BÁO CÁO SAU BUỔI THỰC HÀNH

Môn học: Lập trình Java Cơ bản

Tên buổi thực hành: Thực hành Java: Biến, Câu điều kiện, Vòng lặp, Mảng và Chuỗi

Thời gian thực hành: 180p

Họ và tên sinh viên: Lê Trung Đông

Mã sinh viên: PTIT-HN-010

Lớp: CNTT5

Nhóm: Nhóm 4

I. NỘI DUNG ĐÃ THỰC HÀNH

1. Trình bày các giải pháp

- **Xây dựng cấu trúc Menu:** Sử dụng vòng lặp do-while kết hợp switch-case để tạo giao diện tương tác liên tục, cho phép người dùng chọn lựa 5 bài toán thuật toán (LeetCode) mà không cần khởi động lại chương trình.
- **Xử lý Mảng (Array):** * Áp dụng kỹ thuật **Two Pointers** (Hai con trỏ) để tối ưu hóa bài toán *Two Sum* và *Move Zeroes*, giúp giảm độ phức tạp thời gian.
- **Xử lý Chuỗi & Regex:**
  - Sử dụng replaceAll("[^a-zA-Z0-9]", "") để làm sạch chuỗi trong bài *Valid Palindrome*.
  - Sử dụng split("\\s+") để xử lý các khoảng trắng thừa khi đảo ngược từ trong câu.
- **Xử lý logic số học:** Sử dụng vòng lặp và tính toán tổng bình phương các chữ số để kiểm tra *Happy Number*.

2. Liệt kê các câu hỏi phản biện

- **Câu hỏi:** Tại sao nên dùng StringBuilder thay vì String khi thực hiện nối chuỗi trong vòng lặp (như bài Reverse Words)?
- **Trả lời:** Vì String trong Java là *immutable* (không thể thay đổi). Mỗi khi nối chuỗi bằng toán tử +, Java sẽ tạo ra một đối tượng mới, gây tốn bộ nhớ. StringBuilder cho phép thay đổi nội dung trực tiếp trên cùng một vùng nhớ, giúp tối ưu hiệu năng.

3. Thực hành triển khai code ví dụ

- **Cấu trúc Menu:**

Java

```
int choice;
do {
    System.out.println("1. Two Sum \n2. Move Zeroes... \n0. Exit");
    choice = scanner.nextInt();
    switch(choice) {
        case 1: solveTwoSum(); break;
        // ... các case khác
    }
} while (choice != 0);
```

II. CÔNG VIỆC ĐÃ LÀM

1. Công việc cá nhân:

- Hoàn thành mức độ Trung bình & Khá: Triển khai thành công các bài toán Two Sum và Move Zeroes bằng mảng 1 chiều.
- Xây dựng bộ lọc Regex chuẩn để xử lý chuỗi phức tạp trong bài Valid Palindrome.
- Viết hàm tách và đảo ngược chuỗi sử dụng StringTokenizer hoặc split().

2. Công việc nhóm:

- Thảo luận và triển khai thuật toán cho bài **Happy Number** (Mức độ Giỏi): Sử dụng một tập hợp HashSet (hoặc mảng đánh dấu) để phát hiện vòng lặp vô hạn (infinite loop) nhằm tránh lỗi tràn bộ nhớ.

III. KẾT QUẢ ĐẠT ĐƯỢC

- **Hiểu rõ cú pháp Java:** Thành thạo việc khai báo biến, ép kiểu và sử dụng các lớp hỗ trợ như Scanner, Math, String.
- **Tư duy thuật toán:** Áp dụng thành công các kỹ thuật tối ưu hóa không gian (in-place) thay vì dùng mảng phụ.
- **Hoàn thành mục tiêu:** Đạt mức độ **Xuất sắc**, giải quyết được tất cả 5 nhóm chức năng trong SRS.
- **Kỹ năng debug:** Biết cách đọc lỗi ArrayIndexOutOfBoundsException và xử lý các đầu vào

rỗng.

IV. KHÓ KHĂN VÀ VẤN ĐỀ GẶP PHẢI

- **Khó khăn:** Xử lý các khoảng trắng thừa ở đầu, cuối và giữa các từ trong bài toán *Reverse Words*.
- **Lỗi gặp phải:** Nhập dữ liệu chuỗi sau khi nhập số (nextInt()) bị trôi lệnh (không cho nhập chuỗi).
- **Khắc phục:** Sử dụng thêm một lệnh scanner.nextLine() để xóa bộ nhớ đệm (buffer) trước khi đọc chuỗi mới.
- **Phản biện:** Cách xử lý số 0 trong bài *Move Zeroes* mà không làm thay đổi thứ tự các số khác yêu cầu logic đổi chỗ (swap) cẩn thận để tránh mất dữ liệu.

V. KINH NGHIỆM RÚT RA

- **Lưu ý:** Luôn kiểm tra điều kiện biên (mảng rỗng, chuỗi có độ dài bằng 1) trước khi vào logic chính.
- **Kỹ thuật:** Sử dụng while cho các bài toán chưa biết rõ số lần lặp (Happy Number) và for cho các bài toán thao tác trên mảng.

VI. ĐỀ XUẤT / KIẾN NGHỊ

- Mong muốn được tìm hiểu thêm về các cấu trúc dữ liệu nâng cao như HashMap để tối ưu bài toán *Two Sum* từ $O(n^2)$ xuống $O(n)$.

VII. KẾT LUẬN

- **Tự đánh giá:** Hoàn thành 100% yêu cầu. Nắm vững nền tảng Java Core để sẵn sàng cho phần Lập trình hướng đối tượng (OOP).
- **Ứng dụng:** Các thuật toán xử lý chuỗi và mảng này là nền tảng quan trọng để giải quyết các bài toán thực tế khi làm việc với dữ liệu người dùng sau này.