C프로그래밍 및 실습

플레이리스트 추천

최종 보고서

제출일자: 2023.12.24
제출자명: 김도형
제출자학번: 231630

1. 프로젝트 목표

1) 배경 및 필요성

음악 플랫폼 수가 급격히 증가하면서, 음악 플랫폼 간의 경쟁이 치열 해지고 있다. 최근 이용자들은 자신의 취향에 맞는 노래를 자동 재생하여 주는 것을 선호하기 때문에, 여러 음악재생 플랫폼들 사이에서 살아남기 위해서는 사용자들을 구체적으로 분석하고 파악하여 이용자들에게 맞는 노래는 추천해주어야 한다고 생각하였다.

2) 프로젝트 목표

이용자들의 선호하는 장르를 파악하여 이용자에게 최적의 노래를 추천해 주는 것을 목표로 한다.

3) 차별점

기존에 존재하는 음악 플랫폼들은 인기차트나 새로운 곡을 소개하는 형식으로
되어있지만, 이 프로그램을 통하여 취향이 비슷한 사람의 노래를 추천받고, 선호하는
장르가 비슷한 사람들의 플레이리스트를 공유 받을 수 있다.

2. 기능 계획

1) 기능 1: 재생목록 삭제 및 추가

가장 기본적인 음악 플랫폼의 기능. 사용자들에게 추가할 노래, 삭제할 노래, 수정할 노래, 노래의 장르를 입력 받고 저장 및 출력.
(적용된 내용: 반복문, 함수, 2 차원배열, 조건문, 2 차원 배열, 포인터, 구조체, 헤더파일)

2) 기능 2: 각자의 재생목록 관리 + 선호 장르 노래 추천

이용자 5 명에게 이름, 선호 장르를 입력 받고 각자의 플레이리스트를 생성, 수정, 삭제, 출력, 종료 기능을 관리
(적용된 내용: 다차원 배열, 반복문, 조건문, 함수, 포인터, 헤더파일, 구조체)

3) 기능 3: 가장 인기있는 장르 출력 + 같은 장르의 노래 출력

이용자들에게 장르를 입력 받을 때 겹치거나, 가장 많이 나온 장르를 인기있는
노래로 인식하고, 가장 인기있는 장르로 출력
(적용된 내용: 반복문, 조건문, 함수, 배열, 포인터, 구조체, 헤더파일)

4) 기능 4: 고객센터 + 노래 중복 제거

이용자들에게 메뉴얼을 입력 받아, 고객센터에서 쿠폰을 등록하거나, 사용설명서의 출력, 이용권을 해지할 수 있는 기능. 겹치는 노래를 플레이리스트에서 제거하는 기능
(적용된 내용: str 함수, 배열, 반복문, 조건문, 함수, 헤더파일, 구조체)

3. 기능구현

(1) 재생목록에 노래/장르 추가, 삭제, 수정, 종료

- 입출력: 사용자 이름 입력, 메뉴 선택 인덱스, 사용자 선택 인덱스, 노래 장르, 노래 제목, 추가할 노래, 삭제할 노래
- 설명: 사용자 5 명에게 이름을 입력 받고 저장한 후, 사용자의 번호를 입력하여, 그 번호에 해당하는 플레이리스트를 관리하는 기능.

#define 을 이용하여 최대 노래 개수, 노래 제목 길이, 장르 이름 길이, 장르 개수, 인원수를 정의하였고 구조체 typedef struct(song, person)를 헤더파일에 선언하여 장르와 노래 이름, 노래 개수, 선호 장르를 이루는 배열들을 헤더파일에서 구조체로 관리하도록 구성하였다.
(적용된 배운 내용: 구조체, 헤더파일, 선언, 함수, 포인터, 배열)

(play_list_project_main.h (헤더파일))

```c
//231630 김도형 플레이리스트 프로젝트 헤더파일

#ifndef PLAY_LIST_PROJECT_MAIN_H
#define PLAY_LIST_PROJECT_MAIN_H

#define MAX_SONGS 10
#define SONG_LEN 100
#define GENRE_NAME 100
#define GENRE_NUM 3
#define PERSON_COUNT 5

typedef struct {
    char title[SONG_LEN];
    char genre[GENRE_NAME];
} Song;

typedef struct {
    char name[50];
    Song favoriteSongs[MAX_SONGS];
    int songCount;
    char favoriteGenre[GENRE_NAME];
} Person;

extern Person people[PERSON_COUNT];
extern char genre[GENRE_NUM][GENRE_NAME];
extern int coupon_answer;

void addSong(int personIndex);
void delSong(int personIndex);
void printSongList(int personIndex);
void recommendSong(char* inputGenre);
void printPopularGenre();
void customerCenter();

#endif
```

Add song 함수에서는 정수형의 personIndex 입력 받아와서, 입력 받은 personIndex 번째 사람의 플레이리스트에 노래를 추가하는 역할을 한다. 추가하려는 노래가 이미 재생목록에 존재할 때 strcmp(문자열을 비교하는 함수)를 통해 이미 재생목록에 있다는 문구를 표시하고 재생목록에 추가하지 않는다. 또한 노래의 장르를 같이 입력 받아서, 입력한 노래가 입력한 장르에 저장되도록 작성하였다.

(main.c) <addsong>

```
        switch (choice) {
        case 1:
            printf("사용자를 선택하세요 (1~5): ");
            int personIndex;
            scanf_s("%d", &personIndex);

            if (personIndex <= 0 || personIndex > PERSON_COUNT) {
                printf("잘못된 선택입니다.₩n");
                break;
            }

            addSong(personIndex - 1);
            break;
```

(play_list_project_main.c) <addsong>

```
void addSong(int personIndex) {
    printf("재생 목록에 추가할 노래를 입력하세요: ");
    scanf_s("%s", people[personIndex].favoriteSongs[people[personIndex].songCount].title, SON

    for (int i = 0; i < people[personIndex].songCount; i++) {
        if (strcmp(people[personIndex].favoriteSongs[i].title, people[personIndex].favoriteSo
            printf("이미 재생목록에 있는 노래입니다. 다시 입력해주세요.₩n");
            return;
        }
    }
```

Delsong 함수에서는 main 함수에서 personIndex 를 넘겨받고, 삭제할 노래의 인덱스를 입력하기 위해 delIndex 변수를 선언한다. if 조건문에 따라서 노래의 인덱스를 벗어나면 삭제할 노래가 없다는 문구를 출력하고, 삭제할 노래의 인덱스가 노래의 인덱스 안에 있다면, 노래의 개수를 하나 줄이고 재생목록에서 노래를 삭제한다.

