Problem 1: Understand REST API fundamentals

a) Definition and Characteristics of REST API

REST, **Representational State Transfer** is an architectural style used to design web services that allow different computer systems to communicate over the internet in a standardized way.

Characteristics of REST

1. **Statelessness**
   Each client request to the server must contain all the information required to process it. The server does not store any client session information between requests.
2. **Client–Server Architecture**
   The client and server are separate and independent. The client handles the user interface, while the server manages data storage and business logic.
3. **Uniform Interface**
   REST uses a consistent interface, typically through standard HTTP methods such as:
   - GET (acquire data)
   - POST (add data)
   - PUT (update data)
   - DELETE (remove data)
4. **Cacheable**
   Server responses specify whether they can be cached. This improves performance by allowing clients to reuse previously retrieved data when appropriate.

Resources and URIs

In REST, a **resource** is any data or service that can be accessed by a client. Each resource is identified by a unique URI (Uniform Resource Identifier).

Example: Student Management System

- Retrieve all students:
  `GET /api/iiuc/students`
- Retrieve a specific student by ID:
  `GET /api/iiuc/students/101`
- Retrieve all courses for a specific student:
  `GET /api/iiuc/students/101/courses`

Problem 2: Apply Spring Boot to build REST APIs

a) Employee Model Class

```
package com.iiuc.employeeman.models;

public class Employee{
    private Long id;
    private String name;
    private String email;
    private String designation;

    public Employee() {}
    public Employee(Long id, String name, String email, String
designation) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.designation = designation;
    }
    public Long getId() {return id;}
    public void setId(Long id) {this.id = id;}
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
    public String getEmail() {return email;}
    public void setEmail(String email) {this.email = email;}
    public String getDesignation() {return designation;}
    public void setDesignation(String designation) {this.designation =
designation;}
}
```

b) EmployeeController Class

```
package com.iiuc.employeeman.controllers;
import com.iiuc.employeeman.models.Employee;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("/employees")
public class EmployeeController{

    private List<Employee> employeeList = new ArrayList<>();
    @GetMapping
    public List<Employee> getAllEmployees() {
        return employeeList;
    }
    @PostMapping
    public Employee addEmployee(@RequestBody Employee employee) {
        employeeList.add(employee);
        return employee;
    }
    @PutMapping("/{id}")
    public Employee updateEmployee(@PathVariable Long id, @RequestBody
Employee updatedEmployee) {
        for (Employee emp: employeeList) {
            if (emp.getId().equals(id)) {
                emp.setName(updatedEmployee.getName());
                emp.setEmail(updatedEmployee.getEmail());
                emp.setDesignation(updatedEmployee.getDesignation());
                return emp;
            }
        }
        return null;
    }
    @DeleteMapping("/{id}")
    public String deleteEmployee(@PathVariable Long id) {
        for (int x=0; x<employeeList.size(); x++) {
            if (employeeList.get(x).getId().equals(id)) {
                employeeList.remove(x);
                return "Deletation Successful!";
            }
        }
        return "No Entries Found.";
    }
}
```

Problem 3: Analyze and test REST APIs

a) Testing Steps with Postman

1. **GET Request:** Set the method to GET and enter http://localhost:8080/employees. Click **Send** to see the list of employees.
2. **POST Request:** Set the method to POST and enter the URL. Go to the **Body** tab, select **raw**, choose **JSON**, and paste a JSON object:

```
{
  "id": 1,
  "name": "Cristiano",
  "email": "cristiano@football.com",
  "designation": "Centre Forward"
}
```

   Click **Send**.
3. **PUT Request (Update ID 1):** Set the method to PUT and enter http://localhost:8080/employees/1. Provide the updated JSON in the body and click Send.
4. **DELETE Request (Delete ID 3):** Set the method to DELETE and enter http://localhost:8080/employees/3. Click Send.

b) HTTP Status Codes Reference

Status Code	Meaning	Typical Usage
200 OK	The request was successful.	Standard response for successful GET, PUT, or DELETE.
201 Created	A new resource was created.	Response for a successful POST request.
400 Bad Request	The request was invalid.	Client sent malformed syntax or invalid data.
404 Not Found	Resource not found.	The requested ID or URL does not exist on the server.
500 Internal Error	Server-side failure.	An unexpected error occurred within the Spring Boot application logic.