OOP Assignment 1

# Question 1

## Code

#include<iostream>

using namespace std;

class Mentor;

class Sport;

class Skill;

class Student

{

private:

string studentID;

string name;

int age;

string sportsInterests[5];

string mentorAssigned;

public:

Student(string s, string n, int a, string i[], int size, string m)

{

studentID = s;

name = n;

age = a;

for (int j = 0; j < size; j++)

{

sportsInterests[j] = i[j];

}

mentorAssigned = m;

}

void registerForMentorship(Mentor &m);

void viewMentorDetails()

{

cout << "\nThe name of the mentor who is assigned to you is: " << mentorAssigned << endl;

}

void updateSportsInterest()

{

cout << "\nEnter the number of sports in which you are interested: ";

int n;

cin >> n;

cout << endl << "Enter the sports interests: " << endl;

for (int j = 0; j < n; j++)

{

cin >> sportsInterests[j];

cout << endl;

}

}

};

class Mentor

{

private:

int mentorID;

string name;

string sportsExpertise[5];

int maxLearners;

string assignedLearners[3];

public:

Mentor(int m, string n, string s[], int size)

{

mentorID = m;

name = n;

for (int j = 0; j < size; j++)

{

sportsExpertise[j] = s[j];

}

maxLearners = 3;

}

void assignLearner(Student &s);

void removeLearner(string learner)

{

for (int j = 0; j < maxLearners; j++)

{

if (assignedLearners[j] == learner)

{

assignedLearners[j] = "";

cout << "\nLearner successfully removed" << endl;

return;

}

}

}

void viewLearners()

{

cout << "\nThe assigned learners are: " << endl;

for (int j = 0; j < maxLearners; j++)

{

cout << assignedLearners[j] << endl;

}

}

void provideGuidance()

{

cout << "\nA journey of a thousand steps begins with a single step" << endl;

}

};

class Sport

{

private:

int sportID;

string name;

string description;

string requiredSkills[5];

int max;

public:

Sport(int s, string n, string d, string r[], int size)

{

sportID = s;

name = n;

description = d;

for (int j = 0; j < size; j++)

{

requiredSkills[j] = r[j];

}

max = size;

}

void addSkill()

{

cout << "\nEnter the maximum number of skills: ";

cin >> max;

cout << "\nEnter the names of the skills required: " << endl;

for (int j = 0; j < max; j++)

{

cin >> requiredSkills[j];

}

}

void removeSkill(string skill)

{

for (int j = 0; j < max; j++)

{

if (requiredSkills[j] == skill)

{

requiredSkills[j] = "";

cout << "\nSkill successfully removed" << endl;

return;

}

}

}

};

class Skill

{

private:

int skillID;

string skillName;

string description;

public:

Skill(int i, string n, string d)

{

skillID = i;

skillName = n;

description = d;

}

void showSkillDetails()

{

cout << "\nThe ID of the skill is: " << skillID;

cout << "\nThe name of the skill is: " << skillName;

cout << "\nThe description of the skill is: " << description << endl;

}

void updateSkillDescription(string newDescription)

{

description = newDescription;

cout << "Description successfully updated";

}

};

int main()

{

cout<<"Nafees Uddin 24K-2526"<<endl;

string sports[] = {"Tennis", "Football"};

string skills[] = {"Agility", "Speed"};

Student student1("S123", "Saad", 20, sports, 2, "Ali");

Mentor mentor1(1, "Ali", sports, 2);

Sport sport1(101, "Tennis", "A game of skill and agility", skills, 2);

Skill skill1(201, "Agility", "Improves movement speed and reaction time");

student1.viewMentorDetails();

mentor1.provideGuidance();

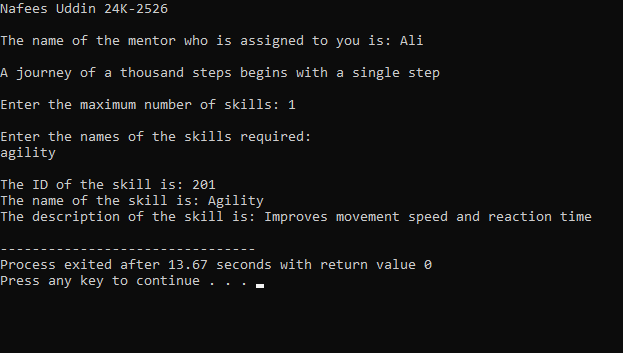
sport1.addSkill();

skill1.showSkillDetails();

return 0;

}

## Output



# Question 2

## Code

#include <iostream>

#include <string>

using namespace std;

class Robot

{

private:

string name;

int hits;

public:

Robot(const string& name)

{

this->name = name;

hits = 0;

}

void hitBall(int& ballX, int& ballY, const string& direction)

{

if (direction == "up")

{

ballY += 1;

}

else if (direction == "down")

{

ballY -= 1;

}

else if (direction == "left")

{

ballX -= 1;

}

else if (direction == "right")

{

ballX += 1;

}

hits++;

}

string getName() const

{

return name;

}

int getHits() const

{

return hits;

