**OOP Assignment 1**

**Muhammad Umer Karim**

**BCS-2k**

**24K-0880**

**Question 1:**

**Code:**

#include<iostream>

using namespace std;

class skill{

public:

int id;

string name;

string description;

skill() : id(0), name(""), description("") {}

skill(int \_id, string \_name, string \_description) : id(id), name(\_name), description(\_description){};

void showSkillDetails(){

cout<<"Skill Details: "<<endl<<"Name: "<<name<<endl<<"ID: "<<id<<endl<<"Description: "<<description<<endl;

}

void updateSkillDescription(string newDescription){

description = newDescription;

}

bool operator==(const skill& other) const {

return id == other.id;

}

};

class sport{

public:

int id;

string name;

string description;

skill \*requiredSkills;

int s\_size;

sport(int \_id, string \_name, string \_description, skill \*\_skills, int size) : id(\_id), name(\_name), description(\_description), s\_size(size){

requiredSkills = new skill[size];

for(int i = 0; i<size; i++){

requiredSkills[i] = \_skills[i];

}

}

void addSkill(skill s){

skill\* temp = new skill[s\_size+1];

for(int i = 0; i<s\_size; i++){

temp[i] = requiredSkills[i];

}

temp[s\_size] = s;

s\_size++;

delete[] requiredSkills;

requiredSkills = temp;

delete[] temp;

}

//for debugging - by Umar

void displaySkills() {

for (int i = 0; i < s\_size; i++) {

cout << "Skill " << i + 1 << ": " << requiredSkills[i].name << endl;

}

}

void removeSkill(skill s) {

int index = -1;

for (int i = 0; i < s\_size; i++) {

if (requiredSkills[i] == s) {

index = i;

break;

}

}

if (index == -1) {

cout << "No such skill found" << endl;

return;

}

skill\* temp = new skill[s\_size - 1];

for (int i = 0, j = 0; i < s\_size; i++) {

if (i != index) {

temp[j++] = requiredSkills[i];

}

}

delete[] requiredSkills;

requiredSkills = temp;

s\_size--;

}

~sport() {

delete[] requiredSkills;

}

};

class mentor;

class student {

public:

int id;

string name;

int age;

string\* Interest;

int interestCount;

mentor\* Mentor;

student(int \_id, string \_name, int \_age, int interestSize, string \*\_interest) : id(\_id), name(\_name), age(\_age) {

Interest = new string[interestSize];

interestCount = interestSize;

for (int i = 0; i < interestCount; i++) {

Interest[i] = \_interest[i];

}

Mentor = NULL;

}

~student() {

delete[] Interest;

}

void registerForMentorship(mentor\* m);

void viewMentorDetails();

};

class mentor {

public:

int id;

string name;

string expertise;

int max;

student\*\* Assigned\_Students;

int studentCount;

mentor(int \_id, string \_name, string \_expertise, int \_max) : id(\_id), name(\_name), expertise(\_expertise), max(\_max) {

Assigned\_Students = new student\*[max];

studentCount = 0;

for (int i = 0; i < max; i++) {

Assigned\_Students[i] = NULL;

}

}

~mentor() {

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

delete Assigned\_Students[i];

}

delete[] Assigned\_Students;

}

void assignLearner(student\* s) {

if (studentCount < max) {

for (int i = 0; i < max; i++) {

if (Assigned\_Students[i] == NULL) {

Assigned\_Students[i] = s;

s->Mentor = this;

studentCount++;

cout << s->name << " assigned to mentor " << name << endl;

return;

}

}

}

cout << "Mentor's limit reached.\n";

}

void removeLearner(student\* s) {

for (int i = 0; i < max; i++) {

if (Assigned\_Students[i] == s) {

Assigned\_Students[i] = NULL;

s->Mentor = NULL;

studentCount--;

cout << s->name << " removed from mentor " << name << endl;

return;

}

}

}

void viewLearners() {

cout << "Mentor " << name << "'s assigned students:\n";

for (int i = 0; i < max; i++) {

if (Assigned\_Students[i] != NULL) {

cout << "Student: " << Assigned\_Students[i]->name << endl;

