CSED101. Programming & Problem solving
Fall 2022

Programming Assignment #4
(75 points)

김장원 (jangwonkim@postech.ac.kr)

■ **제출 마감일: 2022.12.14 23:59**

■ **개발 환경: Windows Visual Studio 2019**

■ **제출물**

- C 소스 코드 (assn4.c, functions.c, functions.h)
  - ➢ 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn4.docx, assn4.hwp 또는 assn4.pdf)
  - ➢ 보고서는 AssnReadMe.pdf를 참조하여 작성하시면 됩니다.
  - ➢ 명예 서약 (Honor code): 표지에 다음의 서약을 기입하여 제출해 주세요: "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예 서약이 기입되어 있지 않은 과제는 제출되지 않은 것으로 처리됩니다.
  - ➢ 작성한 소스 코드와 보고서 파일은 PLMS를 통해 제출해 주세요.

■ **주의 사항**

- 컴파일이나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리).
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 모든 문제의 출력 형식은 채점을 위해 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 "POSTECH 전자컴퓨터공학부 부정행위 정의"를 따릅니다 (PLMS의 본 과목 공지사항에 등록된 글 중, 제목이 [document about cheating]인 글에 첨부되어 있는 disciplinary.pdf를 참조하세요).
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem: 오래된 MP3 플레이어: 음악 플레이리스트 만들기

[요약]

우리가 좋아하는 음악을 추가 또는 삭제할 수 있는 플레이리스트를 만들어봅니다.

[개요]

우리가 구현할 플레이리스트의 주요 기능은 아래와 같습니다.

1. (add) 새로운 음악을 플레이리스트에 추가합니다.
2. (delete) 선택한 음악을 플레이리스트에서 삭제합니다.
3. (show) 플레이리스트 내의 음악 목록을 보여줍니다.
4. (show_favorites) 플레이리스트 내 선호도가 높은 음악 목록을 보여줍니다.
5. (exit) 프로그램을 종료합니다.

[목적]

- 구조체(struct)와 연결리스트(linked list) 사용법을 익힙니다.
- 다중 소스파일의 사용법을 익힙니다.

[주의사항]

1. 이번 과제는 총 5개의 명령어 (표1 참고)를 입력 받아 각 기능을 수행합니다. 적어도 각 명령어 별로 함수를 정의하여 사용해야 합니다. 그 외에 필요한 함수는 따로 정의해서 사용하실 수 있습니다. 기능적으로 독립됐거나 반복적으로 사용되는 기능은 사용자 함수를 정의해서 구현하도록 합니다.

   아래 표는 각 명령어 별로 그 기능을구현해야 할 함수의 이름입니다. 표를 참고하여 함수를 선언을 해주세요.

명령어	함수명
add	add_fn
delete	delete_fn
show	show_fn
show_favorites	show_favorites_fn
exit	exit_fn

2. 이번 과제는 여러 개의 파일로 분할해서 작성합니다.
   - **functions.h**
     구조체 정의 및 5가지 명령어에 대한 기능을 수행하는 함수 선언 부분
   - **functions.c**
     5가지 명령어에 대한 기능 수행을 위한 함수 정의
   - **assn4.c**
     - main() 함수를 포함하여 필요한 함수들을 정의
     - main() 함수 내에서 사용자로부터 명령어를 입력 받아 처리 알맞은 함수를 호출해 처리하는 기능이 구현되어 있어야 함
3. 이번 과제는 구조체와 연결리스트를 활용하는 것이 목표이므로, 문제에 구조체와 연결리스트를 언급한 부분(플레이리스트)을 배열을 이용하여 해결할 시 감점 처리됩니다.

4. 명시된 에러 처리 외에는 고려할 필요가 없습니다.
5. 전역 변수 및 goto 문은 사용할 수 없습니다.
6. string.h 에서 제공하는 라이브러리 함수를 사용할 수 있습니다.
7. 프로그램 구현 시, main() 함수를 호출하여 사용하지 않습니다. 즉, 소스 코드 내에 main(); 이라고 호출하지 않습니다.
8. 문제의 출력 형식은 채점을 위해 아래의 실행 예시와 최대한 비슷하게 작성해 주세요.