(main.c) <delsong>

```
        case 2:
            printf("사용자를 선택하세요 (1~5): ");
            scanf_s("%d", &personIndex);

            if (personIndex <= 0 || personIndex > PERSON_COUNT) {
                printf("잘못된 선택입니다.₩n");
                break;
            }

            delSong(personIndex - 1);
            break;
```

(play_list_project_main.c) <delsong>

```c
void delSong(int personIndex) {
    printf("삭제할 노래의 인덱스를 입력하세요 : ");
    int delIndex;
    scanf_s("%d", &delIndex);

    if (delIndex > people[personIndex].songCount || delIndex <= 0) {
        printf("삭제할 노래가 없습니다.₩n");
    }
    else {
        printf("%d. %s : 노래를 재생목록에서 삭제합니다.₩n", delIndex, people[personIndex].favoriteSongs[delInc

        for (int i = delIndex; i < people[personIndex].songCount; i++) {
            people[personIndex].favoriteSongs[i - 1] = people[personIndex].favoriteSongs[i];
        }

        people[personIndex].songCount--;
    }
}
```

Printsonglist 함수에서는 main 함수에서 personIndex 를 받아와서 입력 받은 사용자의 재생목록을 출력한다. 재생목록을 출력할 때는 for 반복문을 통해 노래 수 만큼 반복하여 장르와 노래 이름, 노래 수를 출력한다.

(main.c) <printsonglist>

```c
        case 3:
            printf("사용자를 선택하세요 (1~5): ");
            scanf_s("%d", &personIndex);

            if (personIndex <= 0 || personIndex > PERSON_COUNT) {
                printf("잘못된 선택입니다.₩n");
                break;
            }

            printSongList(personIndex - 1);
            break;
```

(play_list_project_main.c) <printsonglist>

```c
void printSongList(int personIndex) {
    printf("%s의 재생목록 (노래 수 : %d) :₩n", people[personIndex].name, people[personIndex].songCount);

    for (int i = 0; i < people[personIndex].songCount; i++) {
        printf("%d. %s (장르: %s)₩n", i + 1, people[personIndex].favoriteSongs[i].title, people[personIndex].fa
    }

    printf("₩n");
}
```

- 적용된 배운 내용 : 함수, 포인터, 배열, 구조체, 반복문, 조건문, str 함수

(2) 각자의 플레이리스트 생성, 관리, 선호 장르 곡 추천

- 입출력: 사용자 이름, 사용자 인덱스, 사용자 별 플레이리스트(노래와 장르), 선호 장르와 그에 따른 노래들 목록

- 설명: song, person 구조체를 통해 장르의 이름과 개수, 그리고 선호하는 장르와 노래를 입력 받을 변수를 선언하였다.

Recommendsong 에서는 입력 받은 장르가 같은 경우에 그 장르의 노래들을 추천해주도록 작성하였다. For 를 이용한 이중반복문을 통해 입력한 inputgenre 에 해당하는 노래들을 출력한다.

(main.c) <recommendsong>

```
case 6:
    printf("선호하는 장르를 입력하세요 (POP, 발라드, 힙합): ");
    char inputGenre[GENRE_NAME];
    scanf_s("%s", inputGenre, GENRE_NAME);
    recommendSong(inputGenre);
    break;
```

(play_list_project_main.c)<recommendsong>

```
void recommendSong(char* inputGenre) {
    printf("₩n'%s'장르의 노래를 추천합니다:₩n", inputGenre);

    for (int i = 0; i < PERSON_COUNT; i++) {
        if (strcmp(inputGenre, people[i].favoriteGenre) == 0) {
            for (int j = 0; j < people[i].songCount; j++) {
                printf("'%s'장르 추천 노래 for %s: %s (장르: %s)₩n",
                    inputGenre, people[i].name, people[i].favoriteSongs[j].title, inputGenre);
            }
        }
    }
}
```

재생목록 노래 수정에서는 수정할 사용자의 인덱스를 입력한 수 수정할 노래의 인덱스를 입력 받는다. 수정할 노래의 번호를 입력 받기 위한 modifyindex 변수를 선언하고, 노래를 입력 받아 해당 사용자의 재생목록에서 선택한 노래의 인덱스에 저장한다.

(main.c) <modify>

```
case 5:
    printf("사용자를 선택하세요 (1~5): ");
    scanf_s("%d", &personIndex);

    if (personIndex <= 0 || personIndex > PERSON_COUNT) {
        printf("잘못된 선택입니다.₩n");
        break;
    }

    printf("수정할 노래의 번호를 입력해주세요. (1부터 시작): ");
    int modifyIndex;
    scanf_s("%d", &modifyIndex);

    printf("새롭게 추가할 노래를 입력해주세요: ");
    scanf_s("%s", people[personIndex - 1].favoriteSongs[modifyIndex - 1].title, SONG_LEN);

    printf("새로운 노래가 추가되었습니다: %d. %s (장르: %s)₩n", modifyIndex,
        people[personIndex - 1].favoriteSongs[modifyIndex - 1].title,
        genre[modifyIndex - 1]);
    break;
```

(3) 가장 인기있는 장르 출력

-입출력: 메뉴 7, 가장 인기있는 장르, 선호 장르, 같은 장르 노래
-설명: switch 문에서 7 을 입력했을 때 실행되며, main.c 에서는 함수를 호출하고, play_list_project_main.c 에서 함수를 정의한다. 사용자들에게 노래와 장르를 입력 받을 때 어떤 장르가 얼마나 나왔는지 세기 위한 genreCount 변수를 설정하고, 노래를 아무것도 입력 받지 않았을 때 7 을 입력하면 가장 인기 있는 장르가 출력되지 않아야 하기 때문에 totalsongs 라는 변수를 선언하여 totalsongs 가 0 일때는 메뉴 1 을 입력하여 노래를 입력하라는 문구를 출력한다. for 반복문을 통해 장르가 나올 때 마다 genreCount 를 1 씩 늘려서 가장 높은 장르를 최종적으로 출력한다.

(play_list_project_main.c)

```
void printPopularGenre() {
    int genreCount[GENRE_NUM] = { 0 };
    int totalSongs = 0;

    for (int i = 0; i < PERSON_COUNT; i++) {
        if (people[i].songCount > 0) {
            for (int j = 0; j < GENRE_NUM; j++) {
                if (strcmp(people[i].favoriteGenre, genre[j]) == 0) {
                    genreCount[j]++;
                }
            }
            totalSongs += people[i].songCount;
        }
    }

    if (totalSongs == 0) {
        printf("노래를 입력하지 않았습니다. 메뉴1을 선택해 노래를 입력해주세요.₩n");
    }
    else {
        int popularIndex = 0;

        for (int i = 1; i < GENRE_NUM; i++) {
            if (genreCount[i] > genreCount[popularIndex]) {
                popularIndex = i;
            }
        }

        printf("가장 인기있는 장르: %s₩n", genre[popularIndex]);
    }
}
```

(main.c)

```
        case 7:
            printPopularGenre();
            break;
```

