과목명	자료구조 및 알고리즘	분반	x	담당교수	김화성 교수님
학과	전자통신공학과	학번	2016707079	이름	하상천
H/W 5 - General Linear List					

1, 과제설명 (과제에 대한 설명 및 목표)

강의자료 General Linear List chapter에 기술된 Academy Award List application 문제를 참조하여 다음 설명에 따라 과제를 작성하여 제출하세요. 단, 구현은 non-Generic Coding 버전으로 작성할 것.

과제 보고서에는 프로그램의 구성(구조)에 대한 설명이 반드시 포함되어야 함.

Loop

Input a key from user

If input key is 'P', print all list of awarded pictures

If input key is 'S', search for a year

If input key is 'I', insert a node to the list //임의의 "년도, 영화제목, 감독이름"을 사용자로부터 (키보드) 입력 받아 삽입

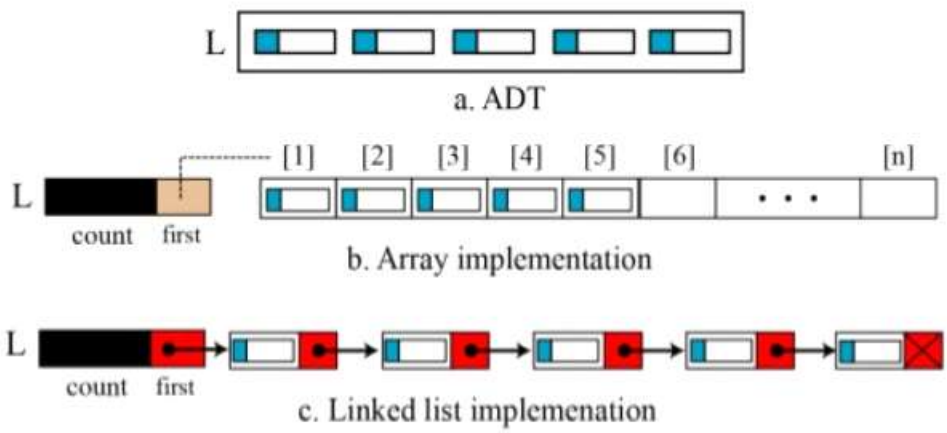
If input key is 'D', delete a node from the list // 임의의 "년도, 영화제목, 감독이름"을 사용자로부터 (키보드) 입력 받아 삭제