}

}

}

};

void student::registerForMentorship(mentor\* m) {

m->assignLearner(this);

\*Mentor = \*m;

}

void student::viewMentorDetails(){

if (Mentor) {

cout << "Mentor Details for " << name << ":\n";

cout << "Name: " << Mentor->name << "\nID: " << Mentor->id << "\nExpertise: " << Mentor->expertise << endl;

} else {

cout << name << " has no assigned mentor.\n";

}

}

#include<iostream>

using namespace std;

int main() {

int choice;

skill skills[3] = {

skill(1, "Dribbling", "Control the ball"),

skill(2, "Shooting", "Hit the target"),

skill(3, "Passing", "Deliver the ball to teammates")

};

sport football(1, "Football", "Team-based sport", skills, 3);

mentor coach(1, "Umar", "Football Coach", 2);

student \*students[2];

students[0] = new student(1, "Akbar", 20, 2, new string[2]{"Football", "Basketball"});

students[1] = new student(2, "Saad", 21, 1, new string[1]{"Football"});

do {

cout << "\n===== Sports Management System =====\n";

cout << "1. View Sport Details\n";

cout << "2. Add Skill to Sport\n";

cout << "3. Remove Skill from Sport\n";

cout << "4. Register Student for Mentorship\n";

cout << "5. View Mentor's Students\n";

cout << "6. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Sport: " << football.name << "\nDescription: " << football.description << "\n";

football.displaySkills();

break;

case 2: {

int id;

string name, desc;

cout << "Enter skill ID: "; cin >> id;

cout << "Enter skill name: "; cin >> name;

cout << "Enter skill description: "; cin >> desc;

football.addSkill(skill(id, name, desc));

cout << "Skill added successfully!\n";

break;

}

case 3: {

int id;

cout << "Enter skill ID to remove: "; cin >> id;

for (int i = 0; i < football.s\_size; i++) {

if (football.requiredSkills[i].id == id) {

football.removeSkill(football.requiredSkills[i]);

cout << "Skill removed successfully!\n";

break;

}

}

break;

}

case 4:

cout << "Assigning " << students[0]->name << " to mentor...\n";

students[0]->registerForMentorship(&coach);

cout << "Student registered successfully!\n";

break;

case 5:

coach.viewLearners();

break;

case 6:

cout << "Exiting program...\n";

break;

default:

cout << "Invalid choice! Try again.\n";

}

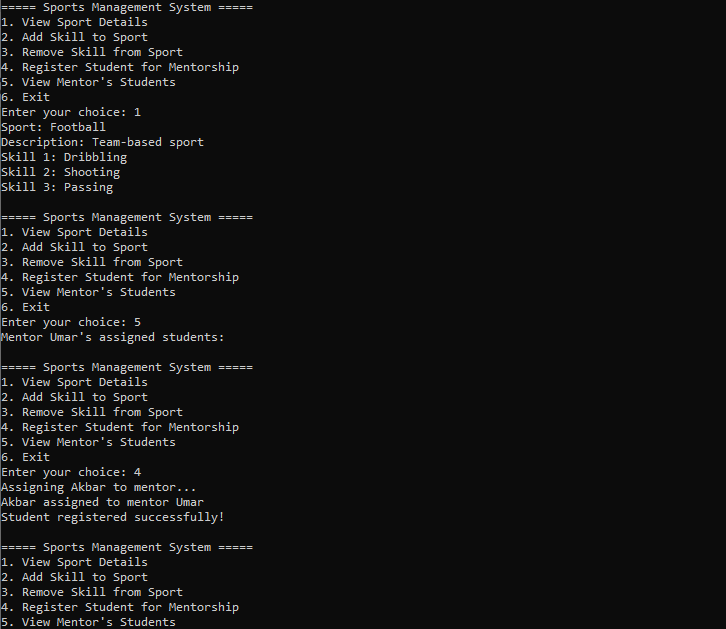
} while (choice != 6);

for (int i = 0; i < 2; i++) delete students[i];

return 0;