[설명 및 요구사항]

프로그램 구현은 구조체와 연결리스트를 사용하여야 하며, 각 항목에 대한 제약은 다음과 같다.

- 플레이리스트: 음악 제목, 아티스트 이름, 용량, 선호도 등의 정보를 가진 구조체를 정의하고 연결리스트로 구현한다.
    - 음악의 제목: 14자 이내의 공백 없는 문자열
    - 아티스트 이름: 14자 이내의 공백 없는 문자열
    - 용량: 음악의 용량(MB). 0 ~ 50 사이의 실수
    - 선호도: 음악의 선호도. 0 ~ 10 사이의 실수

[1. 프로그램 시작 화면]

프로그램을 실행하면 아래와 같이 읽어올 파일(.txt) 이름을 입력 받습니다. 파일에는 초기에 추가할 음악의 타이틀, 아티스트, 용량(MB) 그리고 선호도에 대한 정보가 들어있습니다. 또한 파일 이름에는 공백이 없으며, 확장자를 포함해 30 자를 넘지 않는다고 가정합니다.

```
음악 리스트 파일 이름을 입력해주세요. >>
```

만약 존재하지 않는 파일의 이름을 입력 받으면 아래와 같은 [에러 메시지 #1]를 출력 후 다시 파일 이름을 입력 받습니다. (실행 예시의 노란색 글자는 사용자 입력에 해당하며 초록색 글자는 실제 화면에는 출력되지는 않으나 문제 이해를 돕기 위한 주석입니다.)

```
음악 리스트 파일 이름을 입력해주세요. >> wrong.txt
유효하지 않은 파일입니다. 음악 리스트 파일을 다시 입력해주세요. >>   //에러 메시지 #1
```

[2. 플레이리스트(연결리스트 구조) 구현]

파일을 열어 위에서부터 읽으며, 연결리스트(이하 플레이리스트)에 삽입합니다. 주어진 파일 music_list.txt 의 내용과 구성은 아래와 같습니다.

```
ShutDown        BLACKPINK       8.2     5
ANTIFRAGILE     LESSERAFIM      6.4     8.5
AfterLIKE       IVE             7.8     6
HypeBoy         Newjeans        5.0     7
RushHour        Crush           7.6     8
Nxde            GIDLE           5.5     3
Monologue       TEI             7.3     4
LoveDive        IVE             11.3    9
PinkVenom       BLACKPINK       6.0     6
Cookie          Newjeans        1.9     5.5
FOREVER1        SNSD            5.1     3.5
```

▶ 한 줄에 하나의 음악 정보가 들어가며, 각 음악의 정보는 앞에서부터 차례로 타이틀, 아티스트, 용량 그리고 선호도로 구성됩니다. 한 줄의 정보는 tab(‘ \t ’)로 구분됩니다.

▶ 타이틀명과 아티스트명의 길이는 14 글자를 초과하지 않으며 공백은 존재하지 않습니다. 또한 영문 대소문자 및 숫자로만 구성되어 있습니다.

▶ 중복되는 타이틀명은 존재하지 않는다고 가정합니다.

[주의!]
우리가 구현할 플레이리스트는 용량 제한 (총 50MB)이 있습니다. 따라서 안타깝게도 파일에 있는 음악을 모두 넣을 수 없는 경우가 생길 수 있습니다. 만약 파일을 읽는 도중 용량이 초과되어 어떤 음악을 넣을 수 없을 경우, 아래와 같은 [에러 메시지 #2]를 출력합니다. (예시: Nxde 라는 곡을 못 넣은 경우)

```
용량 초과! 음악(Nxde)은 추가되지 않았습니다.     // 에러 메시지 #2
```

그리고 입력에 실패한 음악은 건너 띄고 다음 음악을 넣는 시행을 재개합니다. 예를 들어 위 예시 파일 music_list.txt 를 입력하였을 때 나오는 에러 메시지는 다음과 같습니다.

```
용량 초과! 음악(LoveDive)은 추가되지 않았습니다.
용량 초과! 음악(PinkVenom)은 추가되지 않았습니다
용량 초과! 음악(FOREVER1)은 추가되지 않았습니다.
```

위 파일 예시에서 타이틀명 Cookie 는 추가되었음에 주의해주세요.

