BÁO CÁO THỰC HÀNH DSA TUẦN 4:

24120261 – Đặng Bùi Thế Bảo

**Linkedlist:*

- Hàm NODE *createNode(int data): tạo một node mới cấp phát truyền dữ liệu → return
- Hàm List *createList(NODE *p_node): tạo list mơi, cấp phát → thêm node đầu tiên vào, return
- Các hàm add như addHead, addTail: thay đổi liên kết giữa các node với con trỏ head hoặc tail để thêm node mới vào.
- Tương tự với những hàm chỉ cần duyện xuôi mảng để thao tác thì chỉ cần tìm được vị trí chính xác rồi thao tác như addAfter. removeAfter,...
- Các hàm như removeBefore, removePos,... removeTail: ta thực hiện việc duyệt list tìm phần tử đứng trước phần tử cần thao tác rồi thực hiện thao tác.
- Hàm reverseList: tạo một list mới rồi lấy các node ở list cũ liên tc thực hiện thao tác addHead vào list mới cho đến khi hết node trong list cũ → return list mới.
- Hàm removeDuplicate: cần 3 con trỏ 1 con trỏ lưu vị trí head, 2 con trỏ duyệt mảng một con chạy trước một con chạy sau. Con trỏ chạy trước duyệt đến node nào có giá trị bằng giá trị con trỏ chạy sau thì thực hiện thao tác xóa node đó.
- Hàm removeElement thì tương tự các hàm remove khác. Duyệt đến node đứng trước node mang giá trị cần xóa. Rồi thực hiện thao tác xóa.
- Ghi chú: việc duyện đến node trước node cần thao tác có mục đích là để nó thể nối liền danh sách lại khi node cần thao tác bị xóa.

**Doubly Linkedlist:*

- Hầu như làm tương tự như Linkedlist. Nhưng mà các thao tác như xóa hay thêm nó đơn giản hơn.
- Thay vì phải duyệt từ đầu tới cuối để để tìm vị trí trước vị trí cần xóa thì ta có thể duyệt hai chiều và thao tác hai chiều.
- Nhưng mà độ phức tạp vẫn tăng lên vì chúng ta cần đồng thời thao tác với cả 2 liên kết: một liên kết p_next để trỏ tới node phía sau và một liên kết p_prev để trỏ đến các node phía trước.
- Một ví dụ để có thể so sánh:
    - Hàm void removeTail(List *&L): ta không cần phải cất công tìm đến node phía trước tail để duy trì liên kết mà trực tiếp thay đổi liên kết giữa node cuối, kế cuối, và tail.