2. **Problem Statement: Student Management System**
   Develop a Spring Boot application to manage student information for an educational institute. The system should allow the institute to add, update, delete, and view student records efficiently, along with searching and filtering capabilities.

   Answer :

```
package com.example.studentmanagement;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import javax.persistence.*;
import javax.validation.Valid;
import javax.validation.constraints.*;
import java.util.List;

@SpringBootApplication
public class StudentManagementApplication {
   public static void main(String[] args) {
      SpringApplication.run(StudentManagementApplication.class, args);
   }
}

@Entity
class Student {
   @Id @GeneratedValue
   private Long studentId;

   @NotBlank
   private String name;

   @Min(5) @Max(100)
   private int age;
```

```
    @NotBlank
    private String studentClass;

    @Email
    private String email;

    @NotBlank
    private String address;

    // getters and setters
}

interface StudentRepository extends JpaRepository<Student, Long> {
    Page<Student> findByNameContainingOrStudentClassContaining(String name, String
studentClass, Pageable pageable);
}

@Controller
@RequestMapping("/students")
class StudentController {

    private final StudentRepository studentRepository;

    public StudentController(StudentRepository studentRepository) {
        this.studentRepository = studentRepository;
    }

    @GetMapping
    public String listStudents(@RequestParam(defaultValue = "") String keyword, Pageable
pageable, Model model) {
        Page<Student> page =
studentRepository.findByNameContainingOrStudentClassContaining(keyword, keyword,
pageable);
        model.addAttribute("page", page);
        model.addAttribute("keyword", keyword);
        return "students/list";
    }

    @GetMapping("/new")
    public String showCreateForm(Model model) {
        model.addAttribute("student", new Student());
        return "students/form";
    }

    @PostMapping
    public String saveStudent(@Valid @ModelAttribute Student student, BindingResult result,
Model model) {
        if (result.hasErrors()) {
            return "students/form";
        }
        studentRepository.save(student);
        return "redirect:/students";
    }

    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable Long id, Model model) {
        Student student = studentRepository.findById(id).orElseThrow(() -> new
IllegalArgumentException("Invalid student Id:" + id));
        model.addAttribute("student", student);
        return "students/form";
    }

    @PostMapping("/update/{id}")
    public String updateStudent(@PathVariable Long id, @Valid Student student, BindingResult
result) {
        if (result.hasErrors()) {
            return "students/form";
        }
        student.setStudentId(id);
        studentRepository.save(student);
        return "redirect:/students";
    }

    @GetMapping("/delete/{id}")
    public String deleteStudent(@PathVariable Long id) {
        studentRepository.deleteById(id);
        return "redirect:/students";
    }
}
```