프로그래밍 과제 04

1. 다음과 같이 작동하는 프로그램을 작성하라. 이 프로그램은 사용자로부터 정수들을 하나씩 입력 받아서 연결리스트의 지정된 위치에 삽입하고, 검색하고, 삭제하는 일을 한다. 먼저 프로그램이 시작되면 화면에 `$`를 출력하고 사용자의 명령을 대기하며 `add`, `find`, `count`, `delete`, `list`, 그리고 `exit`의 6가지 명령을 지원한다.

   - `add` 명령은 다음과 같은 형식이다.

     ```
     $ add index value
     ```

     여기서 `value`는 추가할 정수이고 `index`는 정수를 추가할 위치이다. 엄밀하게 말하면 추가된 값의 연결리스트에서의 인덱스가 `index`가 되어야 한다. 이때 `index`는 0에서 시작한다. 즉 `index`가 0이면 연결리스트의 맨 앞에 추가하라는 의미이고, `index`가 1이면 추가한 값이 결과적으로 두 번째 노드가 되도록 하라는 의미이다. 만약 `index`값이 유효한 범위를 벗어나면 `invalid index`라고 출력한다.

   - `find` 명령은 다음과 같은 형식이다.

     ```
     $ find value
     ```

     이 명령이 하는 일은 연결리스트에 정수 `value`를 찾아서 그 위치, 즉 index를 출력하는 것이다. 정수 `value`가 연결리스트에 여러 개 저장되어 있을 경우에는 그 중 첫 번째 값의 위치를 출력하고, 존재하지 않을 경우에는 -1을 출력한다.

   - `count` 명령은 다음과 같은 형식이다.

     ```
     $ count value
     ```

     이 명령이 하는 일은 연결리스트에 정수 `value`가 몇 개 저장되어 있는지 카운트하여 그 개수를 출력하는 것이다.

   - `delete` 명령은 다음과 같은 형식이다.

     ```
     $ delete value
     ```

     이 명령은 연결리스트에서 정수 `value`를 찾아서 연결리스트로 부터 삭제한다. 정수 `value`가 여러 개 저장되어 있는 경우에는 그 중 첫 번째 값을 제거한다. 만약 정수 `value`가 존재하지 않으면 `not exist`라고 출력한다.

   - `list`명령은 연결리스트에 저장된 정수들을 순서대로 한 줄로 출력하는 것이고, 마지막으로 `exit` 명령은 프로그램을 종료한다.

   프로그램 실행 예:

   ```
   $ add 1 5
   invalid index
   $ add 0 5
   $ find 5
   0
   $ add 0 6
   $ find 5
   1
   $ add 2 7
   $ list
   6 5 7
   ```

```
$ add 4 5
invalid index
$ add 3 5
$ list
6 5 7 5
$ count 5
2
$ find 5
1
$ find 8
-1
$ add 1 7
$ list
6 7 5 7 5
$ add 5 7
$ list
6 7 5 7 5 7
$ delete 8
not exist
$ delete 7
$ list
6 5 7 5 7
$ count 7
2
$ add 3 6
$ list
6 5 7 6 5 7
$ delete 7
$ list
6 5 6 5 7
$ delete 7
$ list
6 5 6 5
$ count 5
2
$ exit
```

2. 아래의 프로그램의 목적은 단어들을 입력받은 후 입력된 단어들의 등장 빈도를 카운트하고 사전식 순서로 정렬하여 출력하는 것이다. 단어 **"EOF"**가 입력되면 종료한다. 함수 **insert**를 완성하라. 함수 **insert**는 입력된 문자열을 연결리스트에 반영하고 **head**노드의 주소를 반환한다. 여기서 반영한다는 것은 이미 존재하는 문자열이면 카운트만 증가하고, 아니라면 정렬된 순서로 연결리스트에 추가한다는 의미이다. 함수 **insert** 외부의 어떤 부분도 수정하거나 전역변수를 추가해서는 안된다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLEN 100

typedef struct  {      /* 하나의 단어와 그 단어의 빈도수를 저장하기 위한 구조체 */
    char *word;
```

```
    int freq;
} Item;

struct node {          /* 연결리스트의 노드의 구조를 정의하는 구조체 */
    Item *data;
    struct node *next;
};
typedef struct node Node;

Node *insert(Node *, char buf[]);

int main()
{
    char buf[MAXLEN];
    Node *head  = NULL;            /* head는 지역변수이다. */

    while(1) {
        scanf("%s", buf);
        if (strcmp(buf, "EOF") == 0) break;  /* 문자열 "EOF"를 입력하면 종료 */

        head = insert(head, buf);        /* 입력된 문자열을 연결리스트에 반영 */
    }

    Node *p = head;
    while(p≠NULL) {
        printf("%s:%d\n", p->data->word, p->data->freq);
        p = p->next;
    }
    return 0;
}

Node *insert(Node *head, char buf[]) {
    /* 입력된 문자열을 연결리스트에 반영하고 head노드의 주소를 반환한다. 이미 존재하는.  */
    /* 문자열이면 카운트만 증가 하고, 아니라면 정렬된 순서로 연결리스트에 추가한다.      */

}
```

입력 예:

```
head hello part one is head hello hello
and board is cool EOF
```

출력 예:

```
and:1
board:1
cool:1
head:2
hello:3
is:2
one:1
part:1
```

3. 2번과 동일한 문제이다. 다만 함수 **insert**의 signature만 다음과 같이 변경되었다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
#define MAXLEN 100

typedef struct  {      /* 하나의 단어와 그 단어의 빈도수를 저장하기 위한 구조체 */
    char *word;
    int freq;
} Item;

struct node {          /* 연결리스트의 노드의 구조를 정의하는 구조체 */
    Item *data;
    struct node *next;
};
typedef struct node Node;

void insert(Node **, char buf[]);

int main()
{
    char buf[MAXLEN];
    Node *head  = NULL;            /* head는 지역변수이다. */

    while(1) {
        scanf("%s", buf);
        if (strcmp(buf, "EOF") == 0) break;  /* 문자열 "EOF"를 입력하면 종료 */

        insert(&head, buf);        /* 입력된 문자열을 연결리스트에 반영 */
    }

    Node *p = head;
    while(p≠NULL) {
        printf("%s:%d\n", p->data->word, p->data->freq);
        p = p->next;
    }
    return 0;
}

void insert(Node **ptr_head, char buf[]) {
    /* 입력된 문자열을 연결리스트에 반영한다. 즉, 이미 존재하는                    */
    /* 문자열이면 카운트만 증가 하고, 아니라면 정렬된 순서로 연결리스트에 추가한다.  */


}
```