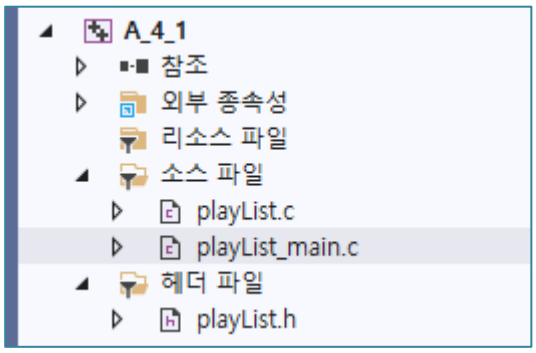
4장. 응용예제 답

응용예제 03 **나만의 플레이리스트 만들기**



[솔루션 탐색기]

playlist.h

```
#pragma once

// 전체 음악목록_playList의 노드 구조를 구조체로 정의
typedef struct playListNode {
        char data[30]; //음악제목
        struct playListNode* link;
} playListNode;

typedef struct {
        playListNode* head;
} playListNode_h;

// 선택된 나만의 플레이리스트_selectsList의 노드 구조를 구조체로 정의
typedef struct selectsListNode {
        struct playListNode* data; //선택한 음악에 대한 playList 노드의 주소
        struct selectsListNode* link;
} selectsListNode;

typedef struct {
        selectsListNode* head;
} selectsListNode_h;

//전체 음악목록_playList의 함수 선언
playListNode_h* creatPlayLinkedList_h(void);
void insertPlayListLastNode(playListNode_h* CL, char* x);
playListNode* searchNode(playListNode_h* CL, char* x);

//선택된 나만의 플레이리스트_selectsList의 함수 선언
selectsListNode_h* createSelectsLinkedList_h(void);
//void insertSelectsListLastNode(selectsListNode_h* sCL, playListNode* p_temp);
void insertSelectsListLastNode(selectsListNode_h* sCL, char* p_temp);
void printSelectsList(selectsListNode_h* CL);
```

playlist.c

```
#define _CRT_SECURE_NO_WARNINGS  //strcpy() 함수에 대한 경고 해제
#include <stdio.h>
#include <string.h>
#include "playList.h"

//전체 음악목록_playList의 함수 정의 =>>> 예제 4-2에서 구조체 타입 이름 수정
// 예제 4-2의 createLinkedList_h()
playListNode_h* creatPlayLinkedList_h(void){
        playListNode_h* CL;
        CL = (playListNode_h*)malloc(sizeof(playListNode_h));      // 헤드 노드 할당
        CL->head = NULL;
// 공백 리스트이므로 NULL로 설정
        return CL;
}

// 예제 4-2의 insertFirstNode()에서 마지막 명령어(CL->head = newNode;) 삭제
void insertPlayListLastNode(playListNode_h* CL, char* x) {
        playListNode* newNode, * temp;
        newNode = (playListNode*)malloc(sizeof(playListNode));     // 삽입할 새 노드 할당
        strcpy(newNode->data, x);
        if (CL->head == NULL) {           // 원형 연결 리스트가 공백인 경우
                CL->head = newNode;       // 새 노드를 리스트의 시작 노드로 연결
                newNode->link = newNode;
        }
        else {                                          // 원형 연결 리스트가 공백이 아닌
경우
                temp = CL->head;
                while (temp->link != CL->head)
                        temp = temp->link;
                newNode->link = temp->link; // 마지막 노드를 첫 번째 노드인 new와 원형 연결
                temp->link = newNode;

        }
}

// 예제 4-2의 searchNode()
playListNode* searchNode(playListNode_h* CL, char* x) {
        playListNode* temp;
        temp = CL->head;

        if (temp == NULL) return NULL;
        do {
                if (strcmp(temp->data, x) == 0) return temp;
                else temp = temp->link;
        } while (temp != CL->head);
        printf("₩n입력한 음악을 찾지못했습니다 ₩n");
        return NULL;
}
```

```
//선택된 나만의 플레이리스트_selectsList의 함수 정의  =>>> 예제 4-2에서 구조체 타입 이름 수정
// 예제 4-2의 createLinkedList_h()
selectsListNode_h* createSelectsLinkedList_h(void) {
        selectsListNode_h* sCL;
        sCL = (selectsListNode_h*)malloc(sizeof(selectsListNode_h));// 헤드 노드 할당
        sCL->head = NULL;
                // 공백 리스트이므로 NULL로 설정
        return sCL;
}

// 예제 4-2의 insertFirstNode() 수정
void insertSelectsListLastNode(selectsListNode_h* sCL, playListNode* p_temp) {
        selectsListNode* newNode, *s_temp;
        newNode = (selectsListNode*)malloc(sizeof(selectsListNode));       // 삽입할 새 노드
할당
        newNode->data = p_temp; //선택한 음악의 주소를 selectsListNode에 저장

        if (sCL->head == NULL) {          // 원형 연결 리스트가 공백인 경우
                sCL->head = newNode;      // 새 노드를 리스트의 시작 노드로 연결
                newNode->link = newNode;
        }
        else {
                //selectsList의 마지막 노드 찾기
                s_temp = sCL->head;
                while (s_temp->link != sCL->head)
                        s_temp = s_temp->link;

                // selectsList의 마지막 노드 뒤에 새 노드 연결
                newNode->link = s_temp->link;
                s_temp->link = newNode;

        }
}

// 예제 4-2의 printList() 수정
void printSelectsList(selectsListNode_h* CL) {
        selectsListNode* s;

        printf("₩n₩nmyPlayList : ");
        s = CL->head;
        if (s == NULL)   return;
        do {
                printf("  %s", s->data->data);
                s = s->link;
        } while (s != CL->head);
        printf("₩n₩n");
}
```

