Homework #1: Linked list 개발

- 과제 내용

  - Linked list 프로그램을 구현하고 실험한다. 주어진 데이터를 linked list에 모두 저장하고, 새로운 데이터 추가, 기존 데이터 삭제, 검색, 리스트의 데이터 순서를 역순으로 재배치 하는 작업을 실험해본다.

  - 삽입할 데이터는 텍스트 파일 student_data.txt에 들어 있는 학생 명단이며 각 학생의 명단은 이름(문자열)과 학번(정수), 전화번호(문자열), 거주지역(문자열)로 구성된 구조체로 아래와 같이 정의한다.

- 데이터 타입 선언

```
//하나의 데이터 아이템의 정의는 다음과 같다.
    typedef struct Listdata {
        char name[30];
        int id;
        char phone[20];
        char province[30];
    } listData;
//하나의 리스트 노드의 정의는 다음과 같다.
    typedef struct Listnode* type_listNode_pointer;
    typedef struct Listnode {
        listData data;
        type_listNode_pointer link;
    } listNode;
//리스트의 첫 번째 노드를 가리키는 head는 다음과 같이 정의한다.
    typedef struct LinkedList {
        type_listNode_pointer head;
        int length;
    } linkedList;
```

- 작업 수행 순서

(1) 리스트 저장

- 프로그램을 실행하면 먼저 주어진 student_data.txt 파일 안의 각 줄의 학생 정보를 차례로 읽어 linked list에 저장한다. 삽입할 노드는 항상 현재 리스트의 마지막 노드가 되도록 저장하는 insertLast( ) 함수를 작성하여, 이를 매 학생마다 반복적으로 호출하여 사용하도록 한다.

  연결리스트에 대한 헤더정보를 가지는 변수는 다음처럼 main 함수의 지역변수로 선언한다 (변수명은 아무 이름이나 가능함. 아래 예시에서는 LL 로 한다고 가정함):

  linkedList **LL** = {NULL, 0 }; // 주의: 이 줄은 main 함수 내에 선언할 것!!

  함수 정의: void insertLast (linkedList* L, listData item)

(2) 테스트

- 명령문을 입력 받아 이를 수행하는 작업을 반복하도록 한다.
- 명령문 입력 시, scanf( ) 함수를 사용하지 않도록 하며 대신에 gets 로 명령문 한 줄 전체을 읽어 온 후 먼저 명령의 종류을 파악하고 그에 따라 후속되는 arguments를 알아내는 방법을 사용하도록 한다.
- 명령문 종류: (한 명령의 모든 정보를 여러 줄에 넣지 말고 반드시 한 줄에 넣도록 할 것.)

print 설명: 리스트의 내용을 차례대로 출력한다. 함수 정의: void printList(linkedList* L)
search 3078 설명: 리스트에서 학번이 3078인 노드를 찾아 학생의 이름, 학번, 전화번호, 거주지역 정보를 같은 한 줄에 출력하고 찾은 노드의 주소를 리턴한다. 탐색에 실패할 경우 탐색 실패 메시지를 출력하고 NULL을 리턴한다. 함수 정의: type_listNode_pointer search(linkedList* L, int x)
insert 5006, 8765, 홍길동, 010-4532-4678, 충남 설명: 리스트에서 search( ) 함수를 통해 학번이 5006인 노드를 먼저 찾고, 해당 노드 뒤에 학번이 8765, 이름 홍길동, 전화 010-4532-4678, 거주지역이 충남인 노드를 추가한 후 리스트의 전체 내용을 차례대로 출력한다. 삽입할 노드의 이전 노드가 리스트에 존재하지 않거나 삽입할 학번의 학생이 이미 리스트 내에 존재하는 경우 삽입이 실패하도록 한다. 삽입에 실패할 경우 삽입 실패 메시지를 출력한다. 함수 정의: void insert(linkedList* L, listNode* pre, listData x) 주의: 이 함수를 호출 전에 main 함수에서 먼저 삽입할 학생 정보를 모두 가진 구조체를 준비한 후 이를 함수 호출시에 insert의 3 번째 인수로 전달해야 한다.

따라서 3 번째 인수의 type은 listData 이어야 한다. 전체 리스트 내용 출력도 함수 안에서 수행하도록 한다(단, 이 작업은 main에서 수행해도 무방함).
delete 3088 설명: 리스트에서 학번이 3088인 노드를 찾아 해당 노드를 삭제한 후 리스트의 내용을 차례대로 출력한다. 삭제에 실패할 경우 삭제 실패 메시지를 출력한다. 아래 정의된 함수 delete 안에서 위 작업 모두를 수행하도록 한다: 함수 정의: void delete(linkedList* L, int x)
reverse 리스트의 노드들의 순서를 역순으로 바꾼 리스트가 되도록 변경한다. 그 다음 리스트의 내용을 차례대로 출력한다. 정렬이 아님을 주의한다. 함수 정의: void reverse(linkedList* L)
getLength 설명: 리스트의 노드 개수를 출력한다. 함수 정의: int getLength(linkedList* L)
exit 설명: 명령문 수행의 반복 작업을 종료한 후 프로그램을 종료한다.

- 제출

- 프로그램 소스 코드와 보고서.

- 보고서 포함 내용

  - 모든 명령에 대해 수행해 본 실행창을 캡처한 이미지.
  - 과제 수행 중 어려웠던 점, 좋았던 점을 보고서에 간단하게 기술. 특히 어떤 도움을 받아서 프로그램 작성, 오류 해결 등의 난관을 극복하게 되었는지 밝혀 주기 바람.