[음악 추가 조건]

음악은 아래와 같은 조건을 따르며 플레이리스트에 추가됩니다.

▶ 플레이리스트 내의 음악의 총 용량은 50MB 를 초과할 수 없습니다.

▶ 플레이리스트 내의 음악들은 타이틀명을 기준으로 사전 순서대로 정렬되어야 합니다.

- **'사전 순서'**는 *alphabetical order* 를 의미합니다. 즉, 아스키코드값(숫자 < 대문자 < 소문자 순)을 기준으로 오름차순으로 정렬하면 됩니다. 예를 들어 다음 3 개의 문자열 (abc, Bbc, 1bc)이 있을 때, 이 문자열들을 사전 순서로 배열하면 (1bc, Bbc, abc)가 됩니다.
- 구현 시, string.h 에서 제공하는 아스키 코드값을 기준으로 비교하는 strcmp 함수를 활용하세요.

파일에 있는 모든 음악의 추가를 시도하였으면, 프로그램은 아래와 같이 [명령 메시지]를 출력하며, 다음 명령을 기다립니다.

```
명령어를 입력해주세요. >>        //명령 메시지
```

우리가 구현해야 할 명령은 총 5 가지이며 각 기능은 다음과 같습니다.

명령어	설명
show	플레이리스트 내 수록된 음악 정보를 출력합니다.
show_favorites	플레이리스트 내 선호도가 높은 음악 정보를 출력합니다.
add	새로운 음악을 플레이리스트에 추가합니다.
delete	특정 타이틀명을 가진 음악을 플레이리스트에서 삭제합니다.
exit	프로그램을 종료합니다.

표 1 명령어의 종류와 설명

만약 위 5 가지 명령 외에 다른 명령을 입력하면 프로그램은 아래와 같은 [에러 메시지 #3]를 출력한 후 다시 [명령 메시지]를 출력합니다.

```
유효하지 않은 명령어입니다.        //에러 메시지 #3

명령어를 입력해주세요. >>        //명령 메시지
```

[3. 플레이리스트 명령어 구현]

A. show 명령어 구현

show 명령어를 통해 프로그램은 현재 플레이리스트 내 음악 목록을 타이틀명을 기준으로 사전순서대로 출력합니다. 출력된 음악 목록 순서는 연결리스트에 삽입 되어있는 음악의 순서와 같아야 합니다.
show 명령어를 통해 음악 목록을 출력한 후 프로그램은 다시 [명령 메시지]를 출력하고 다음 명령을 기다립니다. 위 예시의 music_list.txt 파일을 입력한 후 show 명령어를 입력하면, 프로그램은 아래와 같은 화면을 출력합니다.

```
음악 리스트 파일 이름을 입력해주세요. >> music_list.txt
용량 초과! 음악(LoveDive)은 추가되지 않았습니다.
용량 초과! 음악(PinkVenom)은 추가되지 않았습니다
용량 초과! 음악(FOREVER1)은 추가되지 않았습니다.

명령어를 입력해주세요. >> show
                                    PLAYLIST
================================================================================
 No.          Title                  Artist            Volume          Preference
--------------------------------------------------------------------------------
#1  ¦   ANTIFRAGILE     ¦        LESSERAFIM     ¦       6.40 MB    ¦    8.50
#2  ¦     AfterLIKE     ¦               IVE     ¦       7.80 MB    ¦    6.00
#3  ¦        Cookie     ¦           Newjeans     ¦       1.90 MB    ¦    5.50
#4  ¦       HypeBoy     ¦           Newjeans     ¦       5.00 MB    ¦    7.00
#5  ¦     Monologue     ¦               TEI      ¦       7.30 MB    ¦    4.00
#6  ¦          Nxde     ¦             GIDLE      ¦       5.50 MB    ¦    3.00
#7  ¦      RushHour     ¦             Crush      ¦       7.60 MB    ¦    8.00
#8  ¦      ShutDown     ¦         BLACKPINK      ¦       8.20 MB    ¦    5.00
--------------------------------------------------------------------------------
Total number of songs: 8
Storage: 49.70 MB
================================================================================

명령어를 입력해주세요. >>
```

출력된 화면에는 다음 정보가 반드시 포함되어야 합니다.

