BSTEmpty(BinTreePointer Root);
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item);
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr);
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue);
void RecBSTInorder(BinTreePointer Root);
void RecBSTPreorder(BinTreePointer Root);
void RecBSTPostorder(BinTreePointer Root);
void menu(int *choise);
int BuildBST(BinTreePointer *Root);
void InsertNewStudent(BinTreePointer *Root,int *size);
void SearchStudent(BinTreePointer Root);
void PrintStudent(int RecNum);
void PrintStudentsWithGrade();



int main()
{
    //?????? ??????????.
    BinTreePointer ARoot;
    int size,choise;

    //???????????????? ???????? ??? menou ????? ?? ?????????? ? ??????? ?? ??????????.
    while(TRUE)
    {
        menu(&choise);
        switch(choise)
        {
            //???? ????????? ??? ? ??????? ????? 1 ????? ??? ?????????? ??? ?????????? ???(index).
            case 1:
                size = BuildBST(&ARoot);
                break;
            //???? ????????? ??? ? ??????? ????? 2 ????? ??? ?????????? ??? ???????? ???? ??????? ??? ??? ??? ??? ??????.
            case 2:
                InsertNewStudent(&ARoot,&size);
                break;
            //???? ????????? ??? ? ??????? ????? 3 ????? ??? ?????????? ??? ????????? ???? ??????? ??? ???????? ??? ????????? ???.
            case 3:
                SearchStudent(ARoot);
                break;
            //???? ????????? ??? ? ??????? ????? 4 ????? ??? ?????????? ??? ???????? ???? ??? ???????? ??? ?? ???.
            case 4:
                printf("\nPrint all students data.\n\n");
                RecBSTInorder(ARoot);
                printf("\n");
                break;
            case 5:
            //???? ????????? ??? ? ??????? ????? 5 ????? ??? ?????????? ??? ???????? ????????? ???????? ?? ????? ?????????? ??? ??? ????.
                PrintStudentsWithGrade();
                break;
            //???? ????????? ??? ? ??????? ????? 6,????? ????????????.
            case 6:
                return 0;
        }
    }
}

void CreateBST(BinTreePointer *Root)
{
    *Root = NULL;
}

boolean BSTEmpty(BinTreePointer Root)
{
    return (Root==NULL);
}

void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item)
{
    if (BSTEmpty(*Root)) {
        (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
        (*Root) ->Data = Item;
        (*Root) ->LChild = NULL;
        (*Root) ->RChild = NULL;
    }
    else
        if (Item.code < (*Root)->Data.code)
            RecBSTInsert(&(*Root) ->LChild,Item);
        else if (Item.code > (*Root)->Data.code)
            RecBSTInsert(&(*Root) ->RChild,Item);
        else
            printf("To %d EINAI HDH STO DDA\n", Item.code);
}

void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,boolean *Found, BinTreePointer *LocPtr)
{
    if (BSTEmpty(Root))
        *Found=FALSE;
    else
        if (KeyValue.code < Root->Data.code)
            RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
        else
```

```
            if (KeyValue.code > Root->Data.code)
                RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
            else
                {
                    *Found = TRUE;
                    *LocPtr=Root;
                }
}

void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
{

   BinTreePointer TempPtr;          //* true AN TO STOIXEIO KeyValue EINAI STOIXEIO TOY DDA *)

    if (BSTEmpty(*Root))          //* ????? ?????? ?? KeyValue ?? ?? ?????? *)
        printf("to %d DeN BRE8HKe STO DDA\n", KeyValue.code);
     else
         //* ????????? ?????????? ??? ????? ??? ???????? ??? ???? KeyValue ??? ???????? ???
          if (KeyValue.code < (*Root)->Data.code)
             RecBSTDelete(&((*Root)->LChild), KeyValue);       //* ???????? ????????? *
          else
            if (KeyValue.code > (*Root)->Data.code)
                  RecBSTDelete(&((*Root)->RChild), KeyValue);   //* ????  ????????? *
            else                                       //* TO KeyValue ??????? ???????? ??? ?????? *)
                if ((*Root)->LChild ==NULL)
                 {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;      //* ??? ???? ???????? ????? *)
                      free(TempPtr);
                 }
                else if ((*Root)->RChild == NULL)
                   {
                        TempPtr = *Root;
                        *Root = (*Root)->LChild;      //* ???? ???????? ?????, ???? ??? ???? *)
                        free(TempPtr);
                   }
                   else                               //* ???? 2 ?????? *)
                   {
                        //* ?????? ??? INORDER ???????? ??? *)
                        TempPtr = (*Root)->RChild;
                        while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                        /* ?????????? ??? ???????? ??? ????? ??? ??????????
                        ??? ?????????? ??? ???????? ??? ???????? ?????? */
                        (*Root)->Data = TempPtr->Data;
                        RecBSTDelete(&((*Root)->RChild), (*Root)->Data);
                   }
}