-적용된 배운 내용: 함수, if 조건문, switch 조건문, 반복문

(4-1)고객센터

-입출력: 메뉴 8, 도움 매뉴얼, 고객센터 매뉴얼, 쿠폰번호, 설명서
-설명: switch 문에서 8 을 입력 받았을 때 실행되며, customerCenter 함수로 기능을 구분하였다. Main.c 에서는 함수를 호출하고, play_list_project_main.c 에서 함수를 정의하였다. 도움이 필요한지 아닌지 묻기 위해 select 라는 변수를 선언하고, 1 을 입력한

경우 고객센터로 들어가고 아닌 경우는 나가서 메뉴를 실행한다. 고객센터는 4 개의 메뉴로 이루어져 있는데, 이를 선택 받기 위해 center_select 라는 변수를 선언하였다. 1 을 입력하면 이용권 해지를 도와주고, 2 를 입력하였을 때는 정해진 쿠폰번호를 입력하면 쿠폰을 지급하고 아닌 경우에는 틀렸다는 문구를 출력한다. 3 을 입력하면 플레이리스트 사용설명서를 출력한다.

(main.c) <고객센터>

```
case 8:
    customerCenter();
    break;
```

(play_list_project_main.c) <고객센터>

```
void customerCenter() {
    printf("-----------------------------------------------------₩n");
    printf("<고객센터>.₩n₩n도움이 필요하시면 1, 필요 없다면 2를 입력해주세요 :₩n ");
    printf("-----------------------------------------------------₩n");

    int select;
    scanf_s("%d", &select);

    if (select == 2) {
        printf("고객센터를 종료합니다₩n");
    }
    else if (select == 1) {
        printf("-----------------------------------------------------₩n");
        printf("<고객센터 메뉴>₩n₩n1.해지 신청₩n2.쿠폰 등록₩n3.플레이리스트 사용설명서₩n");
        printf("-----------------------------------------------------₩n");

        int center_select;
        printf("고객센터 메뉴를 선택해주세요.");
        scanf_s("%d", &center_select);

        if (center_select == 1) {
            printf("이용권을 해지합니다.₩n");
        }
        else if (center_select == 2) {
            printf("쿠폰을 등록합니다. 쿠폰 번호를 입력해주세요 : ₩n");

            int coupon;
            scanf_s("%d", &coupon);

            if (coupon == coupon_answer) {
                printf("쿠폰이 확인 되었습니다. 한 달 무료 쿠폰이 증정되었습니다.₩n");
            }
            else if (coupon != coupon_answer) {
                printf("쿠폰 번호가 틀렸습니다.₩n");
            }
        }
        else if (center_select == 3) {
            printf("<플레이리스트 사용 방법>₩n");
            printf("사용자 이름 입력 ₩n");
            printf("메뉴 1 : 노래 추가 - 사용자를 선택하고, 입력할 노래의 장르와 제목을 입력.₩n");
```

-적용된 배운 내용 : 함수, if 문

(4-2)노래 중복제거

-입출력: 재생목록에 추가할 노래, 이미 존재한다는 문구

-설명: for 반복문을 통해 입력 받은 사용자 인덱스의 노래 개수만큼 돌리면서 strcmp(문자열 비교 함수)를 통해 두 문자열이 동일하여 0 을 반환한다면, 이미 재생목록에 존재한다는 문구를 출력하고 return 을 통해 종료하고 빠져나온다.

(play_list_project_main.c) <노래 중복제거>

```
void addSong(int personIndex) {
    printf("재생 목록에 추가할 노래를 입력하세요: ");
    scanf_s("%s", people[personIndex].favoriteSongs[people[personIndex].songCount].title, SONG_LEN);

    for (int i = 0; i < people[personIndex].songCount; i++) {
        if (strcmp(people[personIndex].favoriteSongs[i].title, people[personIndex].favoriteSongs[people[personIn
            printf("이미 재생목록에 있는 노래입니다. 다시 입력해주세요.₩n");
            return;
        }
    }
```

```
gs[people[personIndex].songCount].title, SONG_LEN);

Count; i++) {
ngs[i].title, people[personIndex].favoriteSongs[people[personIndex].songCount].title) == 0) {
다. 다시 입력해주세요.₩n");
```

