DSA TERM PROJECT REPORT

SECTION: CSE(AI&DS)

TEAM MEMBERS:24CSB1A19 - Hardik Prajapati
24CSB1A27 - Kasoju Pavan Tej
24CSB1A54 - Rayi Charan Harsha

Student Records Management System using AVL Tree

Question:

(In the attached PDF)

Project Structure:

DSA Term Project:
- *Header Files:*
  - *Courseinfo.h*
  - *Node.h*
  - *Node_operations.h*
  - *Filehandler.h*
- *CPP Files:*
  - *Courseinfo.cpp*
  - *Node.cpp*
  - *Node_operations.cpp*
  - *Filehandler.cpp*
  - *Main.cpp*
- *TXT Files:*
  - *Input.txt*
- *Others:*
  - *Makefile*

Project Summary:

- [ ] *We made a class named “courses” which creates a data type with both courseId and marks.*
- [ ] *It also contains some getters and setters for the courseId and marks.*
- [ ] *Then we used this course type to make a new class for the node of the AVL tree.*
- [ ] *This “node” class contains all the data of a student, including rollNo, name, CGPA, number of courses, info of each of the courses, and height of each node.*
- [ ] *There are setters and getters for each of these data, and also functions to add student, update, display and delete courses, which are used to update the node information.*
- [ ] *Then we made the node_operations, which constructs and modifies the AVL tree, with functions such as rightRotate, leftRotate that help us to balance the AVl tree.*
- [ ] *There is a function to check if the tree is balanced or not using the heights of each node. This is called after each operation and if the tree is still balanced then no need to modify the tree, but if it is unbalanced then we call the updateTree function to balance it.*
- [ ] *After balancing, the updateHeight is called to change the height of each node.*
- [ ] *The other functions in this class help us to modify, add or remove a node data and also to display the updated tree data.*
- [ ] *Now to read and handle the input file, we made a new class called FileHandler.*
- [ ] *In this, we made use of the fstream and sstream STLs to read and process the file.*
- [ ] *The constructor of this class opens the file and handles the errors. Then the insertion of students is handled with several error handling statements for incorrect format, invalid values for student data, etc.*

- ☐ *Once the input format is judged to be correct, this function uses integer string stream to read the line and using comma(,) as a delimiter, segregates the data into respective fields such as rollNo, etc and finally calls the addStudent function.*
- ☐ *Similarly, other functions are used to handle the input of addCourse, deleteCourse, deleteStudent, modifyCGPA, modifyCourseMarks, handle rollNo and CGPA lesser, greater or in between.*
- ☐ *All these functions are called by the processCommand function after it reads the # command from the input file. This is done by the switch case operator.*
- ☐ *This function is in turn called by the processFile function after it reads the command line and does error handling and command recognition.*
- ☐ *Finally, in the main function, the filehanler is called after the error handling.*
- ☐ *All of this can be run by the use of makefile in Linux operating system.*

Highlights Of The Project:

1. *All the functions do error handling very well, which leads to correct output and no unexpected errors.*
2. *The student data can be accessed using both CGPA and rollNo as keys using different commands, which makes it easy for the user to access and perform changes on the student database.*
3. *Since the project uses AVL tree, the operations take O(log n) time, which significantly improves the efficiency for large databases.*
4. *By changing the input file, the user can easily use this same program for another student database, making it versatile.*

(For the codes and outputs, please refer to the attached document.)