If input key is 'Q', quit

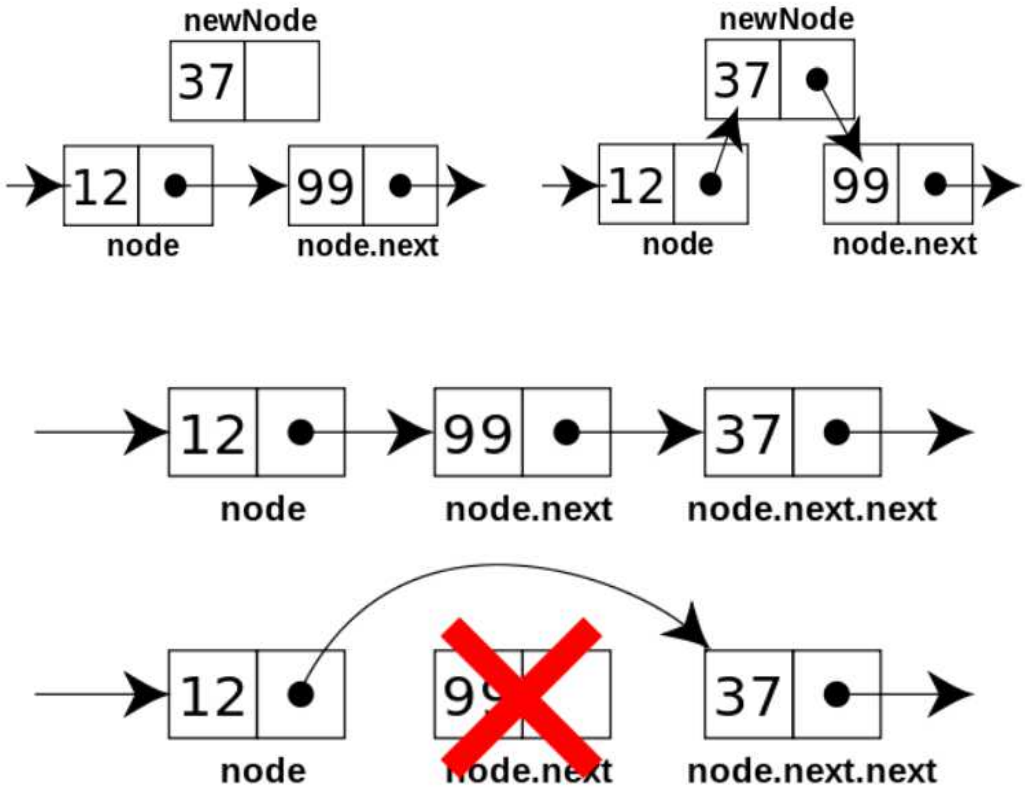
End of loop

사용자로부터 입력 받은 내용은 General Linear List의 해당 위치에 삽입함. 단 리스트는 year key로 정렬되어 있도록 설계하라.

2, 이론 (과제와 관련한 일반적인 내용)



3. 알고리즘 및 자료구조 설계 내용 기술



추가와 삭제의 경우에 첫 번째 노드인지 아닌지를 고려해서 구현해야 한다.
리스트의 중간에서 자료의 추가나 제거가 이루어질 때 현재 노드를 가리키는 pLoc과 이전 노드를 가리키는 pPre가 있으면 추가 및 삭제가 편하다.
추가하려는 노드의 위치를 찾은 후에 새 노드의 link를 다음 노드를 가리키고 이전 노드가 추가하려는 노드를 가리키게 한다.
삭제하려는 노드의 위치를 찾은 후에 이전노드가 삭제하려는 노드의 다음 노드를 가리키도록 하면 된다.

4. 소스코드 설명 (직접 작성한 소스코드중에 핵심 부분을 발췌하여 설명)

```
#define true 1 //보기 쉽게 true, false를 1, 0 으로 정의
#define false 0
typedef int boolean; //가독성을 높이기 위해 int 자료형을 boolean 자료형으로 쓰겠다.

typedef struct ListNode {
        int year;
        char picture_name[100];
        char director[30];
        struct ListNode *link;
}ListNode;  // Data node
typedef struct {
        int count;
        ListNode *head;
        ListNode *pos;
}List; // Head node
```

가독성을 높이기 위해 true를 1로 정의하고, false를 0으로 정의했다. 또한 int 자료형을 boolean 자료형으로 쓰도록 치환했고, Data node와 Head node를 정의했다.

```
boolean search_year_List(List* pList, ListNode **pPre, ListNode **pLoc, int year) {
// year로 검색을 한다.
        for (*pPre = NULL, *pLoc = pList->head; *pLoc != NULL; *pPre = *pLoc, *pLoc =
(*pLoc)->link) {
                if ((*pLoc)->year == year) {
//*pLoc이 가리키는 year와 parameter로 들어온 year가 같다면 true를 리턴.
                        return true;
                }
                else if ((*pLoc)->year > year) {
//*pLoc이 가리키는 year가 parameter로 들어온 year보다 크다면 찾고자 하는 year를 넘어간 것이
니까 더 이상 검색하는 것은 의미가 없기 때문에 break를 통해 반복문을 탈출하고 false를 리턴.
                        break;
                }
        }
        return false;
};
```

pPre와 pLoc의 주소 값을 받아야 리턴하지 않아도 값이 바뀌기 때문에, parameter로 **pPre, **pLoc를 사용했다. 초기화를 할 때 *pLoc은 Head 노드가 가리키는 노드(첫번째 노드)를 가리키게 하고, *pPre는 NULL로 초기화 해주었다. year로 검색을 하면서 *pLoc이 가리키는 year와 parameter로 들어온 year가 같다면 true를 리턴 한다. *pLoc이 가리키는 year가 parameter로 들어온 year보다 크다면 찾고자 하는 year를 넘어간 것이니까 더 이상 검색하는 것은 의미가 없기 때문에 break를 통해 반복문을 탈출하고 false를 리턴 한다.

```
boolean print_picture_name(List* pList, int fromwhere, char picture_name[]) {
	if (fromwhere == 0 || pList->pos == NULL) {
//처음 출력하는 경우에는 pList의 pos가 pList의 head가 가리키는 노드(첫 번째 노드)를 가리키게 함.
		pList->pos = pList->head;
	}
	else {
		pList->pos = pList->pos->link; //pos가 가리키고 있는 노드의 link가 가리키는 노드를 pos가 가리킴.
	}

	if (pList->pos != NULL) {
//마지막 노드가 아니면, pos가 가리키고 있는 노드의 picture name를 배열 picture name으로 복사
		strcpy(picture_name, pList->pos->picture_name);
		return true;
	}
	else {
		return false;
	}
};
```