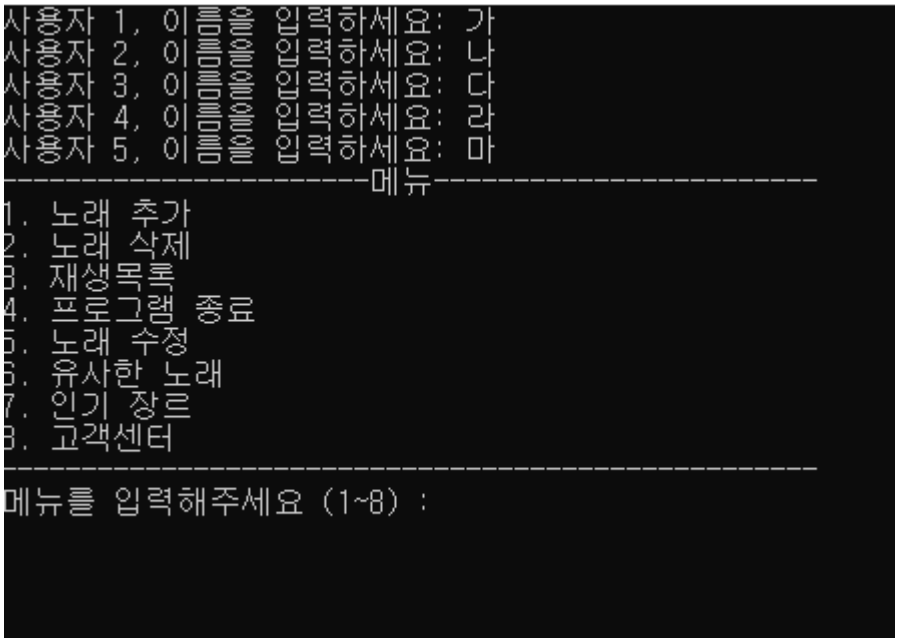
-적용된 배운 내용: 함수, 배열, 구조체, str 함수, 반복문

4. 테스트 결과

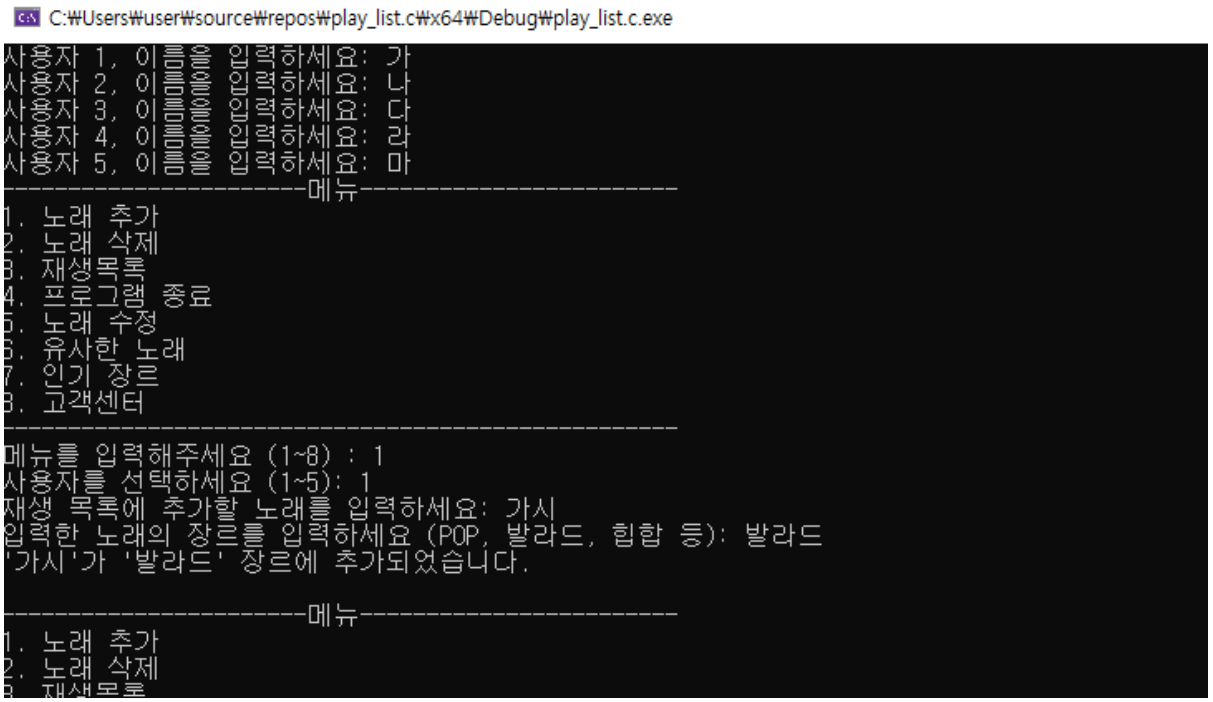
(1) 재생목록 삭제 및 추가 + 재생목록 관리 + 선호 장르 노래 (기능1+기능2)

- 설명: 사용자 5 명에게 이름을 입력 받고 저장한 후, 사용자의 번호를 입력하여, 그 번호에 해당하는 플레이리스트를 관리하는 기능.

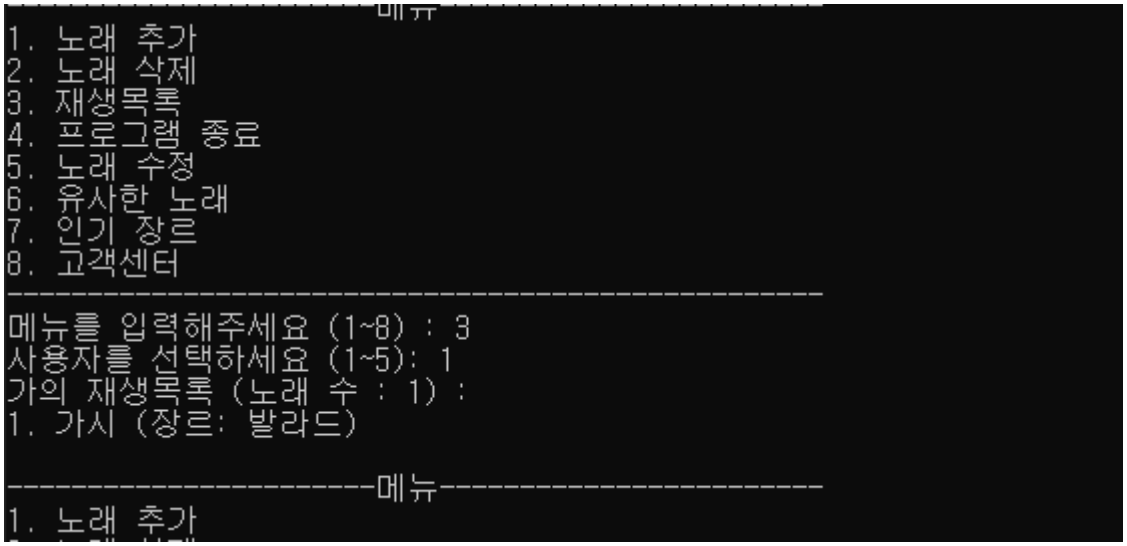
- 테스트 결과 스크린샷



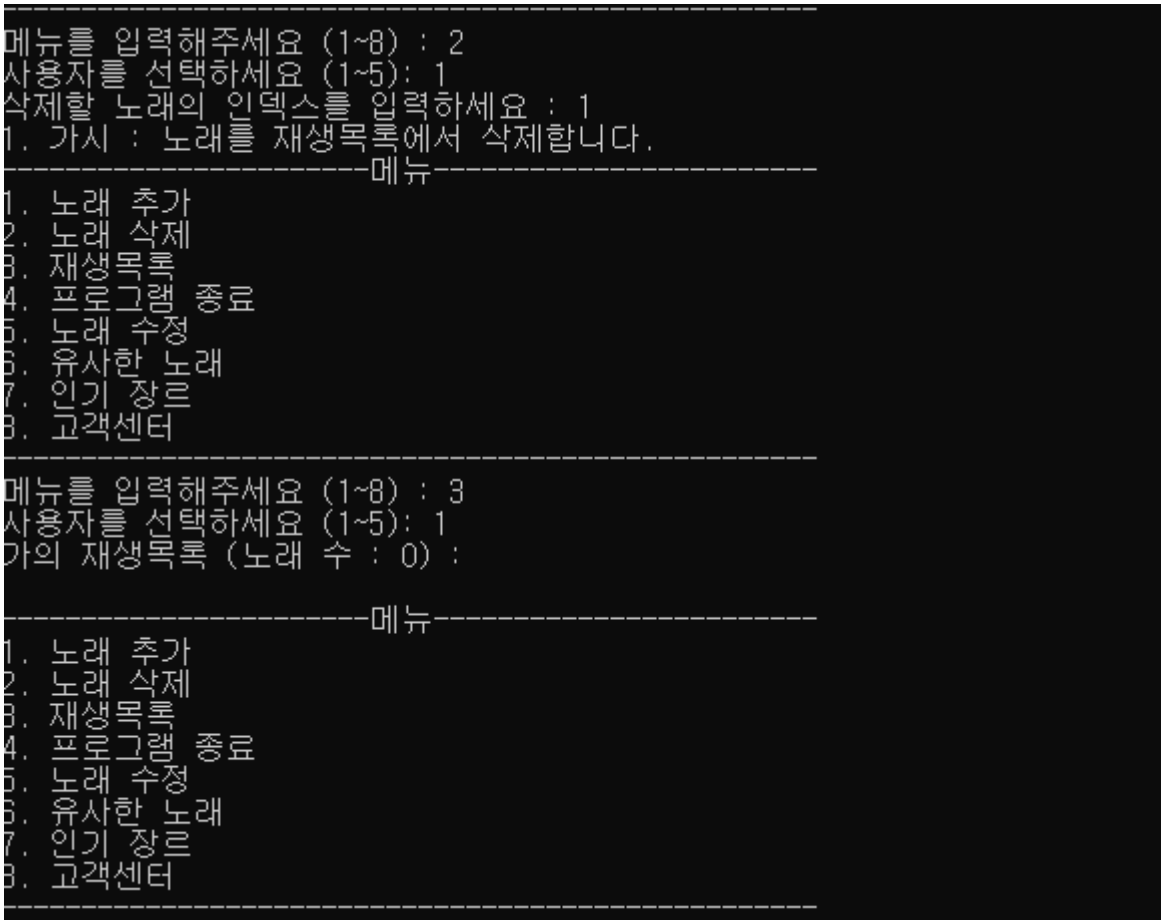
사용자에게 5명의 이름을 입력 받음.



