NỘI DUNG ÔN TẬP GIỮA KỲ

MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

A. NỘI DUNG ÔN TẬP

- Linked List: Xây dựng lớp Linked List hiện thực ListInterface và xây dựng các phương thức thêm, xóa, truy vấn trên danh sách liên kết.
- Stack, Queue: Xây dựng được lớp Stack, Queue theo các phương pháp đã học. Sử dụng thư viện Stack, Queue trong Java và giải các bài toán dùng Stack, Queue.
- Đệ quy: Giải các bài toán thông thường bằng đệ quy, đệ quy giải các công thức truy hồi, đệ quy trên cấu trúc dữ liệu (mảng, danh sách liên kết).
- Sắp xếp: Hiện thực các thuật toán sắp xếp đã học.

B. BÀI TẬP ÔN MẪU

YÊU CẦU SINH VIÊN ĐỌC KỸ TRƯỚC KHI LÀM BÀI:

1. Tạo một thư mục đặt tên là **MSSV_Hotên** (viết liền, không dấu).
2. **Sinh viên nhận 03 thư mục Cau1, Cau2 và Cau3,** sao chép 03 thư mục nhận được vào thư mục MSSV_Hotên vừa tạo.
3. Sau khi làm bài xong, sinh viên nén thư mục MSSV_Hotên và nộp. Bài nộp đúng là bài nộp có thư mục MSSV_Hotên chứa 03 thư mục Cau1, Cau2 và Cau3 chứa bài làm.

PHẦN CÂU HỎI

Sinh viên lưu ý đảm bảo bài làm của từng câu phải biên dịch, chạy được và đặt đúng thư mục theo hướng dẫn. Tất cả các trường hợp **làm sai yêu cầu**, sinh **lỗi** trong **quá trình biên dịch** hoặc trong **quá trình chạy** khi chấm đều bị **0 điểm cả câu**.

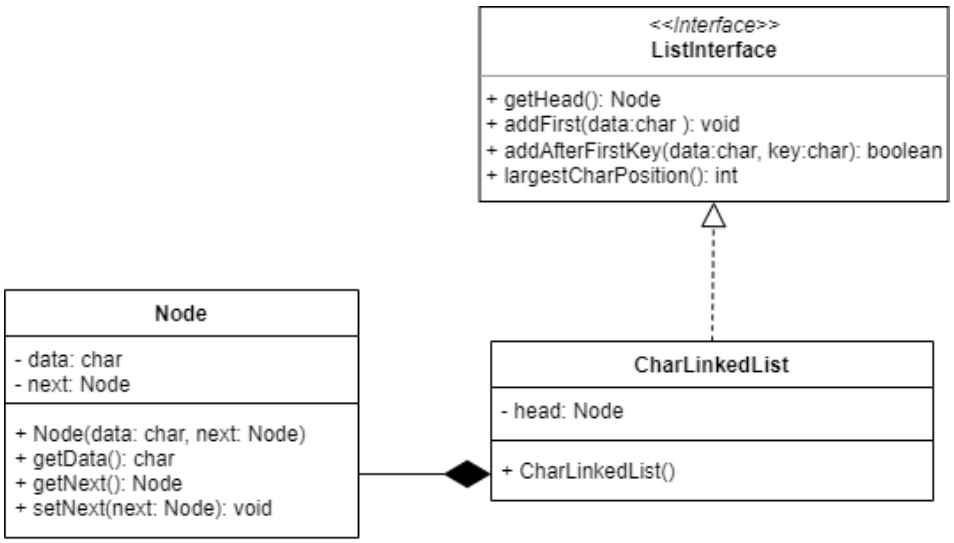
Phần hàm main cho sẵn ở các câu trong đề chỉ để sinh viên kiểm thử bài làm, sinh viên không nhất thiết phải xóa hàm main khi nộp nhưng phải đảm bảo hàm main không gây lỗi cho bài làm.

Sinh viên chú ý KHÔNG chỉnh sửa tên file, tên hàm, thứ tự tham số trong các file đã nhận và làm bài trực tiếp trên các file này.

Thư mục bài làm đúng khi hoàn chỉnh có cấu trúc như sau (52100000_NguyenVanAn thay thế bởi thông tin tương ứng của sinh viên):

```
52100000_NguyenVanAn
├── Cau1
│   ├── CharLinkedList.java
│   ├── ListInterface.java
│   ├── Node.java
│   └── Test.java
├── Cau2
│   └── Cau2.java
└── Cau3
    └── Cau3.java
```

Câu 1:



Trong thư mục Cau1, dựa vào sơ đồ lớp trên và hai file **ListInterface.java** và **Node.java** đính kèm trong thư mục (Sinh viên không chỉnh sửa hoặc thêm code trên hai file này), sinh viên định nghĩa lớp **CharLinkedList** tạo ra danh sách liên kết chứa các ký tự.