picture name을 출력하는 함수이다. 처음 출력하는 경우에는 pList의 pos가 pList의 head가 가리키는 노드(첫 번째 노드)를 가리키게 한다. 나머지의 경우에는 pos가 가리키고 있는 노드의 link가 가리키는 노드를 pos가 가리키게 한다. 그 다음 마지막 노드가 아니면, pos가 가리키고 있는 노드의 picture name를 배열 picture name으로 복사하고 true를 리턴하고, 마지막 노드 즉 pos가 NULL을 가리키고 있다면 false를 리턴 한다.

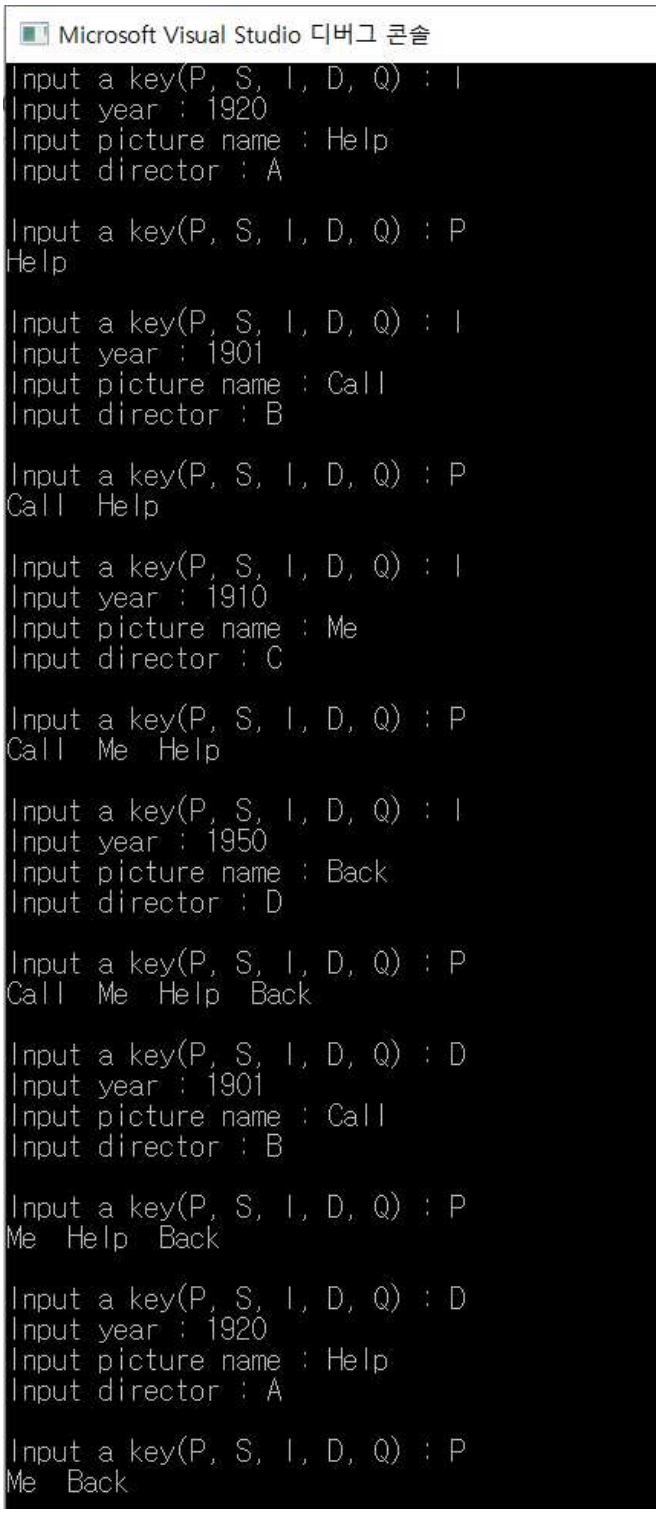
```
void clearInputBuffer()
{
	// 입력 버퍼에서 문자를 계속 꺼내고 ₩n를 꺼내면 반복을 중단
	while (getchar() != '₩n');
};
```

