UST MOCK TEST SOLUTIONS

1] Library Management System

LibraryController.java

```
package com.wecp.library.controller;

import com.wecp.library.domain.Book;
import com.wecp.library.domain.User;
import com.wecp.library.repository.BookRepository;
import com.wecp.library.repository.UserRepository;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@RestController
@RequestMapping("/api/v1")
public class LibraryController {

    private final UserRepository userRepository;
    private final BookRepository bookRepository;

    public LibraryController(UserRepository userRepository, BookRepository bookRepository) {
        this.userRepository = userRepository;
        this.bookRepository = bookRepository;
    }
```

```
    /**
     * {@code GET /user/:id}: get the "id" User.
     * @param id the id of the user to retrieve.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the user, or if not found, returns with status "204 No Content".
     */
    @GetMapping("/user/{id}")
    public ResponseEntity<User> getUser(@PathVariable Long id) {
        Optional<User> user = userRepository.findById(id);
        return user.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.noContent().build());
    }

    /**
     * {@code POST /user}: Create a new user.
     * @param user the user to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the created user.
     */
    @PostMapping("/user")
    public ResponseEntity<User> createUser(@RequestBody User user) {
        User savedUser = userRepository.save(user);
        return ResponseEntity.ok(savedUser);
    }
    /**
     * {@code GET /book/:id}: get the "id" Book.
     * @param id the id of the book to retrieve.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the book, or if not found, returns with status "204 No Content".
     */
    @GetMapping("/book/{id}")
    public ResponseEntity<Book> getBook(@PathVariable Long id) {
        Optional<Book> book = bookRepository.findById(id);
```

```
        return book.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.noContent().build());
    }

    /**
     * {@code POST /book}: Create a new book.
     * @param book the book to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and the created book.
     */
    @PostMapping("/book")
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        Book savedBook = bookRepository.save(book);
        return ResponseEntity.ok(savedBook);
    }
}
```

BookRepository.java

```
package com.wecp.library.repository;
import com.wecp.library.domain.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface BookRepository extends JpaRepository<Book, Long> {
}
```

UserRepository.java

```
package com.wecp.library.repository;
import com.wecp.library.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
}
```

2] Property Manager Real Estate

PropertyController.java

```
package com.property.controller;

import com.property.entity.Property;
import com.property.service.PropertyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/properties")
public class PropertyController {

    private final PropertyService propertyService;

    @Autowired
    public PropertyController(PropertyService propertyService) {
        this.propertyService = propertyService;
```

```
    }

    // POST: Add a new property
    @PostMapping
    public Property addProperty(@RequestBody Property property) {
        return propertyService.addProperty(property);
    }

    // GET: Retrieve all properties
    @GetMapping
    public List<Property> getAllProperties() {
        return propertyService.getAllProperties();
    }
}
```

PropertyService.java

```
package com.property.service;

import com.property.entity.Property;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class PropertyService {
    private static int counter = 1; // Unique ID generator
    private final List<Property> propertyList = new ArrayList<>();

    // Add a new property
```

```
    public Property addProperty(Property property) {
        property.setId(counter++);
        propertyList.add(property);
        return property;
    }

    // Retrieve all properties
    public List<Property> getAllProperties() {
        return propertyList;
    }
}
```

Property.java

```
package com.property.entity;

public class Property {
    private int id;
    private String address;
    private String description;

    // Constructors
    public Property() {}

    public Property(int id, String address, String description) {
        this.id = id;
        this.address = address;
        this.description = description;
    }
```

```
    // Getters & Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

PropertyApplication.java

```
package com.property;

import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

DSA Codes

3] buggyIsValidNPI

```
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean buggyIsValidNPI(String npi) {
        // Check if the length is exactly 10
        if (npi.length() != 10) {
            return false;
        }

        int sum = 0;
        int len = npi.length();

        for (int i = len - 1; i >= 0; i--) {
            char c = npi.charAt(i);
```

```
            // Check if character is a digit
            if (!Character.isDigit(c)) {
                return false;
            }

            int digit = c - '0';

            // Double every second digit from the right
            if ((len - i) % 2 == 0) {
                digit *= 2;
                if (digit > 9) {
                    digit = digit - 9;  // Equivalent to summing the digits of a 2-digit number
                }
            }

            sum += digit;
        }

        // NPI is valid if sum is a multiple of 10
        return sum % 10 == 0;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String npi = scanner.nextLine();
        System.out.println(buggyIsValidNPI(npi));
        scanner.close();
    }
}
```

4] pyramidSum

```
import java.util.*;

public class Solution {
    public static int pyramidSum(int array_length, List<Integer> arr) {
        // Keep reducing the array length until only one element is left
        while (arr.size() > 1) {
            List<Integer> temp = new ArrayList<>();

            // Compute adjacent pair sums
            for (int i = 0; i < arr.size() - 1; i++) {
                temp.add(arr.get(i) + arr.get(i + 1));
            }

            // Update arr to be the new computed array
            arr = temp;
        }

        // The last remaining element is the pyramid sum
        return arr.get(0);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read input
        int array_length = scanner.nextInt();
        List<Integer> arr = new ArrayList<>();
```

```
        for (int i = 0; i < array_length; i++) {
            arr.add(scanner.nextInt());
        }

        // Compute and print pyramid sum
        System.out.println(pyramidSum(array_length, arr));

        scanner.close();
    }
}
```

5] Write a MySQL query to display details of all students who have scored MARKS within the range 400 and 6000 except those whose MARKS are 1200 and 5236.

```
SELECT *
FROM STUDENTS
WHERE MARKS BETWEEN 400 AND 6000
AND MARKS NOT IN (1200, 5236);
```
