#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

typedef struct SinglyLinkedListNode SinglyLinkedListNode;

typedef struct SinglyLinkedList SinglyLinkedList;

struct SinglyLinkedListNode {

    int data;

    SinglyLinkedListNode\* next;

};

struct SinglyLinkedList {

    SinglyLinkedListNode\* head;

    SinglyLinkedListNode\* tail;

};

SinglyLinkedListNode\* create\_singly\_linked\_list\_node(int node\_data) {

    SinglyLinkedListNode\* node = malloc(sizeof(SinglyLinkedListNode));

    node->data = node\_data;

    node->next = NULL;

    return node;

}

void insert\_node\_into\_singly\_linked\_list(SinglyLinkedList\*\* singly\_linked\_list, int node\_data) {

    SinglyLinkedListNode\* node = create\_singly\_linked\_list\_node(node\_data);

    if (!(\*singly\_linked\_list)->head) {

        (\*singly\_linked\_list)->head = node;

    } else {

        (\*singly\_linked\_list)->tail->next = node;

    }

    (\*singly\_linked\_list)->tail = node;

}

void print\_singly\_linked\_list(SinglyLinkedListNode\* node, char\* sep, FILE\* fptr) {

    while (node) {

        fprintf(fptr, "%d", node->data);

        node = node->next;

        if (node) {

            fprintf(fptr, "%s", sep);

        }

    }

}

void free\_singly\_linked\_list(SinglyLinkedListNode\* node) {

    while (node) {

        SinglyLinkedListNode\* temp = node;

        node = node->next;

        free(temp);

    }

}

/\*

 \* Complete the 'insertNodeAtPosition' function below.

 \*

 \* The function is expected to return an INTEGER\_SINGLY\_LINKED\_LIST.

 \* The function accepts following parameters:

 \*  1. INTEGER\_SINGLY\_LINKED\_LIST llist

 \*  2. INTEGER data

 \*  3. INTEGER position

 \*/

/\*

 \* For your reference:

 \*

 \* SinglyLinkedListNode {

 \*     int data;

 \*     SinglyLinkedListNode\* next;

 \* };

 \*

 \*/

SinglyLinkedListNode\* insertNodeAtPosition(SinglyLinkedListNode\* head, int data, int position) {

SinglyLinkedListNode\* newNode=(SinglyLinkedListNode\*)malloc(sizeof(SinglyLinkedListNode));

newNode->data=data;

newNode->next=NULL;

if(position==0){

    newNode->next=head;

    return newNode;

}

SinglyLinkedListNode\* curr=head;

for (int i=0;i<position-1 && curr!=NULL;i++){

    curr=curr->next;

}

newNode->next=curr->next;

curr->next=newNode;

return head;

}

int main()

{

    FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

    SinglyLinkedList\* llist = malloc(sizeof(SinglyLinkedList));

    llist->head = NULL;

    llist->tail = NULL;

    char\* llist\_count\_endptr;

    char\* llist\_count\_str = readline();

    int llist\_count = strtol(llist\_count\_str, &llist\_count\_endptr, 10);

    if (llist\_count\_endptr == llist\_count\_str || \*llist\_count\_endptr != '\0') { exit(EXIT\_FAILURE); }

    for (int i = 0; i < llist\_count; i++) {

        char\* llist\_item\_endptr;

        char\* llist\_item\_str = readline();

        int llist\_item = strtol(llist\_item\_str, &llist\_item\_endptr, 10);

        if (llist\_item\_endptr == llist\_item\_str || \*llist\_item\_endptr != '\0') { exit(EXIT\_FAILURE); }

        insert\_node\_into\_singly\_linked\_list(&llist, llist\_item);

    }

    char\* data\_endptr;

    char\* data\_str = readline();

    int data = strtol(data\_str, &data\_endptr, 10);

    if (data\_endptr == data\_str || \*data\_endptr != '\0') { exit(EXIT\_FAILURE); }

    char\* position\_endptr;

    char\* position\_str = readline();

    int position = strtol(position\_str, &position\_endptr, 10);

    if (position\_endptr == position\_str || \*position\_endptr != '\0') { exit(EXIT\_FAILURE); }

    SinglyLinkedListNode\* llist\_head = insertNodeAtPosition(llist->head, data, position);

    char \*sep = " ";

    print\_singly\_linked\_list(llist\_head, sep, fptr);

    fprintf(fptr, "\n");

    free\_singly\_linked\_list(llist\_head);

    fclose(fptr);

    return 0;

}

char\* readline() {

    size\_t alloc\_length = 1024;

    size\_t data\_length = 0;

    char\* data = malloc(alloc\_length);

    while (true) {

        char\* cursor = data + data\_length;

        char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

        if (!line) { break; }

        data\_length += strlen(cursor);

        if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') { break; }

        size\_t new\_length = alloc\_length << 1;

        data = realloc(data, new\_length);

        if (!data) { break; }

        alloc\_length = new\_length;

    }

    if (data[data\_length - 1] == '\n') {

        data[data\_length - 1] = '\0';

    }

    data = realloc(data, data\_length);

    return data;

}