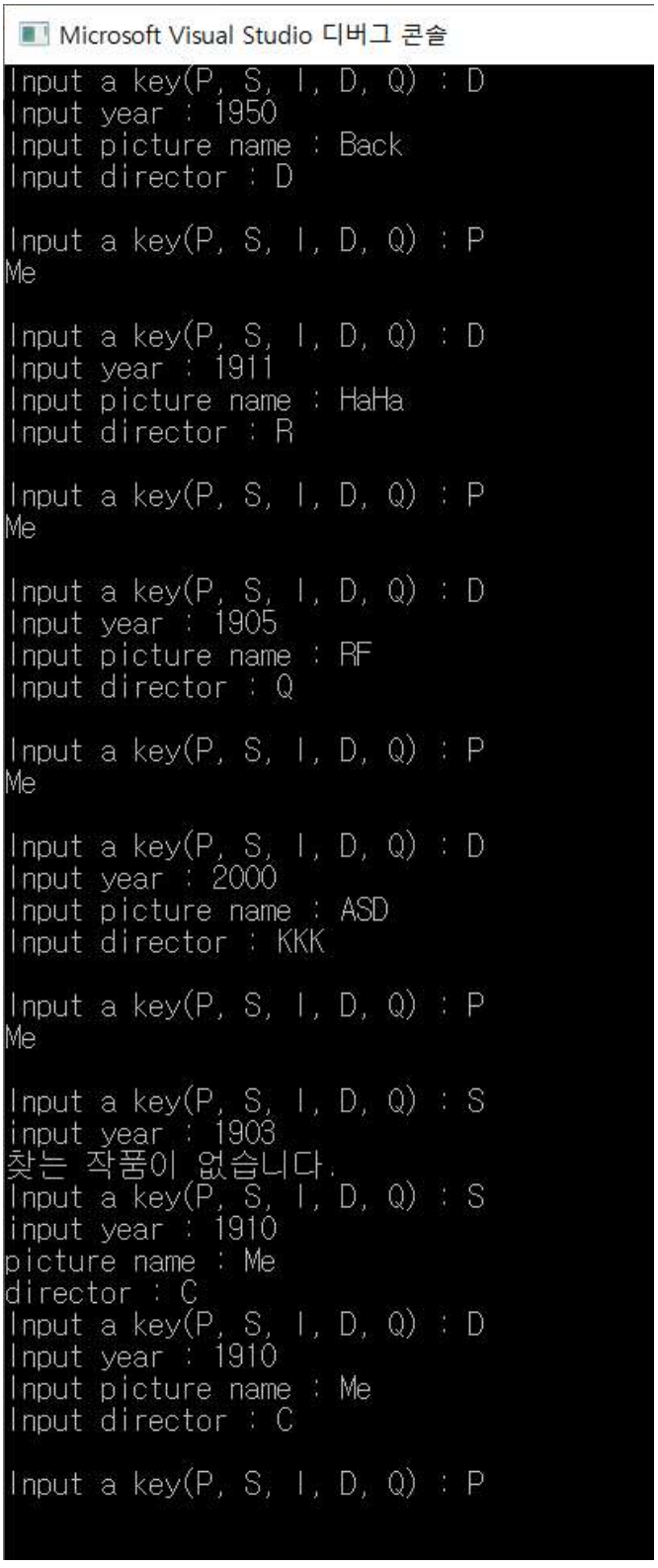
입력 버퍼에서 문자를 계속 꺼내고 ₩n를 꺼내면 반복을 중단한다. key 값에 엔터키가 들어가서 사용하였다.