1) Total number of songs: 플레이리스트에 수록된 음악의 수.
2) Storage: 플레이리스트에 수록된 음악의 총 용량.
3) #(번호): 수록된 곡의 번호 (연결리스트에 삽입된 곡의 순서와 같아야 함).
4) 각 음악의 타이틀명, 아티스트명, 용량, 선호도.

참고)

타이틀명과 아티스트명의 최대 길이는 14 자로 제한했기 때문에, 아래와 같이 출력 형식을 입력하면 예시처럼 빈 공간이 있고, 오른쪽 정렬이 되어있는 결과를 얻을 수 있습니다.

```
printf("Title: %14s, Artist: %14s\n", title, artist);
```

만약 플레이리스트에 아무런 곡도 수록되어 있지 않다면, 아래와 같은 [에러 메시지 #4]를 출력합니다.

```
Empty Playlist!                    //에러 메시지 #4
```

그 후 다시 [명령 메시지]를 출력합니다.
다음 화면은 빈 플레이 리스트에 show 명령어를 입력한 경우의 예시입니다.

```
명령어를 입력해주세요. >> show
                                    PLAYLIST
================================================================================
 No.          Title                  Artist            Volume          Preference
--------------------------------------------------------------------------------
Empty Playlist!
--------------------------------------------------------------------------------
Total number of songs: 0
Storage: 0.00 MB
================================================================================

명령어를 입력해주세요. >>
```

B. show_favorites 명령어 구현

show_favorites 명령어를 통해 프로그램은 플레이리스트 내 선호도가 높은 음악 목록을 보여줍니다. 구체적으로 이 명령어를 사용하면 추출할 음악의 수 K 개를 입력 받는데, 높은 선호도를 기준으로 K 개의 음악 목록을 출력하게 됩니다. 예를 들어, 위 예시 파일 music_list.txt 파일로 생성된 플레이리스트에서 선호도 기준으로 상위 5 개의 음악을 출력하고 싶으면 아래와 같이 show_favorites 명령어를 사용하면 됩니다.

```
명령어를 입력해주세요. >> show_favorites
상위 몇 개의 음악을 추출할까요? >> 5      //추출할 음악의 수를 입력합니다.
                                    FAVORITES
================================================================================
 No.          Title                  Artist            Volume          Preference
--------------------------------------------------------------------------------
#1  ¦   ANTIFRAGILE     ¦        LESSERAFIM     ¦       6.40 MB    ¦    8.50
#2  ¦      RushHour     ¦             Crush     ¦       7.60 MB    ¦    8.00
#3  ¦       HypeBoy     ¦          Newjeans     ¦       5.00 MB    ¦    7.00
#4  ¦     AfterLIKE     ¦               IVE     ¦       7.80 MB    ¦    6.00
#5  ¦        Cookie     ¦          Newjeans     ¦       1.90 MB    ¦    5.50
--------------------------------------------------------------------------------
Total number of songs: 5
Storage: 28.70 MB
================================================================================

명령어를 입력해주세요. >>
```

show_favorites 명령어를 통해 선호도가 높은 음악 목록을 출력한 후 프로그램은 다시 [명령 메시지]를 출력하고 다음 명령을 기다립니다. 또한 선호도(preference)가 같은 경우, 타이틀명의 사전순서가 앞선 음악을 우선하여 출력합니다.