playList_main.c

```c
//응용예제03. 나만의 플레이리스트 만들기
#define _CRT_SECURE_NO_WARNINGS
#pragma warning(disable : 6031) //scanf의 반환값없음warning C6031을 해제하기 위해 추가.

#include<stdio.h>
#include <string.h>
#include "playList.h"
#define MAX_N 52

int main(void)
{
	int i, len, N, K;  //N: 전체 음악 개수,  K: 나만의 플레이리스트 음악 개수
	char tmp[30];
	char* selects[MAX_N] = { 0 };

	playListNode_h* pCL = creatPlayLinkedList_h();
	selectsListNode_h* sCL = createSelectsLinkedList_h();
	playListNode* p;

	//전체 음악 리스트_playList에 음악 제목을 초기화
	char* playList[MAX_N] = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L",
"M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "a", "b", "c", "d", "e",
"f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x",
"y", "z" };

	/*
	char* playList[MAX_N] = {"BTS_Butter", "aespa_Nextlevel", "AKMU_낙하",
"헤이즈_헤픈우연", "SG워너비_Timeless"};

	printf("₩n₩n전체 음악 목록 : ₩n");
	for (i = 0; (i < MAX_N) && (playList[i]); i++) {
		printf("  %s", playList[i]);
	}
	printf("₩n₩n");
	*/

// 1) 입력 받기
	//음악목록에 있는 전체 음악개수 N과
	//그중에서 나만의 플레이리스트에 넣을 음악 개수 K 입력 받기
	scanf("%d %d", &N, &K);

	//나만의 플레이리스트에 넣을 음악 제목(A~Z, a~z)을 K개 입력 받기
	for (i = 0; i < K; i++) {
		scanf("%s", tmp);		//문자열 입력받기
		len = strlen(tmp) + 1;	//문자열 길이 + 1 할당 ('₩0' 문자 포함)

		selects[i] = (char*)malloc(sizeof(char) * len);  //입력받은 문자열 만큼
메모리 동적 할당
		strcpy(selects[i], tmp);
	}
```