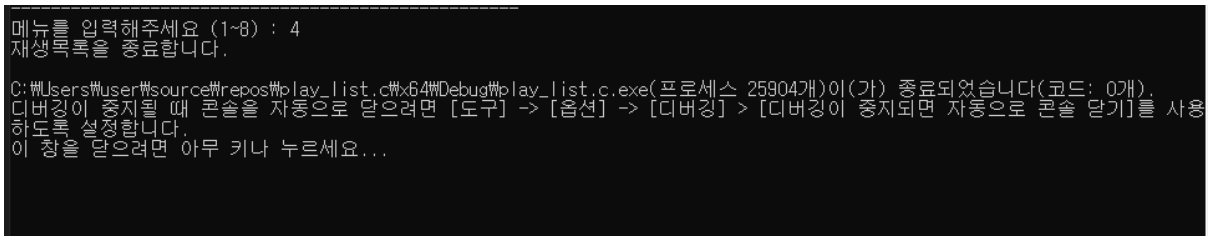
사용자 1의 재생목록에 노래를 추가하기 위해 1 >1을 입력하고, 발라드 장르의 '가시'를 추가함.



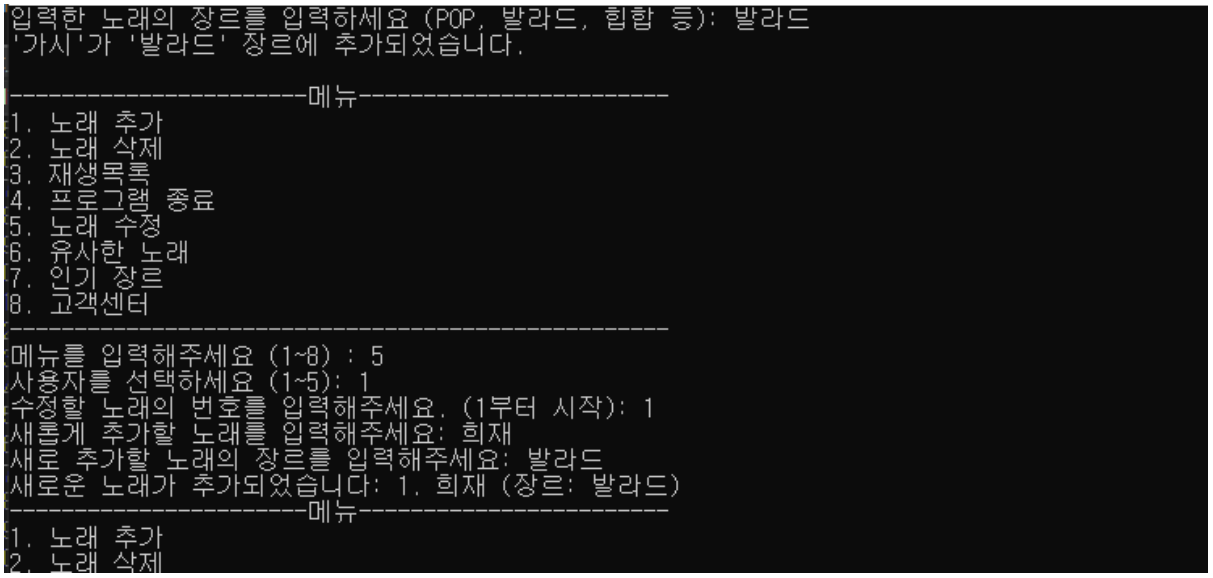
노래가 추가되었는지 확인하기 위해 3 >1을 입력하여 사용자1의 재생목록을 확인하고 발라드 장르의 '가시'노래가 추가된 것을 확인함.



2번 메뉴 노래 삭제가 작동하는지 확인하기 위해 2>1 >1을 입력해 1번사용자의 첫 번째 노래를 삭제하고, 3>1을 눌러 1번사용자의 재생목록을 확인. 추가했던 '가시' 노래가 없어지고 노래수가 0 이 된 것을 확인. 삭제 기능 잘 작동.



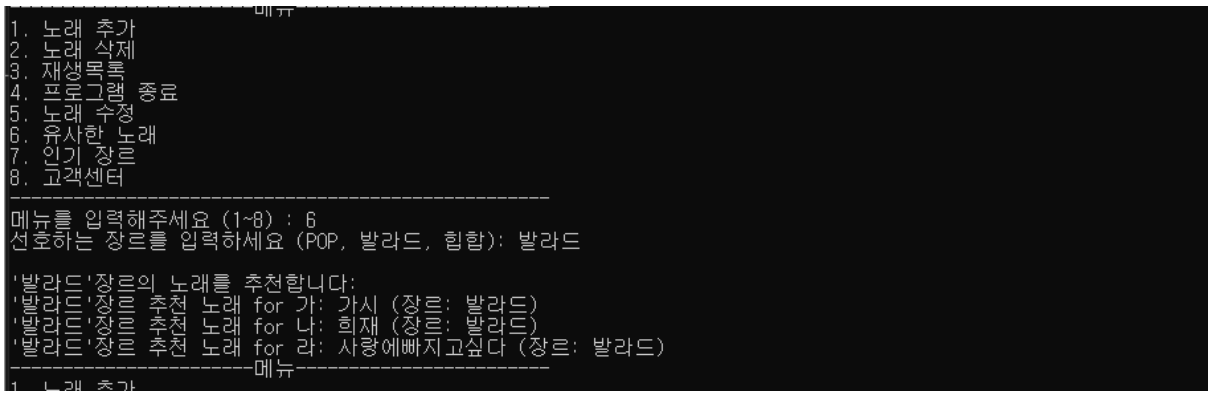
종료 메뉴 4를 입력했을 때, 프로그램이 종료되는 것을 확인.