만약 추출할 음악의 수가 전체 음악의 수보다 많은 경우 아래와 같이 [에러 메시지 #5]를 출력하고 다시 [명령 메시지]를 출력합니다.

```
추출하고자 하는 음악의 수가 전체 음악의 수보다 많습니다.      //에러 메시지 #5

명령어를 입력해주세요. >>
```

또한 추출하고자 하는 음악의 수가 1 보다 작은 경우 아래와 같이 [에러 메시지 #6]를 출력하고 다시 [명령 메시지]를 출력합니다.

```
추출하고자 하는 음악의 수는 1 이상이어야 합니다.          //에러 메시지 #6

명령어를 입력해주세요. >>
```

C. delete 명령어 구현

delete 명령어는 플레이리스트 내 특정 음악을 삭제하는 명령어입니다. 다음과 같이 삭제할 곡의 타이틀명을 입력 받아 삭제합니다. 삭제 시에는 해당 음악 정보의 저장을 위해 동적 할당 받은 메모리를 할당 해제(free) 합니다.
아래 예시는 위 플레이리스트에서 타이틀명으로 AfterLIKE 를 입력하여 해당 음악 정보를 삭제하고 show 명령어로 출력한 화면입니다.

```
명령어를 입력해주세요. >> delete
삭제할 음악의 타이틀을 입력해주세요. >> AfterLIKE

================================================================================
 No.          Title                 Artist            Volume          Preference
--------------------------------------------------------------------------------
#2  ¦     AfterLIKE     ¦              IVE     ¦       7.80 MB    ¦    6.00
--------------------------------------------------------------------------------
위 음악이 삭제되었습니다.

명령어를 입력해주세요. >> show
                                  PLAYLIST
================================================================================
 No.          Title                 Artist            Volume          Preference
--------------------------------------------------------------------------------
#1  ¦   ANTIFRAGILE     ¦        LESSERAFIM     ¦       6.40 MB    ¦    8.50
#2  ¦        Cookie     ¦          Newjeans     ¦       1.90 MB    ¦    5.50
#3  ¦       HypeBoy     ¦          Newjeans     ¦       5.00 MB    ¦    7.00
#4  ¦     Monologue     ¦               TEI     ¦       7.30 MB    ¦    4.00
#5  ¦          Nxde     ¦             GIDLE     ¦       5.50 MB    ¦    3.00
#6  ¦      RushHour     ¦             Crush     ¦       7.60 MB    ¦    8.00
#7  ¦      ShutDown     ¦         BLACKPINK     ¦       8.20 MB    ¦    5.00
--------------------------------------------------------------------------------
Total number of songs: 7
Storage: 41.90 MB
================================================================================

명령어를 입력해주세요. >>
```

만약 플레이리스트에 삭제하고자 하는 음악이 없으면 프로그램은 아래처럼 [에러 메시지 #7]를 출력 합니다.

```
플레이리스트에 해당 음악(검색한 타이틀명)이 없습니다.        //에러 메시지 #7
```

다음은 현재 플레이리스트에 없는 타이틀명을 입력한 경우의 예시입니다.

```
명령어를 입력해주세요. >> delete
추가할 음악의 타이틀을 입력해주세요. >> WrongTitle
플레이리스트에 해당 음악(WrongTitle)이 없습니다.

명령어를 입력해주세요. >>
```

delete 명령어를 시행한 후 (삭제의 성공 여부와 상관없이) 프로그램은 다시 [명령 메시지]를 출력하고 다음 명령을 기다립니다.