Định nghĩa các phương thức trong lớp **CharLinkedList**:

- **CharLinkedList():** Phương thức khởi tạo không tham số, gán ***head* = *null***
- **getHead():** trả về ***head*** của danh sách liên kết.
- **addFirst(char data):** thêm vào đầu danh sách một nút có giá trị ***data***.
- **addAfterFirstKey(char data, char key):** thêm một nút với giá trị ***data*** vào sau nút chứa giá trị ***key*** đầu tiên tính từ đầu danh sách sau đó trả về *true*. Nếu danh sách không có giá trị nào bằng ***key*** nào thì không thêm và trả về *false*.

  Ví dụ: Cho danh sách 'A', 'b', 'c' với ***head*** là phần tử có giá trị 'A'. Ta gọi phương thức **addAfterFirstKey('E', 'b')** thì kết quả danh sách sẽ là 'A', 'b', 'E', 'c'.
- **largestCharPostition():** trả về vị trí của phần tử đầu tiên có mã ASCII lớn nhất trong danh sách. Vị trí đầu tiên của ***head*** bắt đầu từ 0. Nếu danh sách rỗng thì trả về -1.
  Ví dụ: Cho danh sách 'A', 'b', 'c' với ***head*** là phần tử có giá trị 'A'. Ta gọi phương thức **largestCharPostition()** thì kết quả trả về là 2 ('c' có mã lớn nhất).

Sinh viên tự định nghĩa phương thức in danh sách ra màn hình và định nghĩa một lớp chứa phương thức *main* để kiểm tra lại bài làm.

Câu 2: Trong thư mục **Cau2** chứa sẵn file **Cau2.java**, trong lớp **Cau2** sinh viên định nghĩa phương thức **public static int recur(int n, int k)** để giải bài toán sau bằng phương pháp **đệ quy** và trả về kết quả.

$$A(n,k) = \begin{cases} 1, & k = 0 \\ n * A(n, k-1), & k > 0 \end{cases} \quad Với\ n\ và\ k\ là\ số\ nguyên\ và\ luôn\ \geq 0$$

Sinh viên tự hiện thực trong phương thức ***main*** để kiểm tra lại bài làm. Lưu ý phương thức ***main*** gây lỗi cho bài làm thì sinh viên bị 0 điểm cả bài.

Câu 3 (3 điểm): Trong thư mục **Cau3** chứa sẵn file **Cau3.java**, trong lớp **Cau3** sinh viên định nghĩa phương thức **public static int calculate(String[] expression)** dùng Stack để tính toán theo giải thuật mô tả bên dưới.

Tại câu này, sinh viên được phép sử dụng thư viện Stack có sẵn của Java bằng cách thêm thư viện chứa Stack với câu lệnh import java.util.Stack;

Cho một mảng String gồm các phần tử số nguyên, phép toán cộng (+) hoặc trừ (-) sinh viên thực hiện tính và trả về kết quả theo giải thuật sau:

1. Tạo một Stack<Integer>
2. Xét từng phần tử trong mảng String
   2.1. Nếu phần tử là số thì đưa vào Stack
   2.2. Nếu phần tử là phép toán thì lấy 2 phần tử từ Stack ra, gọi phần tử đầu tiên lấy ra là **o1**, phần tử lấy ra thứ 2 là **o2**, ta thực hiện tính toán **o3** = **o2** <*phép toán*> **o1** và đưa lại **o3** vào Stack.
3. Kết quả cuối cùng trong Stack chính là kết quả phải trả về.

Ví dụ cho mảng String gồm các phần tử: “3”, “4”, “+”, “2”, “1”, “+”, “-”

Duyệt mảng	Stack	Tính toán
3	3	
4	4 3	
+	7	3 + 4 = 7
2	2 7	
1	1 2 7	
+	3 7	2 + 1 = 3
-	4	7 – 3 = 4
Kết quả trả về là 4		

Sinh viên có thể định nghĩa phương thức **public static boolean isNumber(String str)** theo đoạn code sau để kiểm tra một phần tử String có phải số hay không:

```
private static boolean isNumber(String str) {
        return str.matches("0|([1-9][0-9]*)");
}
```

Sinh viên tự hiện thực trong phương thức ***main*** để kiểm tra lại bài làm. Lưu ý phương thức ***main*** gây lỗi cho bài làm thì sinh viên bị 0 điểm cả bài.