**Dokuz Eylül University**

**Electrical and Electronics Engineering Department**

**EED 1010 ALGORITHMS AND PROGRAMMING COURSE**

**PROJECT FOR FINAL EXAMINATION**

Write a C program which will open the random access (binary) file “YOURNAME\_YOURSURNAME.bin” (e.g. FAIK\_YAREN.bin) and process the records from the file. The records will be placed into 6 different binary search trees depending on their “Section” member values. There are different .bin files for each student, therefore the results of your program will be different than all the other students’ results (THERE IS NO NEED FOR CHEATING!).

The structure used for records in “YOURNAME\_YOURSURNAME.bin” file is as follows:

struct student{ //Structure for the records in the file

char Name[30];

char Surname[30];

char address[50];

unsigned long long No;

unsigned int st\_class;

unsigned int Section;

int grade;

} ;

typedef struct student Students;

The prototypes of the functions that will be used in your program are as follows (you will write these functions):

void read\_student\_database(char \*, TreeNodePtr []);

void insertNode( TreeNodePtr \*, Students );

int count\_leaf\_nodes( TreeNodePtr );

double sum\_of\_grades\_in\_leaf\_nodes( TreeNodePtr );

int find\_min\_of\_grades\_in\_tree( TreeNodePtr );

int find\_max\_of\_grades\_in\_tree( TreeNodePtr );

void display\_and\_save\_statistics(char \*,TreeNodePtr []);

void inOrder(TreeNodePtr , FILE \*);

void save\_ordered\_sections(char \*,TreeNodePtr []);

and the type TreeNodePtr is given below:

struct treeNode {

struct treeNode \*leftPtr; // pointer to left subtree

struct student data; //data is struct student type

struct treeNode \*rightPtr; // pointer to right subtree

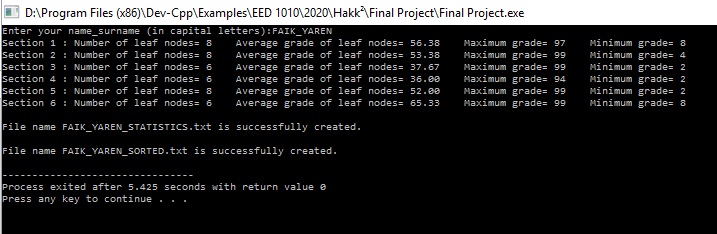
}; // end struct TreeNode

typedef struct treeNode TreeNode;

typedef TreeNode \*TreeNodePtr;

Write the functions, for which the prototypes are given above (DO NOT CHANGE THE PROTOTYPE! OTHERWISE YOUR SOLUTION WILL NOT BE GRADED!).