D. add 명령어 구현

add 명령어를 통해 프로그램은 현재 플레이리스트에 새로운 음악을 추가할 수 있습니다. add 명령어를 입력한 후 다음과 같이 추가할 곡의 타이틀명, 아티스트명, 용량, 그리고 선호도를 용량을 차례로 입력하여 플레이리스트에 곡을 추가합니다.
아래 예시는 위 플레이리스트에서 add 명령어를 통해 새로운 곡을 삽입하고 show 명령어를 통해 출력한 화면입니다.

```
명령어를 입력해주세요. >> add
추가할 음악의 타이틀을 입력해주세요. >> NextLevel
추가할 음악의 아티스트를 입력해주세요. >> aespa
추가할 음악의 용량을 입력해주세요. >> 3.53
추가할 음악의 선호도를 입력해주세요. >> 7

명령어를 입력해주세요. >> show
                                    PLAYLIST
=============================================================================================
 No.          Title                  Artist            Volume          Preference
---------------------------------------------------------------------------------------------
#1  ¦   ANTIFRAGILE     ¦        LESSERAFIM     ¦       6.40 MB    ¦    8.50
#2  ¦        Cookie     ¦          Newjeans     ¦       1.90 MB    ¦    5.50
#3  ¦       HypeBoy     ¦          Newjeans     ¦       5.00 MB    ¦    7.00
#4  ¦     Monologue     ¦               TEI     ¦       7.30 MB    ¦    4.00
#5  ¦     NextLevel     ¦             aespa     ¦       3.53 MB    ¦    7.00
#6  ¦          Nxde     ¦             GIDLE     ¦       5.50 MB    ¦    3.00
#7  ¦      RushHour     ¦             Crush     ¦       7.60 MB    ¦    8.00
#8  ¦      ShutDown     ¦         BLACKPINK     ¦       8.20 MB    ¦    5.00
---------------------------------------------------------------------------------------------
Total number of songs: 8
Storage: 45.43 MB
=============================================================================================

명령어를 입력해주세요. >>
```

add 명령어를 통해 추가할 음악은 다음과 같은 조건을 따릅니다.

[add 명령어 구현 시 조건]

- ▶ 플레이리스트 내의 음악의 총 용량은 50MB 를 초과할 수 없습니다. 만약 추가할 음악이 플레이리스트 안의 용량을 50MB 보다 크게 만들면 음악을 추가할 수 없으며 [에러 메시지 #2]를 출력합니다.
- ▶ 추가할 음악은 반드시 플레이리스트 내의 사전 순서 (타이틀명 기준)를 만족하는 위치로 삽입되어야 합니다.
- ▶ 만약 추가하고자 하는 음악이 이미 플레이리스트 내에 존재한다면, 아래와 같이 [에러 메시지 #8]를 출력합니다.

```
해당 음악이 이미 플레이리스트 내에 존재합니다.      //에러 메시지 #8
```

- ▶ add 명령어를 시행한 후 (추가의 성공 여부와 상관없이) 프로그램은 다시 [명령 메시지]를 출력하고 다음 명령을 기다립니다.

E. exit 명령어 구현

exit 명령어 입력 시 프로그램을 종료합니다. 아래와 같이 저장할 파일명을 입력 받아, 현재까지 편집한 플레이리스트에 있는 내용을 텍스트 파일의 형식으로 저장을 해야 합니다.

- ▶ 파일 이름에는 공백이 없으며, 확장자를 포함해 30 자를 넘지 않는다고 가정합니다.
- ▶ 이 텍스트 파일에는 맨 처음 읽었던 예시 파일(music_list.txt)과 같은 형식으로 내용을 구성합니다. 음악의 목록과 같이 각 줄에 음악의 타이틀과 이에 해당하는 아티스트, 용량(MB) 그리고 선호도를 순서대로 기록합니다. (한 줄의 정보는 tab(' \t ')로 구분됩니다.) 파일은 소스파일이 있는 경로에 저장하도록 합니다.

이후, 아래와 같이 [종료 메시지]를 출력 후 프로그램을 종료합니다.

```
명령어를 입력해주세요. >> exit
저장할 파일명을 입력해주세요. >> new_music_list.txt
프로그램을 종료합니다.                //종료 메시지

계속하려면 아무 키나 누르십시오 . . .
```

프로그램 종료 시 동적 할당 받은 메모리를 모두 할당 해제한 후 종료해야 합니다.

[헤더 파일 작성]

- 헤더 파일 작성 예시 (1) (functions.h)

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

// 구조체 정의
…
// 함수 선언
…

#endif
```

- 헤더 파일 작성 예시 (2) (functions.h)

```
#pragma once
// 구조체 정의
…
// 함수 선언
…
```

위의 사용자 정의 헤더 파일 functions.h를 필요한 곳에서 include 하여 사용합니다.

헤더 파일 작성시 위와 같이 작성을 하는 이유에 대해서 간략하게 조사하여 보고서에 서술합니다. (과제 점수에 포함됩니다.)