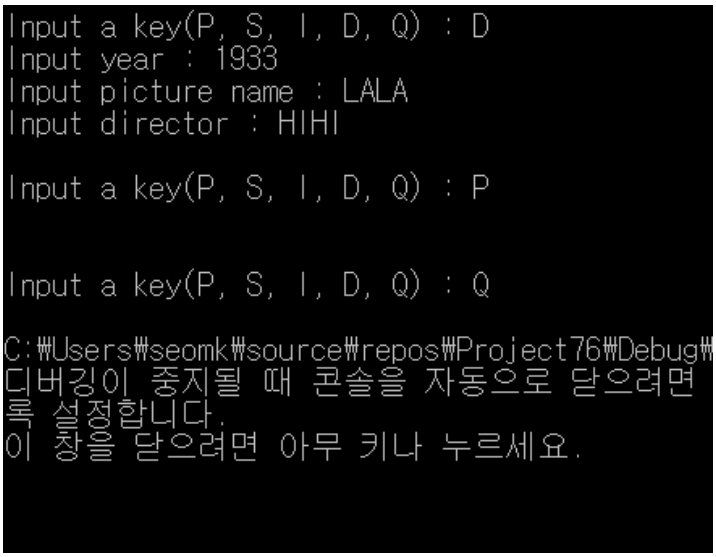
```
void addListNode(List *pList, int year, char* picture_name, char* director) {
        ListNode *pPre = NULL, *pLoc = NULL;

        if (pList->count == 0) {  //처음 Data node를 삽입하는 경우
                insert_firstList(pList, year, picture_name, director);
                return;
        }
        boolean found = search_year_List(pList, &pPre, &pLoc, year);
//pPre, pLoc의 주소값을 parameter로 사용.
        if (!found) { //search 하여 얻은 pPre를 이용해 data node 삽입.
                insertList(pList, pPre, year, picture_name, director);
        }
};
```

처음 Data Node를 삽입하는 경우에는 insert_firstList 함수를 이용한다. pPre, pLoc의 주소값을 parameter로 사용하여 똑같은 year가 있는지 검색하고, 없다면 찾은 pPre를 이용하여 data node를 삽입한다.

5. 실행결과 및 설명 (실행 결과를 캡쳐하여 첨부한 후 설명)


Microsoft Visual Studio 디버그 콘솔
Input a key(P, S, I, D, Q) : I
Input year : 1920
Input picture name : Help
Input director : A
Input a key(P, S, I, D, Q) : P
Help
Input a key(P, S, I, D, Q) : I
Input year : 1901
Input picture name : Call
Input director : B
Input a key(P, S, I, D, Q) : P
Call Help
Input a key(P, S, I, D, Q) : I
Input year : 1910
Input picture name : Me
Input director : C
Input a key(P, S, I, D, Q) : P
Call Me Help
Input a key(P, S, I, D, Q) : I
Input year : 1950
Input picture name : Back
Input director : D
Input a key(P, S, I, D, Q) : P
Call Me Help Back
Input a key(P, S, I, D, Q) : D
Input year : 1901
Input picture name : Call
Input director : B
Input a key(P, S, I, D, Q) : P
Me Help Back
Input a key(P, S, I, D, Q) : D
Input year : 1920
Input picture name : Help
Input director : A
Input a key(P, S, I, D, Q) : P
Me Back


Microsoft Visual Studio 디버그 콘솔
Input a key(P, S, I, D, Q) : D
Input year : 1950
Input picture name : Back
Input director : D
Input a key(P, S, I, D, Q) : P
Me
Input a key(P, S, I, D, Q) : D
Input year : 1911
Input picture name : HaHa
Input director : R
Input a key(P, S, I, D, Q) : P
Me
Input a key(P, S, I, D, Q) : D
Input year : 1905
Input picture name : RF
Input director : Q
Input a key(P, S, I, D, Q) : P
Me
Input a key(P, S, I, D, Q) : D
Input year : 2000
Input picture name : ASD
Input director : KKK
Input a key(P, S, I, D, Q) : P
Me
Input a key(P, S, I, D, Q) : S
input year : 1903
찾는 작품이 없습니다.
Input a key(P, S, I, D, Q) : S
input year : 1910
picture name : Me
director : C
Input a key(P, S, I, D, Q) : D
Input year : 1910
Input picture name : Me
Input director : C
Input a key(P, S, I, D, Q) : P



Insert a node into a null list
Insert a node before the first data node //기존 노드들보다 year가 가장 앞선 데이터를 입력 받아 삽입함
Insert a node between two data nodes
Insert a node after the last node

Delete to a null list //delete all of the data nodes from the list
Delete the first data node in the list
Delete the node between two data nodes
Delete the node at the end of the list

Try to delete a node that does not exist
Try to delete a node whose key is less than the first data node's key
Try to delete a node whose key is greater than the last data node's key
Try to delete from an empty list

순서대로 실행한 결과 화면입니다.