메뉴 5가 잘 실행되는지 확인하기 위해, 1>1을 입력하여 발라드 : 가시를 추가한 후

5>1>희재>발라드 순으로 입력하여 발라드 : 가시를 >발라드:희재로 변경

올바르게 수정되었는지 확인하기 위해 3>1을 입력하여 가시가 희재로 바뀐 것을 확인.

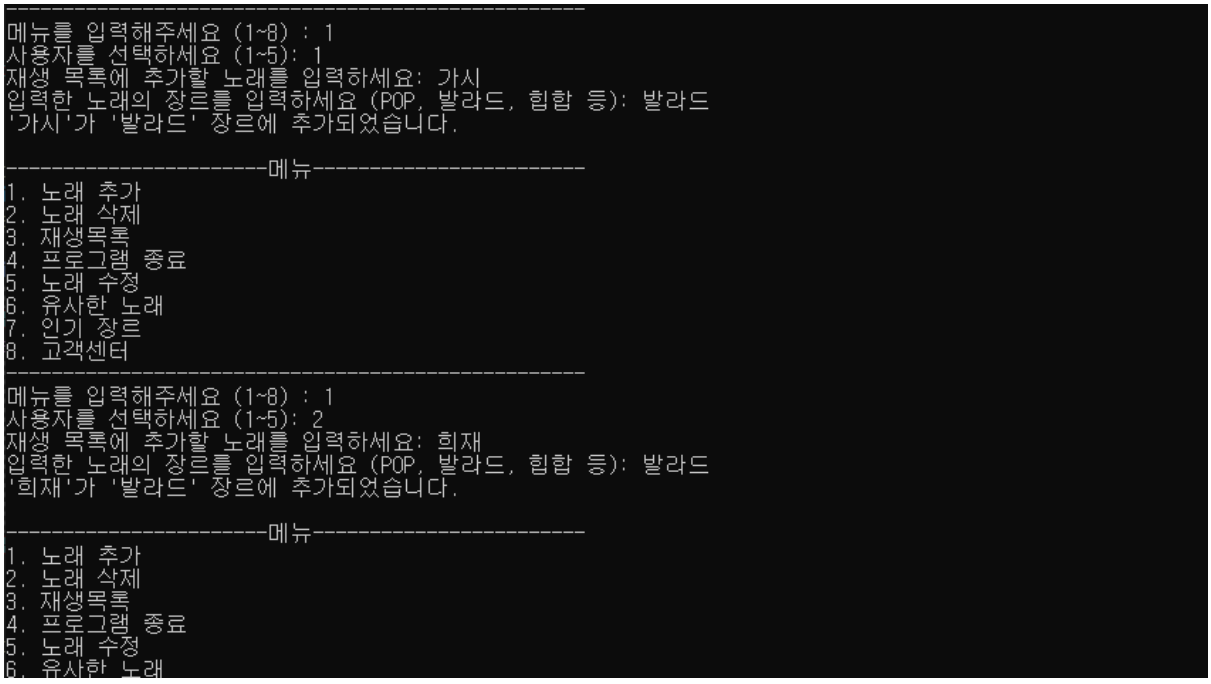


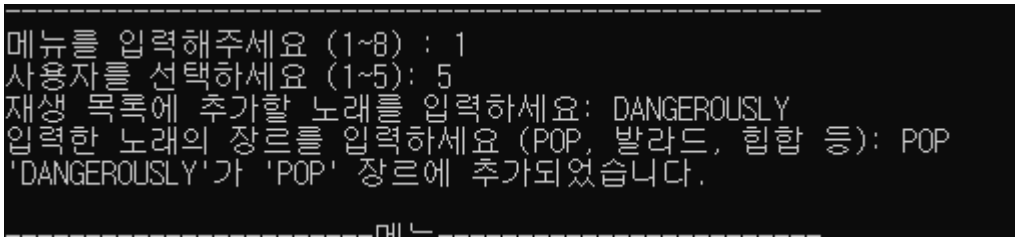
유사한 노래를 출력하기 위해, 메뉴 6을 입력하고, 선호하는 장르를 입력했을 때, 입력한 장르와 같은 장르의 노래들을 모두 출력해주는 것 확인.

(3) 가장 인기있는 장르 출력하기

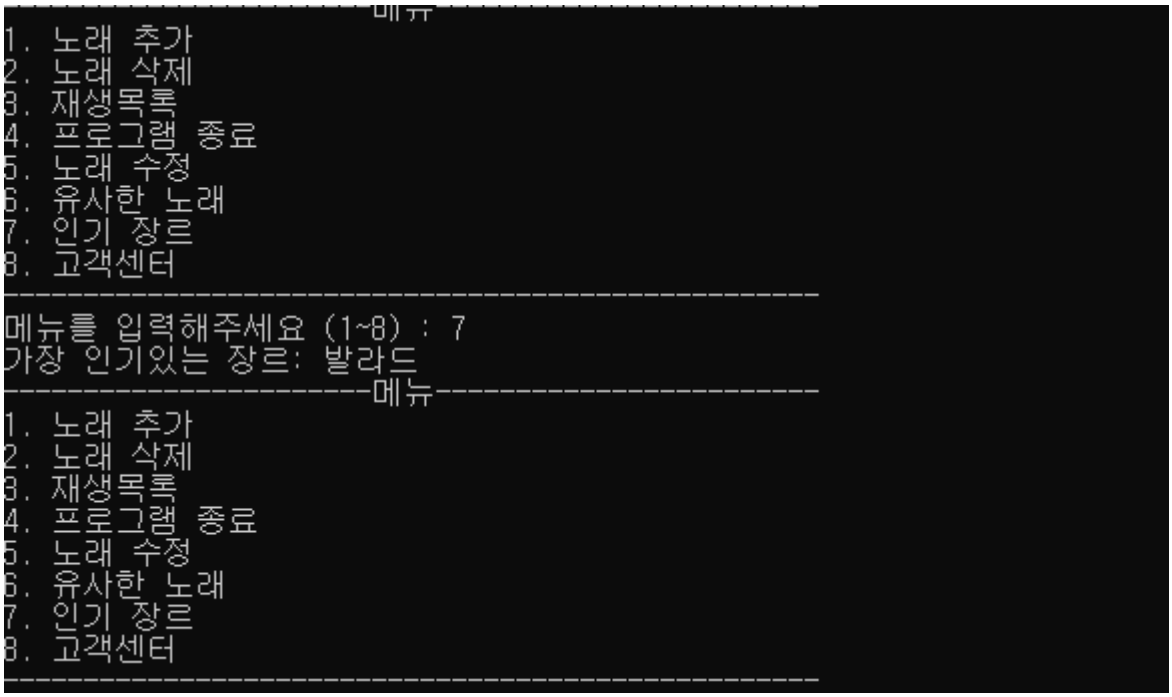
- 설명: 이용자들에게 장르를 입력 받을 때 겹치거나, 가장 많이 나온 장르를 인기있는 노래로 인식하고, 가장 인기있는 장르로 출력

- 테스트 결과 스크린샷





가장 인기있는 장르를 확인하기 위해 사용자1, 2, 4, 에게는 발라드를, 사용자 3에게는 힙합을, 사용자 5에게는 pop송을 넣는다.



