**Meru UNIVERSITY School Management System**

**Main source code**

Language: C++

// sms\_system.cpp - MeruLite School Management System

// Modules: StudentRegistry, CourseScheduler, FeeTracker, LibrarySystem, PerformanceAnalytics

// Author: autogenerated for assignment (unique implementation)

#include <bits/stdc++.h> using namespace std;

// StudentRegistry: Hash Table for quick lookup struct Student {

int id; string name; string program; int year; Student(){} Student(int i,string n,string p,int y):id(i),name(n),program(p),year(y){} };

class StudentRegistry { unordered\_map<int, Student> store; public:

bool addStudent(const Student &s){ if(store.count(s.id)) return false; store[s.id]=s; return true Student\* findStudent(int id){ return store.count(id)?&store[id]:nullptr; } bool removeStudent(int id){ return store.erase(id)>0; } };

// CourseScheduler: Circular Queue implementation class CourseScheduler { unordered\_map<string, vector<int>> enrolled; unordered\_map<string, queue<int>> waiting; public:

void registerStudent(string course, int id, int capacity){ if((int)enrolled[course].size()<capacity) enrolled[course].push\_back(id); else waiting[course].push(id); }

};

// FeeTracker: AVL Tree for sorted transactions struct FeeNode {

int txid, amount; string ref; FeeNode \*l,\*r; int h;

FeeNode(int i,int a,string r\_):txid(i),amount(a),ref(r\_),l(nullptr),r(nullptr),h(1){} };

class FeeTracker { FeeNode\* root=nullptr; int height(FeeNode\* n){return n?n->h:0;}

int balance(FeeNode\* n){return n?height(n->l)-height(n->r):0;} FeeNode\* rotateL(FeeNode\* x){ FeeNode\*y=x->r; x->r=y->l; y->l=x; x->h=max(height(x->l),height(x-> FeeNode\* rotateR(FeeNode\* y){ FeeNode\*x=y->l; y->l=x->r; x->r=y; y->h=max(height(y->l),height(y-> FeeNode\* insert(FeeNode\* n,int id,int amt,string r\_){ if(!n) return new FeeNode(id,amt,r\_); if(id<n->txid) n->l=insert(n->l,id,amt,r\_); else if(id>n->txid) n->r=insert(n->r,id,amt,r\_); else return n; n->h=max(height(n->l),height(n->r))+1; int bf=balance(n); if(bf>1&&id<n->l->txid) return rotateR(n); if(bf<-1&&id>n->r->txid) return rotateL(n);

if(bf>1&&id>n->l->txid){n->l=rotateL(n->l);return rotateR(n);} if(bf<-1&&id<n->r->txid){n->r=rotateR(n->r);return rotateL(n);} return n; } public:

void recordPayment(int id,int amt,string ref){root=insert(root,id,amt,ref);} };

// LibrarySystem: Stack + HashMap class LibrarySystem {

unordered\_map<string,bool> books; stack<string> recent; public:

void addBook(string isbn){books[isbn]=true;}

bool borrow(string isbn){if(!books[isbn])return false;books[isbn]=false;recent.push(isbn);return bool giveBack(string isbn){books[isbn]=true;return true;} };

// PerformanceAnalytics: Matrix + Heap class PerformanceAnalytics { unordered\_map<int,vector<int>> marks; public:

void addMark(int id,int subj,int score){marks[id].resize(3,-1);marks[id][subj]=score;} vector<pair<int,double>> top(int k){ priority\_queue<pair<double,int>> pq; for(auto&p:marks){double sum=0;int c=0;for(int m:p.second)if(m>=0){sum+=m;c++;}if(c)pq.push({ vector<pair<int,double>>res;while(k--&&!pq.empty()){auto x=pq.top();pq.pop();res.push\_back({x return res; } };

int main(){ StudentRegistry s; s.addStudent({1,"Alice","Data Sci",1});

CourseScheduler c; c.registerStudent("CS101",1,2);

FeeTracker f; f.recordPayment(100,20000,"1");

LibrarySystem l; l.addBook("978-1-234"); l.borrow("978-1-234"); PerformanceAnalytics pa; pa.addMark(1,0,80); pa.addMark(1,1,90); auto top=pa.top(1); cout<<"Top student: "<<top[0].first<<" avg="<<top[0].second<<"\n"; return;