(그림을 문서에 포함, 글자처럼 취급 옵션, 잉크 절약과 잘 보이게 하기위해 그림 반전)

6. 고찰 (과제를 진행하면서 배운점 이나, 시행 착오 내용, 기타 느낀점)
 이번 과제는 General Linear List을 이해하고, Academy Award List application 문제를 c언어로 구현해 보는 것이었다. 처음에는 어떻게 접근해야 할지 몰라서 linked list를 노트에 적어가며 생각해보았다. 강의 자료도 이용하니까 어떤 식으로 코딩을 해야 할지 감이 왔다. main문에서 P를 입력하면 year로 정렬 된 picture name을 출력하는 것이고, S를 입력하면 year를 입력받아 똑같은 year이 있으면 그 작품의 정보를 출력하고, 없으면 작품이 없다고 출력한다. I를 입력하면 year, picture_name, director를 입력받아 data node에 삽입한다. D를 입력하면 year, picture_name, director를 입력받아 data node를 삭제한다. Q를 입력하면 break를 통해 반복문을 탈출하고 프로그램을 종료한다. 우선 P를 입력하였을 때는 do while문을 통해 picture name을 출력하는데, pList와 index, 작품 이름을 받을 배열이 parameter로 들어간다. Head Node가 가리키고 있는 Data node부터 작품 이름을 복사해 와서 출력한다. pos가 가리키는 것이 NULL값이면 마지막 노드이기 때문에 false를 리턴하고, do while문을 탈출한

다. S를 입력하면 year를 입력받고, pList와 year를 parameter로 print_search_year() 함수를 사용한다. pPre, pLoc의 주소값을 parameter로 하여 똑같은 year가 있는지 검색한다. 주소 값을 넣는 이유는 리턴하지 않아도 data 값이 변하기 때문이다. 만약 있다면 picture name과 director 정보를 출력하고 없으면 작품이 없다고 출력한다. I를 입력하면, count가 0일 때는 첫 번째 data node를 삽입하는 것이기 때문에 insert_firstList() 함수를 사용하고, 나머지의 경우에는 insertList() 함수를 사용한다. 우선 첫 번째 노드를 삽입 할 때는 pNewNode 하나를 동적할당 받고, parameter로 들어온 정보들을 저장해준다. 그리고 Head Node가 pNewNode를 가리키게 하고, pNewNode의 link는 가리킬 것이 없으니까 NULL 값을 가진다. 나머지의 경우에는 search_year_List() 함수를 통해 얻은 pPre를 이용하여 data node를 삽입한다. 우리가 입력한 year가 이미 있다면 삽입을 하지 않고, 우리가 입력한 year보다 작은 year를 가진 노드 중에 가장 큰 year 값을 가진 노드를 pPre가 가리킨다. pPre가 NULL이면 첫 번째 노드로 삽입하는 것이므로 Head node가 가리키던 노드를 삽입되는 노드가 가리키게 하고, Head node는 삽입된 노드를 가리킨다. 그 이외에 경우에는 pPre가 가리키는 노드가 가리키는 노드를 삽입 되는 노드가 가리키게 하고, pPre가 가리키는 노드는 새로 삽입되는 노드를 가리키게 하여 list를 연결해준다. 그리고 count를 1 증가 시킨다. D를 입력하면 search_year_List() 함수로 똑같은 year가 있는지 검색하고 있다면 deleteList()함수로 data node를 삭제한다. 이 때도 pLoc은 똑같은 year를 가진 노드를 가리키고, pPre는 그 전 노드를 가리키고 있다. deleteList() 함수에서 첫 번째 노드를 삭제하는 경우에는 pLoc이 가리키고 있는 노드(즉, 삭제하려는 노드)가 가리키던 노드를 Head node가 가리키게 하고, 그 이외에 경우에는 pLoc이 가리키고 있는 노드(즉, 삭제하려는 노드)가 가리키던 노드를 pPre가 가리키는 노드가 가리키게 하여 list를 연결해준다. 그리고 count를 1 감소 시킨다. Q를 입력하면 break를 통해 while문을 탈출하고 프로그램을 종료한다.