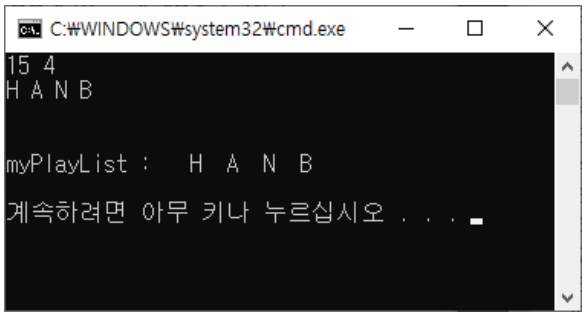
```
// 2) 전체 음악 N개에 대한 원형 연결리스트_playList 구성하기
        for (i = 0; i<N; i++) {
                insertPlayListLastNode(pCL, playList[i]);
        }

// 3) 나만의 플레이리스트_selectsList 구성하기
        for (i = 0; i < K; i++) {
                p = searchNode(pCL, selects[i]); //선택한 음악제목이 있는 playList의 노드
찾기
                insertSelectsListLastNode(sCL, p);//찾은  playList의 노드 주소를
        selectsList에 추가하기
        }

// 4) 완성된  나만의 플레이리스트_selectsList 출력하기
        printSelectsList(sCL);

        return 0;
}
```

[실행 결과]



[추가 실행 : 주석 풀고, playList를 바꾸어 실행]

playList_main.c

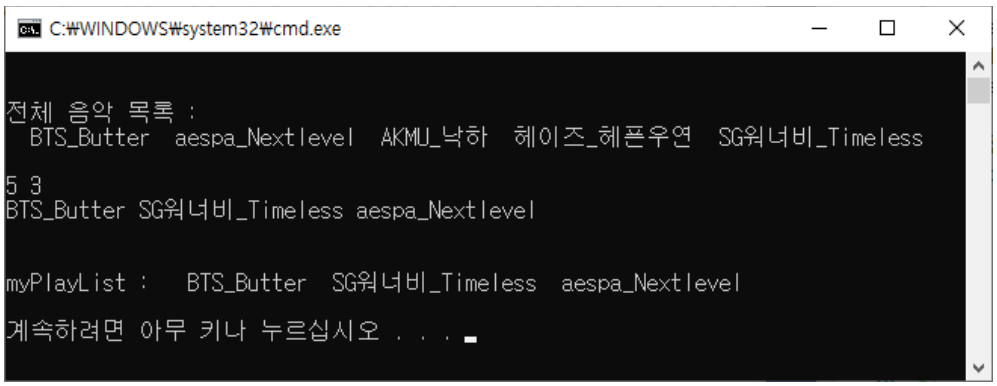
```
        //전체 음악 리스트_playList에 음악 제목을 초기화
        /*char* playList[MAX_N] = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L",
"M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "a", "b", "c", "d", "e",
"f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x",
"y", "z" };

        */
        char* playList[MAX_N] = {"BTS_Butter", "aespa_Nextlevel", "AKMU_낙하",
"헤이즈_헤픈우연", "SG워너비_Timeless"};

        printf("₩n₩n전체 음악 목록 : ₩n");
        for (i = 0; (i < MAX_N) && (playList[i]); i++) {
                printf("  %s", playList[i]);
```

```
        }
        printf("₩n₩n");
```

[추가 실행 결과]



응용예제
04

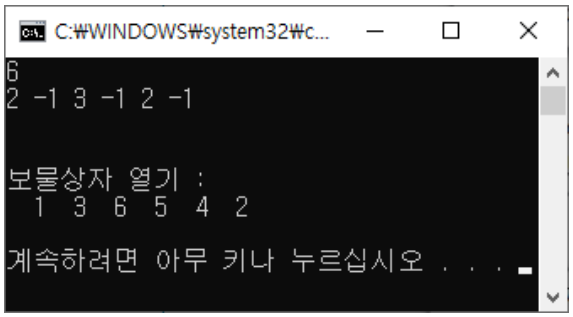
보물상자를 열어라!

입출력 예시가 잘못되었습니다.

