PIYUSH KUMAR MISHRA

230957212

ROLL N0:70

WEEK 5:

Q1:Exercise 1: Student Grades Management System

Create a simple student grades management system which perform the following functions

(Use a dictionary where the keys are student names, and the values are lists of grades.):

• Add a student: Add a student's name and their grades for multiple subjects.

• Update a Grade: Update a specific grade for a student in each subject.

• Remove a student: Remove a student from the system.

• Get Average Grade: Calculate and return the average grade for a student across all subjects.

• Get Subject Average: Calculate and return the average grade for a specific subject.

• List All Students: List all students with their average grades for each subject and overall.

• Get Highest Grade: Find the highest grade in a specific subject

```
class StudentGradesSystem:
    def __init__(self):
        self.students = {}
```

```
def add_student(self, name, grades):
    self.students[name] = grades
    print(f"Student {name} added with grades: {grades}")

def update_grade(self, name, subject_index, new_grade):
    if name in self.students:
        if 0 <= subject_index < len(self.students[name]):
            self.students[name][subject_index] = new_grade
            print(f"Grade updated for {name} in subject {subject_index + 1}: {new_grade}")
        else:
            print(f"Subject index {subject_index} is invalid.")
    else:
        print(f"Student {name} not found.")

def remove_student(self, name):
    if name in self.students:
        del self.students[name]
        print(f"Student {name} removed.")
    else:
        print(f"Student {name} not found.")

def get_average_grade(self, name):
    if name in self.students:
        grades = self.students[name]
        avg = sum(grades) / len(grades) if grades else 0
        return f"Average grade for {name}: {avg:.2f}"
    else:
```

```
        return f"Student {name} not found."

    def get_subject_average(self, subject_index):
        total, count = 0, 0
        for grades in self.students.values():
            if 0 <= subject_index < len(grades):
                total += grades[subject_index]
                count += 1
        if count > 0:
            return f"Average grade for subject {subject_index + 1}: {total / count:.2f}"
        else:
            return f"No data available for subject {subject_index + 1}."

    def list_all_students(self):
        if self.students:
            for name, grades in self.students.items():
                overall_avg = sum(grades) / len(grades) if grades else 0
                print(f"{name}: Grades: {grades} | Overall Average: {overall_avg:.2f}")
        else:
            print("No students available.")

    def get_highest_grade(self, subject_index):
        highest_grade = -1
        highest_student = None
        for name, grades in self.students.items():
            if 0 <= subject_index < len(grades):
                if grades[subject_index] > highest_grade:
```

```
                highest_grade = grades[subject_index]
                highest_student = name
        if highest_student:
            return f"Highest grade in subject {subject_index + 1}: {highest_grade} by {highest_student}"
        else:
            return f"No data available for subject {subject_index + 1}."

system = StudentGradesSystem()
system.add_student("Alice", [85, 90, 78])
system.add_student("Bob", [88, 76, 92])
system.add_student("Charlie", [95, 85, 80])

system.update_grade("Alice", 1, 95)
system.remove_student("Bob")
print(system.get_average_grade("Alice"))
print(system.get_subject_average(1))
system.list_all_students()
print(system.get_highest_grade(2))
```

OUTPUT:

```
Student Alice added with grades: [85, 90, 78]
Student Bob added with grades: [88, 76, 92]
Student Charlie added with grades: [95, 85, 80]
Grade updated for Alice in subject 2: 95
Student Bob removed.
Average grade for Alice: 86.00
Average grade for subject 2: 90.00
```

Alice: Grades: [85, 95, 78] | Overall Average: 86.00

Charlie: Grades: [95, 85, 80] | Overall Average: 86.67

Highest grade in subject 3: 80 by Charlie

Q2)Exercise 2: Employee Management System

Implement Employee Management System using nested dictionaries and lists and implement following functions to handle different operations.

- **add_employee(): Adds a new employee or updates an existing employee's details.**
- **update_salary(): Updates the salary of an existing employee.**
- **add_performance_score(): Adds a performance score to an employee's record.**
- **remove_employee(): Removes an employee from the records.**
- **get_average_salary_by_department(): Computes the average salary of employees in a specified department.**
- **get_employee_with_highest_performance(): Finds the employee with the highest average performance score.**
- **list_employees_by_department(): Lists all employees in a specified department.**

```
class EmployeeManagementSystem:
    def __init__(self):
        self.employees = {}

    def add_employee(self, employee_id, name, department, salary, performance_scores=None):
        if performance_scores is None:
            performance_scores = []
        self.employees[employee_id] = {
            "name": name,
            "department": department,
```

```
            "salary": salary,
            "performance_scores": performance_scores
        }
        print(f"Employee {name} added/updated.")

    def update_salary(self, employee_id, new_salary):
        if employee_id in self.employees:
            self.employees[employee_id]['salary'] = new_salary
            print(f"Salary updated for {self.employees[employee_id]['name']}: {new_salary}")
        else:
            print(f"Employee {employee_id} not found.")

    def add_performance_score(self, employee_id, score):
        if employee_id in self.employees:
            self.employees[employee_id]['performance_scores'].append(score)
            print(f"Performance score {score} added for {self.employees[employee_id]['name']}.")
        else:
            print(f"Employee {employee_id} not found.")

    def remove_employee(self, employee_id):
        if employee_id in self.employees:
            print(f"Employee {self.employees[employee_id]['name']} removed.")
            del self.employees[employee_id]
        else:
            print(f"Employee {employee_id} not found.")

    def get_average_salary_by_department(self, department):
```

```
    total_salary, count = 0, 0
    for employee in self.employees.values():
        if employee['department'] == department:
            total_salary += employee['salary']
            count += 1
    if count > 0:
        return f"Average salary in {department}: {total_salary / count:.2f}"
    else:
        return f"No employees found in {department}."

def get_employee_with_highest_performance(self):
    highest_avg = -1
    top_employee = None
    for employee_id, employee in self.employees.items():
        scores = employee['performance_scores']
        if scores:
            avg_score = sum(scores) / len(scores)
            if avg_score > highest_avg:
                highest_avg = avg_score
                top_employee = employee
    if top_employee:
        return f"Employee with highest performance: {top_employee['name']} with average score {highest_avg:.2f}"
    else:
        return "No performance scores available."

def list_employees_by_department(self, department):
    employees_in_dept = [emp['name'] for emp in self.employees.values() if emp['department'] == department]
```

```
        if employees_in_dept:
            print(f"Employees in {department}: {', '.join(employees_in_dept)}")
        else:
            print(f"No employees found in {department}.")

system = EmployeeManagementSystem()
system.add_employee(1, "Alice", "HR", 50000, [4.5, 4.7])
system.add_employee(2, "Bob", "IT", 60000, [4.0, 3.8])
system.add_employee(3, "Charlie", "HR", 55000, [4.8, 4.9])

system.update_salary(2, 65000)
system.add_performance_score(1, 4.6)
system.remove_employee(3)

print(system.get_average_salary_by_department("HR"))
print(system.get_employee_with_highest_performance())
system.list_employees_by_department("IT")
```

OUTPUT:

```
Student Alice added with grades: [85, 90, 78]
Student Bob added with grades: [88, 76, 92]
Student Charlie added with grades: [95, 85, 80]
Grade updated for Alice in subject 2: 95
Student Bob removed.
Average grade for Alice: 86.00
Average grade for subject 2: 90.00
Alice: Grades: [85, 95, 78] | Overall Average: 86.00
```

Charlie: Grades: [95, 85, 80] | Overall Average: 86.67

Highest grade in subject 3: 80 by Charlie