가장 인기있는 장르를 출력하기 위해 7을 입력하면, 발라드: 3, 힙합: 1, POP: 1 이기 때문에 가장 인기있는 장르로 발라드가 출력된다.

(4) 고객센터, 중복되는 노래 제거

- 설명: 이용자들에게 메뉴얼을 입력 받아, 고객센터에서 쿠폰을 등록하거나, 사용설명서의 출력, 이용권을 해지할 수 있는 기능. 겹치는 노래를 플레이리스트에서 제거하는 기능

- 테스트 결과 스크린샷

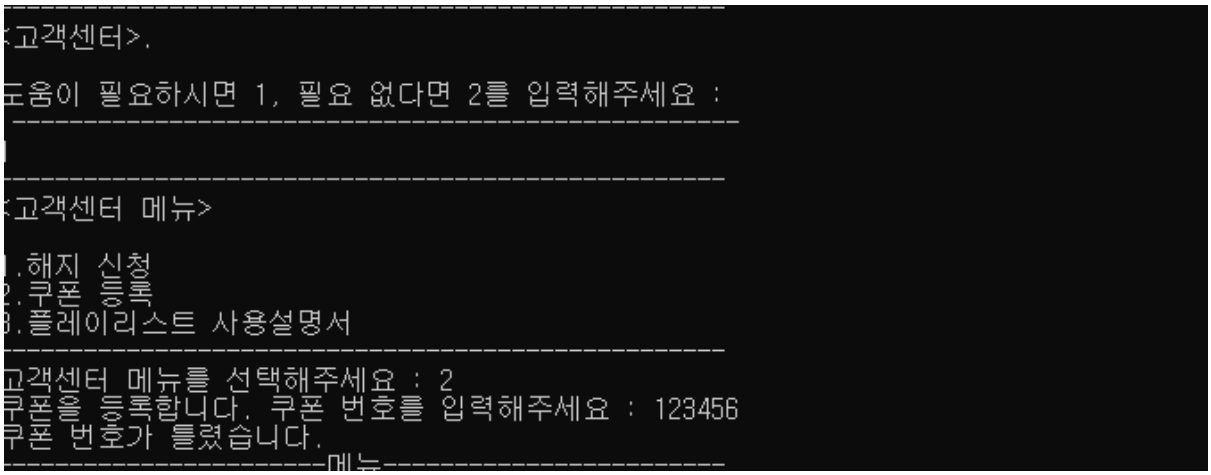
```
------------------------------------------------
메뉴를 입력해주세요 (1~8) : 8
------------------------------------------------
<고객센터>.

도움이 필요하시면 1, 필요 없다면 2를 입력해주세요 :
 ------------------------------------------------
2
고객센터를 종료합니다
----------------------메뉴----------------------
1. 노래 추가
2. 노래 삭제
3. 재생목록
4. 프로그램 종료
5. 노래 수정
6. 유사한 노래
7. 인기 장르
8. 고객센터
------------------------------------------------
```

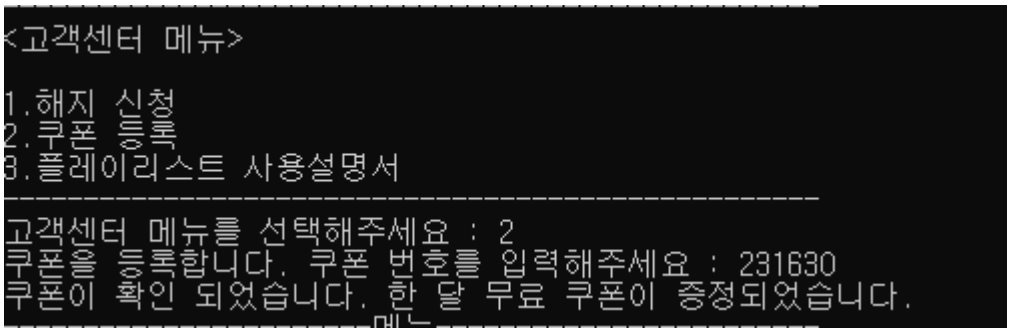
```
------------------------------------------------------
메뉴를 입력해주세요 (1~8) : 8
------------------------------------------------------
<고객센터>.

도움이 필요하시면 1, 필요 없다면 2를 입력해주세요 :
 ------------------------------------------------------
1
------------------------------------------------------
<고객센터 메뉴>

1.해지 신청
2.쿠폰 등록
3.플레이리스트 사용설명서
------------------------------------------------------
고객센터 메뉴를 선택해주세요.
```

고객센터에 들어가기 위해 8을 입력 후, 도움이 필요 없을 시 2를 눌러 나가고, 1을 눌러 고객센터에 들어감.

```
<고객센터 메뉴>

1.해지 신청
2.쿠폰 등록
3.플레이리스트 사용설명서
------------------------------------------------------
고객센터 메뉴를 선택해주세요.1
이용권을 해지합니다.
```