}

};

class Ball

{

private:

int x, y;

public:

Ball(int x = 0, int y = 0)

{

this->x = x;

this->y = y;

}

void move(int dx, int dy)

{

x += dx;

y += dy;

}

int getX() const

{

return x;

}

int getY() const

{

return y;

}

};

class Goal

{

private:

int goalX, goalY;

public:

Goal(int x = 3, int y = 3)

{

goalX = x;

goalY = y;

}

bool isGoalReached(int ballX, int ballY) const

{

return (ballX == goalX && ballY == goalY);

}

};

class Team

{

private:

string teamName;

Robot\* robot;

public:

Team(const string& name, Robot\* robot)

{

teamName = name;

this->robot = robot;

}

string getTeamName() const

{

return teamName;

}

Robot\* getRobot() const

{

return robot;

}

};

class Game

{

private:

Team\* teamOne;

Team\* teamTwo;

Ball ball;

Goal goal;

public:

Game(Team\* teamOne, Team\* teamTwo)

{

this->teamOne = teamOne;

this->teamTwo = teamTwo;

Ball(0,0);

}

void startGame()

{

play(teamOne);

play(teamTwo);

declareWinner();

}

void play(Team\* team)

{

Robot\* robot = team->getRobot();

int ballX = ball.getX(), ballY = ball.getY();

string direction;

cout << team->getTeamName() << "'s turn:" << endl;

while (!goal.isGoalReached(ballX, ballY))

{

cout << "Ball Position: (" << ballX << ", " << ballY << ")" << endl;

cout << robot->getName() << ", choose direction (up, down, left, right): ";

cin >> direction;

robot->hitBall(ballX, ballY, direction);

}

cout << robot->getName() << " reached the goal with " << robot->getHits() << " hits!" << endl;

}

void declareWinner()

{

int teamOneHits = teamOne->getRobot()->getHits();

int teamTwoHits = teamTwo->getRobot()->getHits();

if (teamOneHits < teamTwoHits)

{

cout << teamOne->getTeamName() << " wins!" << endl;

}

else if (teamTwoHits < teamOneHits)

{

cout << teamTwo->getTeamName() << " wins!" << endl;

}

else

{

cout << "It's a draw!" << endl;

}

}

};

int main()

{

cout<<"Nafees Uddin 24K-2526"<<endl;

Robot robot1("Robot1");

Robot robot2("Robot2");

Team teamOne("Team One", &robot1);

Team teamTwo("Team Two", &robot2);

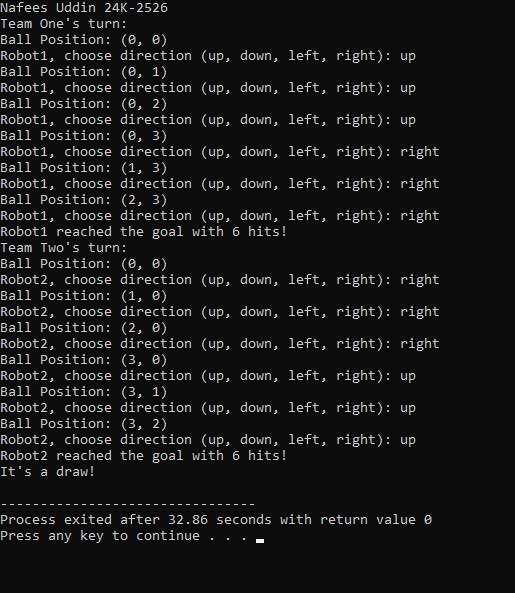
Game game(&teamOne, &teamTwo);

game.startGame();

return 0;

}

## Output



# Question 3

## Code

#include <iostream>

#include <string>

using namespace std;

class User

{

private:

int UserId;

string name;

int age;

string licenseType;

string contact;

public:

User(int id, string n, int a, string l, string c)

{

UserId = id;

name = n;

licenseType = l;

contact = c;

}

int getUserID()

{

return UserId;

}

string getName()

{

return name;

}

int getAge()

{

return age;

}

string getLicenseType()

{

return licenseType;

}

string getContact()

{

return contact;

}

void setName(string n)

{

name = n;

}

void setAge(int a)

{

age = a;

}

void setLicenseType(string l)

{

licenseType = l;

}

void setContact(string c)

{

contact = c;

}

};

class Vehicle

{

private:

string model;

double rentalPrice;

string requiredLicense;

public:

Vehicle(string m, double r, string license)

{

model = m;

rentalPrice = r;

requiredLicense = license;

}

string getModel()

{

return model;

}

double getRentalPrice()

{

return rentalPrice;

}

string getRequiredLicense()

{

return requiredLicense;

}

};

class RentalSystem

{

private:

Vehicle \*vehicle[3];

int vehicleCount;

public:

RentalSystem()

{

vehicleCount = 0;

}

void addVehicle(Vehicle \*v)

{

if (vehicleCount < 3)

{

vehicle[vehicleCount++] = v;

}

}

void showAvailableVehicles()

{

for (int i = 0; i < vehicleCount; i++)