}

**Output:**



**Question 2:**

**Code:**

#include <iostream>

#include <tuple>

#include <cstdlib>

#include <ctime>

using namespace std;

class Robot{

public:

string name;

int hits;

Robot(string n): name(n), hits(0) {}

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

if (direction == "up") ballY--;

else if (direction == "down") ballY++;

else if (direction == "left") ballX--;

else if (direction == "right") ballX++;

hits++;

cout << name << " hit the ball " << direction << " to (" << ballX << ", " << ballY << ")\n";

}

int getHits(){

return hits;

}

};

class Ball{

private:

int x;

int y;

public:

Ball(int startX = 0, int startY = 0): x(startX), y(startY) {}

int getX(){ return x; }

int getY(){ return y; }

void move(int dx, int dy){

x += dx;

y += dy;

}

tuple<int, int> getPosition(){

return make\_tuple(x, y);

}

};

class Goal{

private:

int x;

int y;

public:

Goal(int \_x = 3, int \_y = 3): x(\_x), y(\_y) {}

bool isGoalReached(int ballX, int ballY){

return (ballX == x && ballY == y);

}

};

class Team{

public:

string teamName;

Robot \*player;

Team(string name, Robot \*robotPlayer): teamName(name), player(robotPlayer) {}

};

class Game{

public:

Team team1;

Team team2;

Ball ball;

Goal goal;

Game(Team t1, Team t2): team1(t1), team2(t2), ball(0, 0), goal() {}

void startGame(){

cout << "Starting the game!\n";

play(&team1);

play(&team2);

declareWinner();

}

void play(Team \*team){

cout << "\nTeam " << team->teamName << " is playing!\n";

while (!goal.isGoalReached(ball.getX(), ball.getY())){

string directions[] = {"up", "down", "left", "right"};

int randomIndex = rand() % 4;

int ballX = ball.getX();

int ballY = ball.getY();

team->player->hitBall(ballX, ballY, directions[randomIndex]);

ball.move(ballX - ball.getX(), ballY - ball.getY());

}

cout << team->teamName << " reached the goal in " << team->player->getHits() << " hits!\n";

}

void declareWinner(){

cout << "\nGame Over!\n";

int hitsTeam1 = team1.player->getHits();

int hitsTeam2 = team2.player->getHits();

if (hitsTeam1 < hitsTeam2)

cout << "Winner: " << team1.teamName << endl;

else if (hitsTeam2 < hitsTeam1)

cout << "Winner: " << team2.teamName << endl;

else

cout << "It's a draw!\n";

}

};

int main(){

srand(time(0));

Robot r1("Robo-One"), r2("Robo-Two");

Team t1("Red Warriors", &r1), t2("Blue Strikers", &r2);

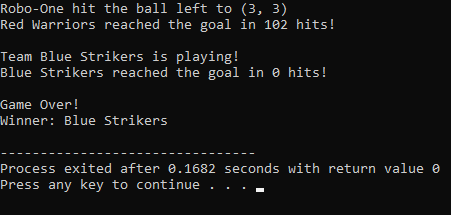
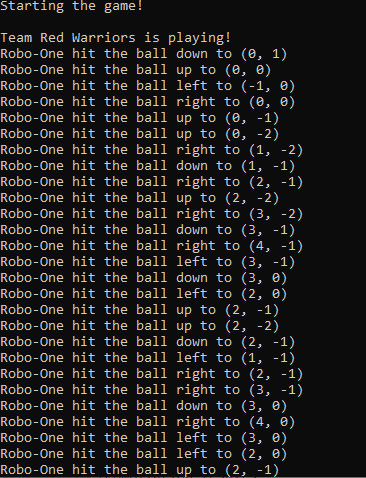
Game footballGame(t1, t2);

footballGame.startGame();

return 0;