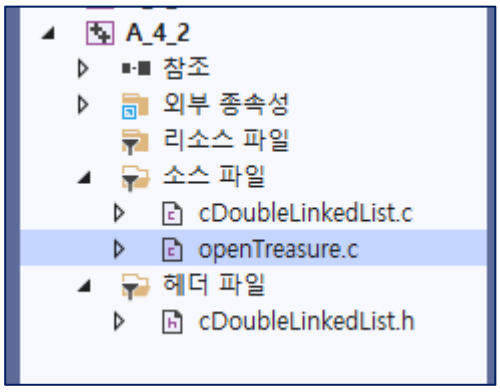
수정->

6 2 -1 3 **-1** 2 1	1 3 6 5 **4 2**

[실행 화면]



[솔루션 탐색기]



cDoubleLinkedList.h

```h
#pragma once
// 예제 4-4 이중연결리스트를 원형-이중연결리스트로 수정하여 사용

// 이중 연결 리스트의 노드 구조를 구조체로 정의
typedef struct ListNode {
        struct ListNode* llink;   // 왼쪽(선행) 노드에 대한 링크
        int number;  //데이터필드1: 현재 보물상자 번호
        int step;    //데이터필드2: 다음 이동 간격
        struct ListNode* rlink;   // 오른쪽(다음) 노드에 대한 링크
} listNode;

// 리스트 시작을 나타내는 head 노드를 구조체로 정의
typedef struct {
        listNode* head;
} linkedList_h;

linkedList_h* createLinkedList_h(void);
void printList(linkedList_h* cDL);
listNode* insertNode(linkedList_h* cDL, listNode* pre, int x, int step);
void deleteNode(linkedList_h* cDL, listNode* old);
```

cDoubleLinkedList.c

```c
// 예제 4-4 이중연결리스트를 원형-이중연결리스트로 수정하여 사용
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include "cDoubleLinkedList.h"

// 예제 4-4의 createLinkedList_h()
linkedList_h* createLinkedList_h(void) {
        linkedList_h* cDL;
        cDL = (linkedList_h*)malloc(sizeof(linkedList_h)); // 헤드 노드 할당
        cDL->head = NULL;
        // 공백 리스트이므로 NULL로 설정
        return cDL;
}

// 예제 4-4의 insertNode() 수정
listNode* insertNode(linkedList_h* cDL, listNode* pre, int x, int step){
        listNode* newNode;
        newNode = (listNode*)malloc(sizeof(listNode));
        newNode->number = x;  //데이터필드1 채우기
        newNode->step = step;   //데이터필드2 채우기

        if (cDL->head == NULL) {
                cDL->head = newNode;
                newNode->rlink = newNode; //원형으로 연결
                newNode->llink = newNode;  //원형으로 연결
        }
```

```
        else {
                newNode->rlink = pre->rlink; // 마지막 노드를 뒤에 새 노드 연결
                pre->rlink = newNode;
                newNode->llink = pre;
                cDL->head->llink = newNode;
        }
        return newNode;
}

// 예제 4-4의 deleteNode() 수정
// 이중 연결 리스트에서 old 노드를 삭제하는 연산
void deleteNode(linkedList_h* cDL, listNode* old) {
        if (cDL->head == NULL) return;    // 공백 리스트인 경우, 삭제 연산 중단
        if (cDL->head->rlink == cDL->head) { // 리스트에 노드가 한 개만 있는 경우
                free(cDL->head);   // 첫 번째 노드의 메모리를 해제하고
                cDL->head = NULL;  // 리스트 시작 포인터를 NULL로 설정
                return;
        }
        else if (old == NULL) return; // 삭제할 노드가 없는 경우, 삭제 연산 중단
        else {
                old->llink->rlink = old->rlink;
                old->rlink->llink = old->llink;
                if (old == cDL->head)
                        cDL->head = old->rlink;
                free(old);       // 삭제 노드의 메모리 해제
        }
}
```

openTreasure.c

```
//응용예제04. 보물상자를 열어라
#define _CRT_SECURE_NO_WARNINGS
#pragma warning(disable : 6031) //scanf의 반환값없음warning C6031을 해제하기 위해 추가.

#include<stdio.h>
#include "cDoubleLinkedList.h"
#define MAX_N 10

int main(void)
{
        int i, N, step, moves[MAX_N] = {0};
        linkedList_h* treasureList = createLinkedList_h();
        listNode* temp, *old;

        // 1) 입력 받기
        scanf("%d", &N);  //N: 보물상자 개수
        for (i=0; i<N; i++) //moves: 보물상자 안에 들어있는 다음 상자 위치 목록
                scanf("%d", &moves[i]);

        // 2) 보물상자를 원형 이중연결리스트로 구성
        temp = treasureList->head;
```