고객센터에서 1 입력 시, 이용권을 해지한다는 문구 출력

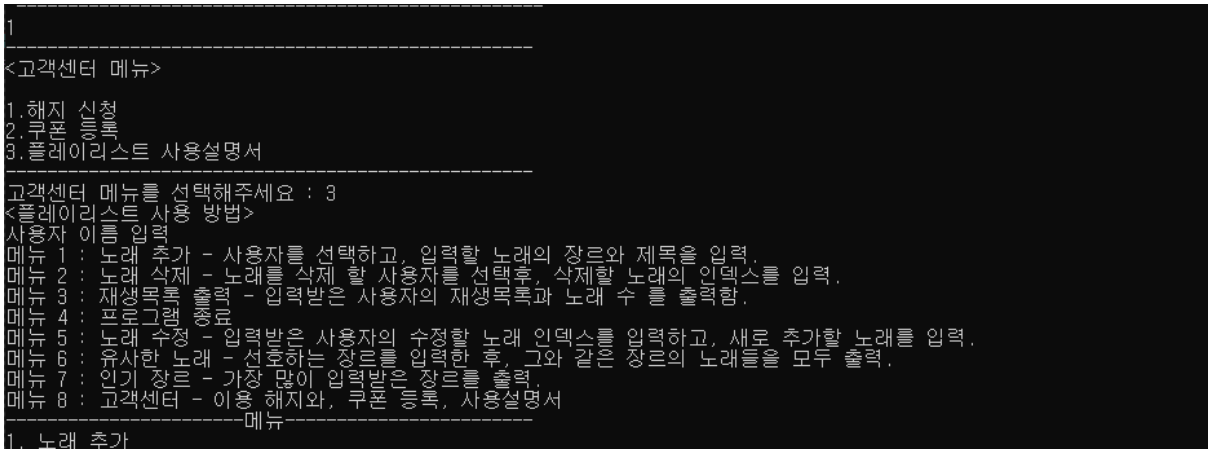




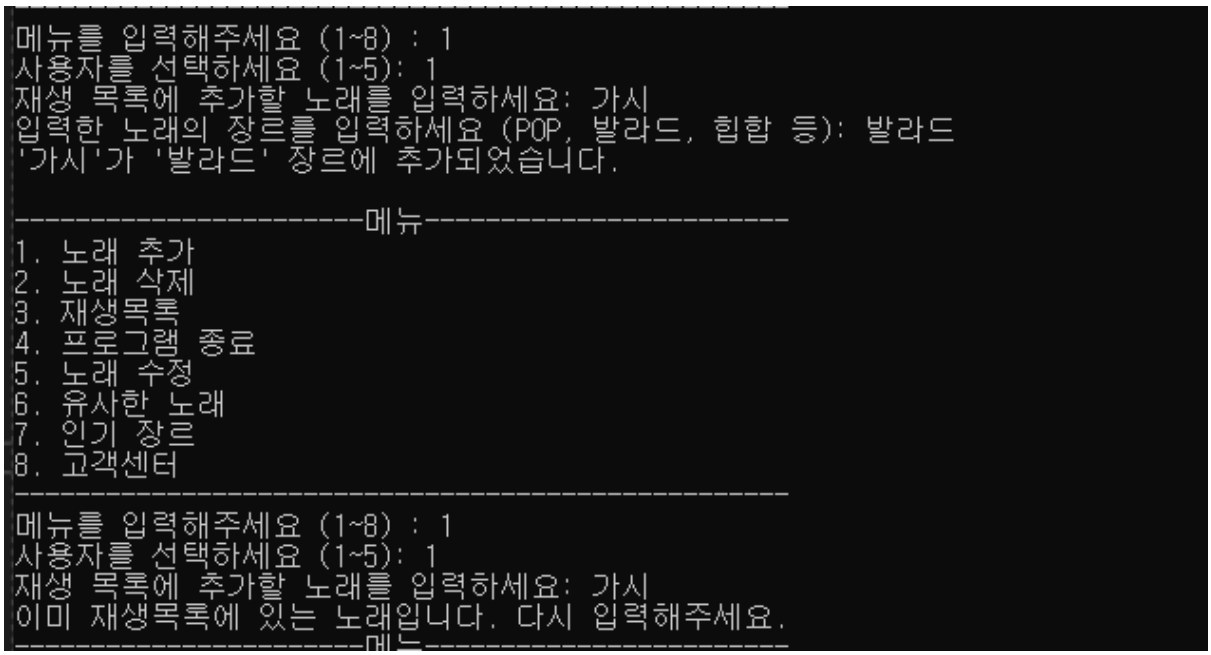
쿠폰을 등록하기 위해 고객센터에 2 입력 시, 쿠폰을 입력 받음.

쿠폰번호 정답은 coupon_answer 변수를 통해 학번 : 231630으로 설정해 놓음.

틀렸을 시에는 틀렸다는 문구를, 쿠폰 정답과 일치할 시에는 무료 쿠폰을 증정문구 출력.



플레이리스트의 사용방법을 출력하기 위해 8>1>3을 순서대로 입력하면 플레이리스트의 메뉴 입력 별 사용 방법이 출력되는 것을 확인.



중복 노래를 제거하기 위해, 1>1을 입력하여 사용자 1에게 '가시' 노래를 추가한 뒤

또 다시 1>1을 입력하여 '가시' 노래를 추가하려 했을 때, 이미 재생목록에 있는 노래라는 문구를 출력하고 다시 메뉴로 돌아가는 것을 확인할 수 있다.

5. 계획 대비 변경 사항

1) 기능 4: 고객센터 + 노래 중복 제거

- 이전: 고객센터 + 노래 중복 제거 + 나이 입력
- 이후: 고객센터(기능 추가) + 노래 중복 제거
- 사유: 나이를 입력 받고 추가하는 것은 어렵지 않았지만, 나이를 입력 받고, 노래와 장르와 엮으려다 보니 너무 복잡해질 거 같아서 계획을 변경하였다.

6. 느낀점

C 수업을 듣기 전에, 수업을 듣는 전공자 친구들과는 달리, 비전공자이며 컴퓨터 언어를 접해본 적이 없는 저에게 C프로그래밍 수업 조금 벅찰 것이라고 생각했지만, 파이썬 수업과 함께 병행하다 보니, 서로 다른 두 언어의 차이점을 비교해가며 배우고, 실습하는 것이 흥미로웠던 거 같습니다. 초반엔 파이썬과 많이 다른 코드 작성 방식에 적응하기 힘들었지만, 올려 주신 실습과제나 수업시간에 배운 내용을 조금씩 따라 쳐보니 금방 익숙해질 수 있었고, 시험공부를 하면서 혼자 실습을 하며 에러나는 부분을 검색해보고, 수업시간에 배운 개념을 조금 더 확장하여 독학하는 재미가 있었던 거 같습니다. 그리고 수업시간에 c프로그래밍 뿐만 아니라 코딩스타일이나, 디버깅 방법, 깃허브 사용법 등 유용한 정보들을 많이 알려주셔서 도움이 많이 됐습니다.

또한 숙제를 할 때, 기존에 작성해 놓았던 코드(students_grade.c, to_do_manager 등)을 계속 확장하여서 배열을 포인터로 바꾸어 보고, 구조체를 통해 배열을 관리해보고, 코드를 함수화하고, 함수를 헤더파일화 했던 것이 이 프로젝트를 혼자 진행하는 데에 있어서 많은 도움이 된 거 같습니다. 프로젝트를 진행하기 전에는 내가 스스로 어디까지 코딩할 수 있고, 어디까지 적용할 수 있는지 감이 잘 오지 않아서 거창한 계획을 세워놓았지만, 진척보고서를 쓰면서 계획도 수정하고, 코드 테스트 결과 설명, 코드 설명하기를 통해 좀 더 구체적이고 실현가능한 계획이 필요하다는 것을 알게 되었습니다. 내가 계획한 것을 스스로 성취해 나갈 수 있다는 것이 좋은 경험이었고, 다음에는 조별 프로젝트나 단체활동을 통해 한 문제를 해결해보고 싶은 마음이 생겼습니다. 한 학기동안 수고하셨습니다!!