}

**Output:**



**Question 3:**

**Code:**

#include <iostream>

using namespace std;

class vehicle{

public:

string model;

int rent;

string type;

vehicle(string m = "", int r = 0, string t = "") : model(m), rent(r), type(t) {}

void showVehicleDetails(){

cout << "Vehicle Model: " << model << "\nRent: " << rent << "\nRequired License: " << type << endl;

}

};

class user{

private:

int id;

int age;

int number;

string License;

vehicle\* Vehicle;

public:

user(int \_id, int \_age, int num, string l) : id(\_id), age(\_age), number(num), License(l), Vehicle(nullptr) {}

~user(){

delete Vehicle;

}

void updateDetails(int \_id, int \_age, int num, string l){

id = \_id;

age = \_age;

number = num;

License = l;

}

void registerVehicle(vehicle\* v){

if (License != v->type) {

cout << "Not eligible for this vehicle! You need a " << v->type << " license.\n";

return;

}

if (Vehicle) delete Vehicle;

Vehicle = new vehicle(\*v);

cout << "Vehicle rented successfully: " << Vehicle->model << endl;

}

void showDetails(){

cout << "\nUser Details:\n";

cout << "ID: " << id << "\nAge: " << age << "\nNumber: " << number << "\nLicense Type: " << License << endl;

if (Vehicle) {

cout << "\n Rented Vehicle Details:\n";

Vehicle->showVehicleDetails();

} else {

cout << "No Vehicle Registered\n";

}

}

};

int main(){

int id, age, number;

string l;

cout << "Enter Your Details:\n";

cout << "ID: ";

cin >> id;

cout << "Age: ";

cin >> age;

cout << "Number: ";

cin >> number;

do{

cout << "License (beginner | intermediate | full) {Case sensitive - type exactly}: ";

cin >> l;

if (l != "beginner" && l != "intermediate" && l != "full") {

cout << "Invalid input! Please type exactly as given.\n";

}

}while (l != "beginner" && l != "intermediate" && l != "full");

user user1(id, age, number, l);

vehicle\* vehicles[3];

vehicles[0] = new vehicle("Toyota Corolla", 3000, "beginner");

vehicles[1] = new vehicle("Honda Civic", 4000, "intermediate");

vehicles[2] = new vehicle("BMW X5", 7000, "full");

cout << "\nAvailable Vehicles:\n";

for (int i = 0; i < 3; i++){

cout << i + 1 << ". ";

vehicles[i]->showVehicleDetails();

cout << "------------------\n";

}

int choice;

do{

cout << "Enter the number of the vehicle you want to rent (1-3): ";

cin >> choice;

if (choice < 1 || choice > 3) {

cout << "Invalid choice! Please enter a number between 1 and 3.\n";

}

}while (choice < 1 || choice > 3);

user1.registerVehicle(vehicles[choice - 1]);

user1.showDetails();

for (int i = 0; i < 3; i++){

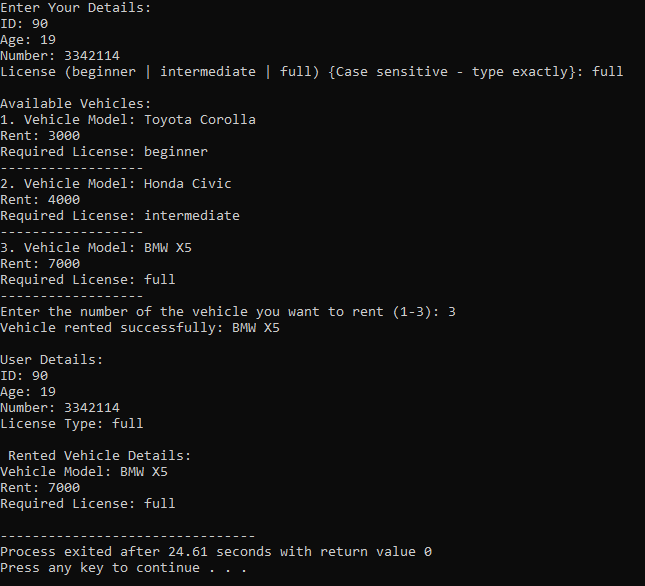
delete vehicles[i];