1. **void read\_student\_database(char \*, TreeNodePtr []) function**: main() function will send two arguments to this function while calling it. The first argument is a string, that is the name of the random access (.bin) file to be opened. The second argument is an array whose elements are of type TreeNodePtr. **JOB OF FUNCTION**: This function will open the random access (.bin) file given to you. It will store each record from the file into one of the trees, whose root nodes are pointed by elements of that array, depending on the “Section” member value of that record. The array will have 6 elements that are pointers to root nodes of different trees. For each section (1 – 6) a different tree will be used. So first array element will point to the root node of the tree into which students registered to section 1 will be inserted. The nodes will be inserted to the trees using their student number members (unsigned long long No). This function will call insertNode() function to insert one student’s data into the corresponding tree. It will close the file.
2. **void insertNode( TreeNodePtr \*, Students ) function**: read\_student\_database() function will send two arguments to this function while calling it. The first argument is the address of a pointer to root node of a binary search tree. The second argument is of type Students which holds the data of a student to be inserted into the binary search tree. **JOB OF FUNCTION:** This function will insert a new node into binary search tree, whose root node is pointed by a TreeNodePtr variable, whose address is passed as first argument to the function. Note that “No” (student number) member of Students type variable will be used in insertion process. The tree will be formed by using “No” (student number) members of Students type variable, but of course tree nodes will contain all members as data (notice TreeNode structure definition!). Duplicate insertion (inserting two nodes with the same value as “No” members) will not be allowed into the binary search tree! (**NOTE: THIS FUNCTION WILL BE RECURSIVE!**)
3. **int count\_leaf\_nodes( TreeNodePtr )** **function**: display\_and\_save\_statistics() function will send an argument of type TreeNodePtr to this function while calling it. This argument is a pointer to the root node of a binary search tree. **JOB OF FUNCTION:** This function will count the leaf nodes in the binary search tree and return this number as result. (**NOTE: THIS FUNCTION WILL BE RECURSIVE!**)
4. **double sum\_of\_grades\_in\_leaf\_nodes( TreeNodePtr** **) function**: display\_and\_save\_statistics() function will send an argument of type TreeNodePtr to this function while calling it. This argument is a pointer to the root node of a binary search tree. **JOB OF FUNCTION:** This function will find the sum of “grade” member values of the leaf nodes in the binary search tree and return this number as (double type) result. (**NOTE: THIS FUNCTION WILL BE RECURSIVE!**)
5. **int find\_min\_of\_grades\_in\_tree( TreeNodePtr ) function:** display\_and\_save\_statistics() function will send an argument of type TreeNodePtr to this function while calling it. This argument is a pointer to the root node of a binary search tree. **JOB OF FUNCTION:** It will search the nodes of the binary search tree to find the node that has minimum value in its “grade” member among all nodes in the tree. It will return this value (of the “grade” member) as result. (**NOTE: THIS FUNCTION WILL BE RECURSIVE!**)
6. **int find\_max\_of\_grades\_in\_tree( TreeNodePtr )** **function:** display\_and\_save\_statistics() function will send an argument of type TreeNodePtr to this function while calling it. This argument is a pointer to the root node of a binary search tree. **JOB OF FUNCTION:** It will search the nodes of the binary search tree to find the node that has maximum value in its “grade” member among all nodes in the tree. It will return this value (of the “grade” member) as result. (**NOTE: THIS FUNCTION WILL BE RECURSIVE!**)
7. **void display\_and\_save\_statistics(char \*,TreeNodePtr []) function**: : main() function will send two arguments to this function while calling it. The first argument is a string representing a file name. The second argument is an array, whose elements are pointers to the root nodes of different binary search trees (each tree contains nodes for students registered to the same section). **JOB OF FUNCTION:** It will display the statistics such as number of leaf nodes in each section’s tree, average of grades in the leaf nodes for each section’s tree, maximum and minimum of grades of students (nodes) in each section’s tree on a separate line on the user screen. Averages will be displayed by using 2 digits precision after decimal point. A sample user screen will be given. This function will call other functions to compute the statistics. This function will create a sequential access file (text file) whose name is sent by main() function. It will write the statistics to the text file in a similar format that is used to display this information on user screen. A sample data file created by calling this function will be given to you (FAIK\_YAREN\_STATISTICS.txt). **(OF COURSE YOUR PROGRAM WILL PRODUCE A DIFFERENT DATA FILE!)**
8. **void inOrder(TreeNodePtr , FILE \*) function:** save\_ordered\_sections() function will send two arguments to this function while calling it. The first argument is a pointer to the root node of a binary search tree. Its type is TreeNodePtr. The second argument is a file pointer to a text file that is opened in save\_ordered\_sections() function. **JOB OF FUNCTION:** This function will perform inorder traversal of the binary search tree and write all the information stored in “data” members of the nodes visited to the file pointed by its second argument. (**NOTE: THIS FUNCTION WILL BE RECURSIVE!**)
9. **void save\_ordered\_sections(char \*,TreeNodePtr []) function**: main() function will send two arguments to this function while calling it. The first argument is a string representing a file name. The second argument is an array, whose elements are pointers to the root nodes of different binary search trees (each tree contains nodes for students registered to the same section). **JOB OF FUNCTION:** This function will create a sequential access file (text file) whose name is sent by main() function. It will write the inorder traversals of the trees of different sections to this file by calling inOrder() function. Inorder traversal of first section’s tree will be written to the file first, then inorder traversal of the second section’s tree will be written and so on. A sample data file created by calling this function will be given to you (FAIK\_YAREN\_SORTED.txt). **(OF COURSE YOUR PROGRAM WILL PRODUCE A DIFFERENT DATA FILE!)**
10. **main() function:** An array of TreeNodePtr type will be declared with 6 elements (since the students in the data file are registered to six different sections). The program will ask the user to enter his / her name in “NAME\_SURNAME” format. (Here you will enter YOURNAME\_YOURSURNAME when you run your program. So your program will open file “YOURNAME\_YOURSURNAME.bin” by calling read\_student\_database() function.) Then it will create file name as “NAME\_SURNAME.bin”. It will call read\_student\_database() function. Then it will create the string “NAME\_SURNAME\_STATISTICS.txt” and send this as the first argument to display\_and\_save\_statistics() function. It will then create the string “NAME\_SURNAME\_SORTED.txt” and send this as the first argument to save\_ordered\_sections() function. The program will end after returning from save\_ordered\_sections() function. A sample user screen is given below:

****

**GRADING CRITERIA FOR YOUR FINAL PROJECT**

1. Your code: 50 points
2. Indentation and comments in your code: 5 points.
3. “YOURNAME\_YOURSURNAME\_STATISTICS.txt” file: 20 points.
4. “YOURNAME\_YOURSURNAME\_SORTED.txt” file: 25 points.

You will download and use the file named “**YOURNAME\_YOURSURNAME.bin**” as datafile for your program. **This is the same file that was used in your midterm project**.

You will upload your code (Yourname\_Yoursurname\_Yourstudentnumber\_Final.cpp file), “YOURNAME\_YOURSURNAME\_STATISTICS.txt” file created by your program and “YOURNAME\_YOURSURNAME\_SORTED.txt” file created by your program.

NOTE: You are given sample files FAIK\_YAREN\_STATISTICS.txt and FAIK\_YAREN\_SORTED.txt . Your program will produce files using same formats as in these files. **So inspect these files and use the same format for your own files**. (Remember that 45 points of your grade will be evaluated from these files!) For example, in FAIK\_YAREN\_SORTED.txt file the fifth column represents st\_class member values, the sixth column represents Section member values of records.

**IMPORTANT NOTES**

1. **DO NOT CHANGE THE STRUCTURES AND FUNCTION PROTOTYPES! OTHERWISE YOUR SOLUTION WILL NOT BE GRADED!**
2. **FOR RECURSIVE FUNCTIONS, IF YOU WRITE A FUNCTION, WHICH IS NOT RECURSIVE OR IF YOU USE LOOPS IN YOUR FUNCTION, IT WILL NOT BE GRADED!**
3. **YOU ARE EXPECTED TO DO YOUR PROJECT YOURSELF! ALTHOUGH IT IS FORBIDDEN, IF YOU STUDY TOGETHER WITH ONE OR MORE (AT MOST THREE) STUDENTS (OTHER THAN YOU) ON THE PROJECT THEN YOU HAVE TO GIVE REFERENCE TO (GIVE THE NAMES OF) ALL THESE STUDENTS. ALL THE STUDENTS STUDYING TOGETHER (IN OTHER WORDS CHEATING) WILL BE GRADED OVER 75, NOT 100. IF YOU CLAIM THAT YOU HAVE STUDIED ON YOUR PROJECT ALONE, AND AFTER INSPECTING YOUR PROJECTS WE FIND THAT YOU HAVE CHEATED, THEN YOU WILL GET ZERO AS FINAL PROJECT GRADE!**
4. **YOU WILL ADD COMMENT LINES IN THE BEGINNING OF YOUR PROGRAM SUCH AS:**

**//I CLAIM THAT I HAVE WRITTEN ALL PARTS OF THE FINAL PROJECT //CODE MYSELF. THE TEXT FILES ARE ALSO PRODUCED BY ME. STUDENT //NUMBER NAME SURNAME.**

**OR IN CASE OF GETTING HELP: (A GROUP WITH 5 OR MORE STUDENTS WILL ALL GET ZERO AS FINAL PROJECT GRADE!)**

**//I CLAIM THAT WE HAVE WRITTEN THE FINAL PROJECT CODE //(COMPLETELY OR PARTLY) AND THE TEXT FILES TOGETHER WITH //THESE STUDENTS:**

**//YOUR STUDENT NUMBER YOUR NAME YOUR SURNAME**

**//FRIEND-1 STUDENT NUMBER FRIEND-1 NAME FRIEND-1 SURNAME**

**//FRIEND-2 STUDENT NUMBER …**

1. **DO NOT COMPRESS THE FILES (ZIP, RAR, ETC..) BEFORE UPLOADING. JUST USE THE FORMATS (.CPP , .TXT) EXPLAINED ABOVE, DO NOT USE ANY OTHER FORMAT! OTHERWISE YOUR PROJECT WILL NOT BE GRADED!**