7. 전체 소스코드 (글자크기 9에 줄간격을 120%로 유지하고 한 줄이 너무 길지 않게 작성)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h> //strcpy 사용하기 위함.
#define true 1 //보기 쉽게 true, false를 1, 0 으로 정의
#define false 0
typedef int boolean; //가독성을 높이기 위해 int 자료형을 boolean 자료형으로 쓰겠다.

typedef struct ListNode {
        int year;
        char picture_name[100];
        char director[30];
        struct ListNode *link;
}ListNode;  // Data node
typedef struct {
        int count;
        ListNode *head;
        ListNode *pos;
}List; // Head node
List* createList() {  // Head node 생성 및 초기화
        List *pNewList = (List*)malloc(sizeof(List)); //동적 할당
        if (pNewList == NULL) {
                return NULL;
        }
```

```
        pNewList->count = 0;
        pNewList->head = NULL;
        pNewList->pos = NULL;
        return pNewList;
};
void insert_firstList(List* pList, int year, char picture_name[], char director[]) {
//처음으로 노드 생성 할 때
        ListNode* pNewNode = (ListNode*)malloc(sizeof(ListNode));

        pNewNode->year = year;
        strcpy(pNewNode->picture_name, picture_name);  //문자열 복사
        strcpy(pNewNode->director, director);  //문자열 복사

        pList->head = pNewNode; // Head 노드가 새로 만든 노드를 가리키게 한다.
        pNewNode->link = NULL;  // 새로 만든 노드의 link에 NULL 대입
        pList->count++;  //count 1증가
};
void insertList(List* pList, ListNode *pPre, int year, char picture_name[], char director[]) {
        ListNode* pNewNode = (ListNode*)malloc(sizeof(ListNode));

        pNewNode->year = year;
        strcpy(pNewNode->picture_name, picture_name); //문자열 복사
        strcpy(pNewNode->director, director); //문자열 복사



        if (pPre == NULL) {  //첫번째 노드로 들어가는 경우
                pNewNode->link = pList->head;
//새로 만든 노드는 Head 노드가 가리켰던 노드를 가리킴.
                pList->head = pNewNode;
// Head 노드는 새로 만든 노드를 가리킨다.
        }
        else {
                pNewNode->link = pPre->link;
//pPre가 가리키던 노드를 새로 만든 노드가 가리킨다.
                pPre->link = pNewNode;
// pPre는 새로 만든 노드를 가리킨다.
        }

        pList->count++;  //count 1 증가
};

boolean search_year_List(List* pList, ListNode **pPre, ListNode **pLoc, int year) {
// year로 검색을 한다.
        for (*pPre = NULL, *pLoc = pList->head; *pLoc != NULL; *pPre = *pLoc, *pLoc =
(*pLoc)->link) {
```

```
                if ((*pLoc)->year == year) {
//pLoc이 가리키는 year와 parameter로 들어온 year가 같다면 true를 리턴.
                        return true;
                }
                else if ((*pLoc)->year > year) {
//pLoc이 가리키는 year가 parameter로 들어온 year보다 크다면 찾고자 하는 year를 넘어간 것이니까 더 이상 검색하는 것은 의미가 없기 때문에 break를 통해 반복문을 탈출하고 false를 리턴.
                        break;
                }
        }
        return false;
};

void addListNode(List *pList, int year, char* picture_name, char* director) {
        ListNode *pPre = NULL, *pLoc = NULL;

        if (pList->count == 0) {  //처음 Data node를 삽입하는 경우
                insert_firstList(pList, year, picture_name, director);
                return;
        }
        boolean found = search_year_List(pList, &pPre, &pLoc, year);
//pPre, pLoc의 주소값을 parameter로 사용.
        if (!found) { //search 하여 얻은 pPre를 이용해 data node 삽입.
                insertList(pList, pPre, year, picture_name, director);
        }
};
boolean print_picture_name(List* pList, int fromwhere, char picture_name[]) {
        if (fromwhere == 0 || pList->pos == NULL) {
//처음 출력하는 경우 pList의 pos가 pList의 head가 가리키는 노드(첫 번째노드)를 가리키게 함.
                pList->pos = pList->head;
        }
        else {
                pList->pos = pList->pos->link; //pos가 가리키고 있는 노드의 link가 가리키는 노드를 pos가 가리킴.
        }

        if (pList->pos != NULL) {
//마지막 노드가 아니면, pos가 가리키고 있는 노드의 picture name를 복사
                strcpy(picture_name, pList->pos->picture_name);
                return true;
        }
        else {
                return false;
        }
};
```