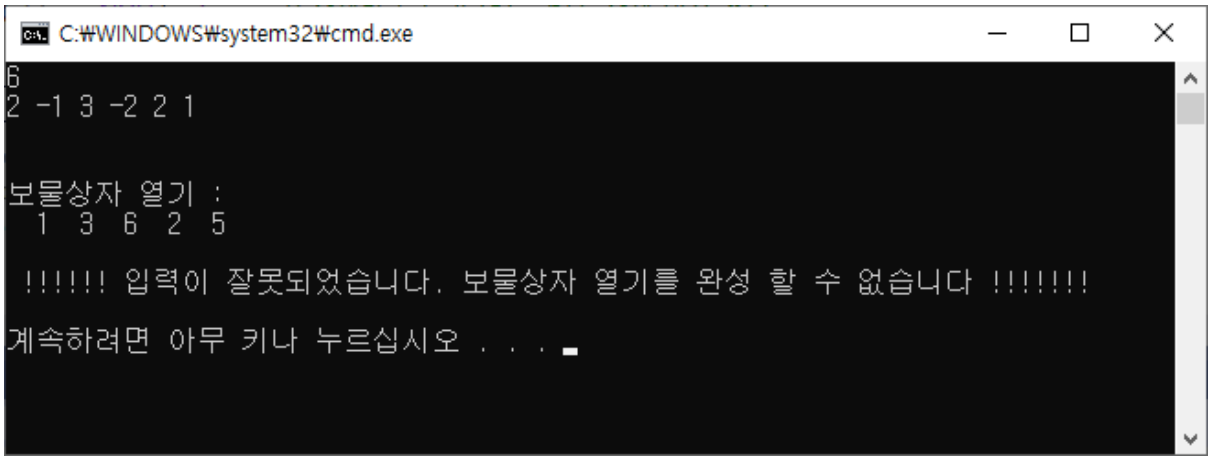
```
        for (i = 0; i < N; i++) {
                temp = insertNode(treasureList, temp, i+1, moves[i]);
        }

        // 3) 순서 찾아서 보물상자 열기
        temp = treasureList->head;
        printf("₩n₩n보물상자 열기 :₩n%3d", temp->number);  //첫번째 보물상자 번호 출력
        do {
                old = temp; //현재 보물상자를 old로 설정
                step = temp->step;
                if (step > 0) {
                        for (i = 0; i < step; i++) //양수이면, 오른쪽으로 이동
                                temp = temp->rlink;
                }
                else {
                        for (i = step; i <0 ; i++) //음수이면, 왼쪽으로 이동
                                temp = temp->llink;
                }

                if (temp == old) {
                        if (temp == treasureList->head) break;  //마지막 보물상자인 경우.
반복 종료!
                        else if ((temp->step) % 2 == 0) {
                                // 마지막 두 개의 보물상자에서 끝나지 않는 경우: moves 입력이
잘못된 경우.
                                printf("₩n₩n !!!!!! 입력이 잘못되었습니다. 보물상자 열기를
완성 할 수 없습니다 !!!!!!! ");
                                break;
                        }
                }
                else{
                        printf("%3d", temp->number);  // 이동 후의 현재 보물상자 번호 출력
                        deleteNode(treasureList, old);   // 이동 전에 열었던 보물상자 삭제
                }
        }while (1);
        printf("₩n₩n");

        return 0;
}
```

※ 책에 있는 잘못 된 원래 입력값을 사용하여 실행한 경우 :


C:₩WINDOWS₩system32₩cmd.exe
6
2 -1 3 -2 2 1
보물상자 열기 :
1 3 6 2 5
!!!!!! 입력이 잘못되었습니다. 보물상자 열기를 완성 할 수 없습니다 !!!!!!!
계속하려면 아무 키나 누르십시오 . . .