void RecBSTInorder(BinTreePointer Root)
{
    if (Root!=NULL) {
        RecBSTInorder(Root->LChild);
        printf(" (%d, %d),",Root->Data.code,Root->Data.recNo);
        RecBSTInorder(Root->RChild);
    }

}

void RecBSTPreorder(BinTreePointer Root)
{
    if (Root!=NULL) {
        printf("/%d /",Root->Data.code);
        printf("L");
        RecBSTPreorder(Root->LChild);
        printf("R");
        RecBSTPreorder(Root->RChild);
    }
    printf("U");
}

void RecBSTPostorder(BinTreePointer Root)
{
    if (Root!=NULL) {
        printf("L");
        RecBSTPostorder(Root->LChild);
        printf("R");
        RecBSTPostorder(Root->RChild);
        printf("/%d /",Root->Data.code);
    }
    printf("U");
}

/*????????? menu ??? ????????? ??? ???????? ??? ???? ? ??????? ??? ??????????
??? ??????? ???? ???? ??? ?????????? ?????? choise.?? ? ???? ??? ?? ????? ???
????? ?????? 1-6 ,???????? ????????? ??????.*/
void menu(int *choise)
{
    printf("\n                       MENOY                     \n");
    printf("------------------------------------------------\n");
    printf("1. Build BST\n");
    printf("2. Insert new student\n");
    printf("3. Search for a student\n");
    printf("4. Print all students (Traverse Inorder)\n");
    printf("5. Print students with a >= given grade\n");
    printf("6. Quit\n");

    do{
        printf("\nEpilogh: ");
        scanf("%d",&(*choise));

        if(*choise < 1 || *choise > 6)
            printf("Number must be between 1-6.Try again.\n");
    }while(*choise < 1 || *choise > 6);
}

int BuildBST(BinTreePointer *Root)
{
    //?????? ??????????
    FILE *infile;
    studentT student;
    BinTreeElementType indexRec;
    int size,nscan;

    //?????????? ???.
    CreateBST(&(*Root));

    //??????? ??? ??????? ??? ????????.
    infile = fopen("foitites.dat","r");

    //???????????? ??? ??????? ????????
    size = 0;

    //?? ??????? ????? ?? ??????.
    if(infile != NULL)
        while(TRUE)
        {
            //???????? ??? ??????? ??????-?????? ??? ??????? ??? ????????? ??? ??????? ????? ??? ???????? Student.
            nscan = fscanf(infile,"%d, %20[^,], %20[^,], %c, %d, %f\n",&student.code, student.surname, student.name, &student.sex, &student.year, &student.grade);
            if(nscan == EOF) break;
            if(nscan != 6)
            {
                printf("Improper file format.\n");
                break;
            }
            //??????? ??? ?? ??? ??? ??????? ?????? ??? ????????? ? ??????? ??? ?????????(index).
            indexRec.code = student.code;
            indexRec.recNo = size;

            //???????? ??? ???.
            RecBSTInsert(&(*Root),indexRec);

            //?????? ??? ?????? ???? 1.
            size++;
        }
    //???????? ??? ???????.
    fclose(infile);

    //????????? ??? ???????.
    return size;
}

void InsertNewStudent(BinTreePointer *Root,int *size)
{
    //?????? ??????????.
    FILE *infile;
    studentT student;
    BinTreeElementType indexRec;
    boolean found;
    BinTreePointer LocPtr;

    //??????? ??? ??????? ??? ??????????.
    infile = fopen("foitites.dat","a");

    //???????? ??? ?? ???? ?????????.
    printf("Give students AM: ");
    scanf("%d",&indexRec.code);

    //???????? ??? ???.
    RecBSTSearch(*Root,indexRec,&found,&LocPtr);