```
void deleteList(List *pList, ListNode *pPre, ListNode *pLoc)
{
	if (pPre == NULL) //첫 번째 노드를 삭제하는 경우
		pList->head = pLoc->link;
	else  // 첫 번째 이외에 노드를 삭제하는 경우
		pPre->link = pLoc->link;
	free(pLoc);  //동적 메모리 반납
	pList->count--;  //count 1 감소.
};

void removeList(List *pList, int year)
{
	ListNode *pPre = NULL, *pLoc = NULL;
	boolean found = search_year_List(pList, &pPre, &pLoc, year);
//pPre, pLoc의 주소값을 parameter로 대입하여 search 한다.
	if (found) { //삭제할 노드가 있다면
		deleteList(pList, pPre, pLoc);
	}
};
void print_search_year(List *pList, int year)
{  //year로 검색하여 작품이 있다면 정보 출력, 없다면 작품이 없다고 출력.
	ListNode *pPre = NULL, *pLoc = NULL;
	boolean found = search_year_List(pList, &pPre, &pLoc, year);
//pPre, pLoc의 주소값을 parameter로 대입하여 search 한다.
	if (found) {
		printf("picture name : %s₩n", pLoc->picture_name);
		printf("director : %s", pLoc->director);
	}
	else {
		printf("찾는 작품이 없습니다.");
	}
}

void destroyList(List* pList) {
	ListNode *pDel = NULL, *pNext = NULL;
	for (pDel = pList->head; pDel != NULL; pDel = pNext) {
// 하나하나씩 꼬리 물며 동적 메모리 반납.
		pNext = pDel->link;
		free(pDel);
	}
	free(pList); // 마지막으로 Head node 동적 메모리 반납
};
void clearInputBuffer()
{
	// 입력 버퍼에서 문자를 계속 꺼내고 ₩n를 꺼내면 반복을 중단
```

```
        while (getchar() != '₩n');
};
void main() {
        List* pList = createList();
        char key;
        int year;
        char picture_name[100];
        char director[30];

        while (true) {
                printf("Input a key(P, S, I, D, Q) : ");
                scanf("%c", &key);  //key 값 입력받기

                if (key == 'P') {
                        int n = 0;
                        boolean check;
                        do {
                                check = print_picture_name(pList, n++, picture_name);
                                if (check) {
                                        printf("%s  ", picture_name);
                                }
                        } while (check);
                        printf("₩n");

                }
                else if (key == 'S') {
                        printf("input year : ");
                        scanf("%d", &year);
                        print_search_year(pList, year);
//입력 받은 year로 검색하여 작품 출력
                }
                else if (key == 'I') {
                        printf("Input year : ");
                        scanf("%d", &year);
                        printf("Input picture name : ");
                        scanf("%s", picture_name);
                        printf("Input director : ");
                        scanf("%s", director);
                        addListNode(pList, year, picture_name, director);
//입력 받은 year, picture name, director data node에 삽입.
                }
                else if (key == 'D') {
                        printf("Input year : ");
                        scanf("%d", &year);
                        printf("Input picture name : ");
```

```
                        scanf("%s", picture_name);
                        printf("Input director : ");
                        scanf("%s", director);
                        removeList(pList, year);
//year를 통해 data node 삭제
                }
                else if (key == 'Q') {
                        break; //break를 통해 반복문 탈출.
                }
                clearInputBuffer(); //입력 버퍼 비우기, key에 엔터키가 들어가서 사용함.
                printf("₩n");
        }
        destroyList(pList); //동적 메모리 반납.
}
```

(글자크기는 10으로 유지하고 줄간격도 160%를 유지할 것)