{

cout << "Model: " << vehicle[i]->getModel() << ", Price: " << vehicle[i]->getRentalPrice() << endl;

}

}

void rentVehicle(User &user, string model)

{

for (int i = 0; i < vehicleCount; i++)

{

if (vehicle[i]->getModel() == model)

{

if (user.getLicenseType() == vehicle[i]->getRequiredLicense())

{

cout << user.getName() << " rented " << model << " successfully!" << endl;

}

else

{

cout << "You do not have the required license to rent this vehicle." << endl;

}

return;

}

}

cout << "Vehicle not found!" << endl;

}

};

int main()

{

cout<<"Nafees Uddin 24K-2526"<<endl;

User user1(249, "Nafees", 18, "Full", "854-693");

Vehicle v1("Tesla", 5000, "Full");

Vehicle v2("Porsche", 3000, "Learner");

RentalSystem rentalSystem;

rentalSystem.addVehicle(&v1);

rentalSystem.addVehicle(&v2);

cout << "Available vehicles:" << endl;

rentalSystem.showAvailableVehicles();

rentalSystem.rentVehicle(user1, "Tesla");

rentalSystem.rentVehicle(user1, "Porsche");

return 0;

}

## Output

A screenshot of a computer screen

AI-generated content may be incorrect.

# Question 4

## UML

A diagram of a student

AI-generated content may be incorrect.

## Code

#include <iostream>

#include <string>

using namespace std;

class Student

{

private:

int studentID;

string name;

int semester;

bool transportCardActive;

public:

Student()

{

studentID = 00000;

name = "ABC";

semester = 0;

transportCardActive = false;

}

void registerStudent(int id, string n, int sem)

{

studentID = id;

name = n;

semester = sem;

transportCardActive = false;

cout << "Student registered: " << name << "ID: " << studentID <<endl;

}

void payFees()

{

transportCardActive = true;

cout << "Fees paid! Transport card activated for " << name << "ID: " << studentID <<endl;

}

int getID()

{

return studentID;

}

string getName()

{

return name;

}

bool isCardActive()

{

return transportCardActive;

}

};

class BusRoute

{

private:

int routeID;

string stops[10];

int stopCount;

string busNumber;

public:

BusRoute()

{

routeID = 0000;

busNumber = "ABC";

stopCount = 0;

}

void createRoute(int id, string bus)

{

routeID = id;

busNumber = bus;

stopCount = 0;

}

void addStop(string stop)

{

if (stopCount < 10)

{

stops[stopCount] = stop;

stopCount++;

}

else

{

cout << "Cannot add more stops."<<endl;

}

}

void displayStops()

{

cout << "Route " << routeID << " Bus: " << busNumber << " Stops: ";

for (int i = 0; i < stopCount; i++)

{

cout << stops[i] << " -> ";

}

cout << "End"<<endl;

}

};

class Attendance

{

private:

int studentID;

string date;

bool status;

public:

void recordAttendance(int id, string d)

{

studentID = id;

date = d;

status = true;

cout << "Attendance recorded for Student ID: " << studentID << " on " << date << endl;

}

};

class TransportSystem

{

private:

Student students[10];

int studentCount;

BusRoute routes[5];

int routeCount;

Attendance attendance[50];

int attendanceCount;

public:

TransportSystem()

{

studentCount = 0;

routeCount = 0;

attendanceCount = 0;

}

void registerStudent(int id, string name, int sem)

{

if (studentCount < 10)

{

students[studentCount].registerStudent(id, name, sem);

studentCount++;

}

else

{

cout << "Cannot register more students."<<endl;

}

}

void processPayment(int id)

{

for (int i = 0; i < studentCount; i++)

{

if (students[i].getID() == id)

{

students[i].payFees();

return;

}

}

cout << "Student ID not found!"<<endl;

}

void addBusRoute(int id, string bus)

{

if (routeCount < 5)

{

routes[routeCount].createRoute(id, bus);

routeCount++;

}

else

{

cout << "Cannot add more routes.\n";

}

}

void addStopToRoute(int id, string stop)

{

for (int i = 0; i < routeCount; i++)

{

routes[i].addStop(stop);

return;

}

cout << "Route not found!"<<endl;

}

void recordAttendance(int id, string date)

{

for (int i = 0; i < studentCount; i++)

{

if (students[i].getID() == id && students[i].isCardActive())

{

if (attendanceCount < 50)

{

attendance[attendanceCount].recordAttendance(id, date);

attendanceCount++;

}

else

{

cout << "Attendance record full.\n";

}

return;

}

}

cout << "Attendance failed: Card not active or student not found.\n";

}

};

int main()

{

cout<<"Nafees Uddin 24K-2526"<<endl;

TransportSystem system;

system.registerStudent(1, "Nafees", 3);

system.registerStudent(2, "Anas", 2);

system.processPayment(1);

system.addBusRoute(101, "Bus 1");

system.recordAttendance(1, "2025-02-20");

system.recordAttendance(2, "2025-02-21");

return 0;

}

## Output

A screenshot of a computer program

AI-generated content may be incorrect.