}

return 0;

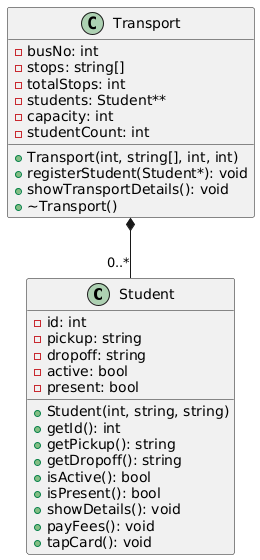
}

**Output:**



**Question 4:**

**UML Diagram:**

****

**Code:**

#include <iostream>

using namespace std;

class Student{

private:

int id;

string pickup;

string dropoff;

bool active;

bool present;

public:

Student(int \_id, string PU, string DO) : id(\_id), pickup(PU), dropoff(DO), active(false), present(false) {}

int getId() const { return id; }

string getPickup() const { return pickup; }

string getDropoff() const { return dropoff; }

bool isActive() const { return active; }

bool isPresent() const { return present; }

void showDetails() const {

cout << "Student ID: " << id << "\nPickup: " << pickup << "\nDropoff: " << dropoff;

cout << "\nStatus: " << (active ? "Active" : "Inactive") << "\nAttendance: " << (present ? "Present" : "Absent") << endl;

}

void payFees(){

if (!active){

active = true;

cout << "Student " << id << " has paid semester fees.\n";

} else {

cout << "Fees already paid for Student " << id << ".\n";

}

}

void tapCard(){

if (active){

present = true;

cout << "Attendance marked for Student " << id << ".\n";

} else{

cout << "Payment required! Student " << id << " cannot mark attendance.\n";

}

}

};

class Transport{

private:

int busNo;

string\* stops;

int totalStops;

Student\*\* students;

int capacity, studentCount = 0;

public:

Transport(int no, string\* stopList, int totalStops, int cap) : busNo(no), totalStops(totalStops), capacity(cap) {

stops = new string[totalStops];

for (int i = 0; i < totalStops; i++){

stops[i] = stopList[i];

}

students = new Student\*[capacity];

for (int i = 0; i < capacity; i++){

students[i] = nullptr;

}

}

void registerStudent(Student\* s){

if (studentCount < capacity){

for (int i = 0; i < totalStops; i++){

if (s->getPickup() == stops[i]){

students[studentCount] = new Student(\*s);

studentCount++;

cout << "Student " << s->getId() << " registered for bus " << busNo << ".\n";

return;

}

}

cout << "Error: Student's pickup location does not match bus stops.\n";

} else {

cout << "Error: Bus " << busNo << " is full.\n";

}

}

void showTransportDetails() const{

cout << "\nBus Number: " << busNo << "\nStops: ";

for (int i = 0; i < totalStops; i++){

cout << stops[i] << (i == totalStops - 1 ? "" : ", ");

}

cout << "\nCapacity: " << capacity << "\nRegistered Students: " << studentCount << endl;

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

if (students[i] != nullptr) {

students[i]->showDetails();

cout << "-----------------\n";

}

}

}

~Transport(){

delete[] stops;

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

delete students[i];

}

delete[] students;

}

};

int main(){

string stopsList[] = {"Stop A", "Stop B", "Stop C", "Stop D"};

Transport bus1(101, stopsList, 4, 3);

Student s1(1, "Stop A", "Stop C");

Student s2(2, "Stop B", "Stop D");

Student s3(3, "Stop C", "Stop A");

Student s4(4, "Stop X", "Stop D");

bus1.registerStudent(&s1);

bus1.registerStudent(&s2);

bus1.registerStudent(&s3);

bus1.registerStudent(&s4);

s1.payFees();

s2.payFees();

s1.tapCard();

s2.tapCard();

s3.tapCard();

bus1.showTransportDetails();

return 0;

}

**Output:**