    //?? ??? ???????
    if(!found)
    {
        //??????? ?? ???????? ???? ???????.
        student.code = indexRec.code;

        printf("Give student surname: ");
        scanf("%s",student.surname);

        printf("Give student name: ");
        scanf("%s",student.name);

        printf("Give student's registration year: ");
        scanf("%d",&student.year);

        printf("Give student's grade: ");
        scanf("%f",&student.grade);

        printf("Give student sex F/M: ");
        scanf(" %c",&student.sex);

        indexRec.recNo = *size;

        //???????? ??? ???.
        RecBSTInsert(&(*Root),indexRec);
        //???????? ??? ??????.
        fprintf(infile,"%d, %s, %s, %c, %d, %.1f\n", student.code, student.surname, student.name, student.sex, student.year, student.grade);

        //?????? ??? ?????? ???? 1.
        size++;
    }
    //?? ??? ??????? ???????? ???????? ?????????.
    else
        printf("Afto to AM yparxei hdh.\n");

    //???????? ??? ???????.
    fclose(infile);
}

void SearchStudent(BinTreePointer Root)
{
    //?????? ??????????.
    BinTreeElementType indexRec;
    BinTreePointer LocPtr;
    boolean found;

    //???????? ??? ?? ???? ?????????.
    printf("\nGive student's code: ");
    scanf("%d",&indexRec.code);
    printf("\n");

    //???????? ??? ???.
    RecBSTSearch(Root,indexRec,&found,&LocPtr);

    //?? ??????? ???????? ????????? ?? ??? ??????? ??? ?????????? PrintStudent.
    if(found)
        PrintStudent(LocPtr->Data.recNo);
    //?? ??? ??????? ???????? ???????? ?????????.
    else
        printf("Student NOT found.\n");
}

void PrintStudent(int RecNum)
{
    //?????? ??????????.
    FILE *infile;
    int nscan,lines;
    studentT student;

    //??????? ??? ??????? ??? ????????.
    infile = fopen("foitites.dat","r");

    //???????????? ??? ?????????? ??? ????? ??? ??????? ??? ???????????
    lines = 0;

    //?? ??????? ????? ?? ??????.
    if(infile != NULL)
    {

        while(lines <= RecNum)
        {
            //???????? ??? ??????? ??????-?????? ??? ??????? ??? ????????? ??? ??????? ????? ??? ???????? Student.
            nscan = fscanf(infile,"%d, %20[^,], %20[^,], %c, %d, %f\n",&student.code, student.surname, student.name, &student.sex, &student.year, &student.grade);
            if(nscan == EOF) break;
            if(nscan != 6)
            {
                printf("Improper file format.\n");
                break;
            }
            //?????? ??? ??????? ????.
            lines++;
        }
        //????? ?????? ???? ?????? ??? ??????? ???????? ??? ????? ????????? ??????? ???????? ??? ????????? ??? ???????.
        if(lines)
            printf("%d,%s,%s,%c,%d,%.1f\n",student.code,student.name,student.surname,student.sex,student.year,student.grade);
    }
    //???????? ??? ???????.
    fclose(infile);
}
void PrintStudentsWithGrade()
{
    //?????? ??????????.
    FILE *infile;
    int nscan;
    studentT student;
    float theGrade;

    //???????? ??? ?????? ?? ??? ????? ??????? ?? ???????????.
    printf("Give the grade: ");
    scanf("%f",&theGrade);

    //??????? ??? ??????? ??? ????????.
    infile = fopen("foitites.dat","r");

    //??????????? ??????.
    printf("Student's with grade >= %.1f\n\n",theGrade);

    //?? ??????? ????? ?? ??????.
    if(infile != NULL)
        while(TRUE)
        {
            //???????? ??? ??????? ??????-?????? ??? ??????? ??? ????????? ??? ??????? ????? ??? ???????? Student.
            nscan = fscanf(infile,"%d, %20[^,], %20[^,], %c, %d, %f\n",&student.code, student.surname, student.name, &student.sex, &student.year, &student.grade);
            if(nscan == EOF) break;
            if(nscan != 6)
            {
                printf("Improper file format.\n");
                break;
            }
            //?? ? ??????? ??? ?????????? ???? >= ????? ??? ????? ??? ?????? ?? ??????????? ?? ???????? ???.
            if(student.grade >= theGrade)
                printf("%d,%s,%s,%c,%d,%.1f\n",student.code,student.name,student.surname,student.sex,student.year,student.grade);
        }
        printf("\n");

        //???????? ??? ???????.
        fclose(infile);
}
```