

# CSED101: Assignment 4

무은재학부 손량 (20220323)

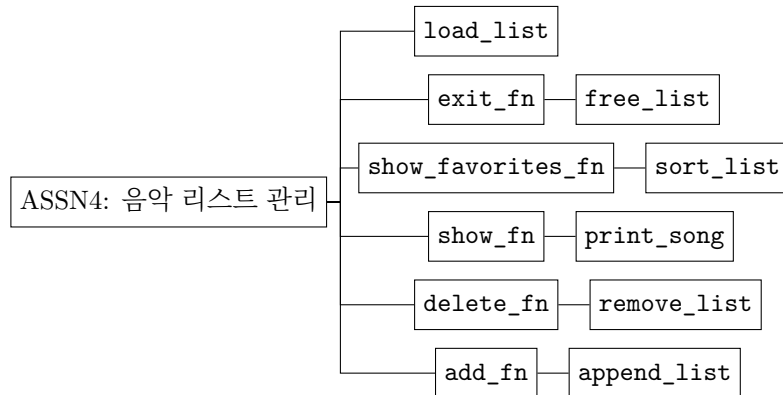
POVIS ID: ryangsohn

담당 교수: 윤은영 교수님

“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”

## 1 문제의 개요

이 프로그램은 음악 리스트 관리를 C언어로 구현한 것이다. 이 프로그램의 structure chart는 다음과 같다.<sup>1</sup>



## 2 프로그램 구조 및 설명

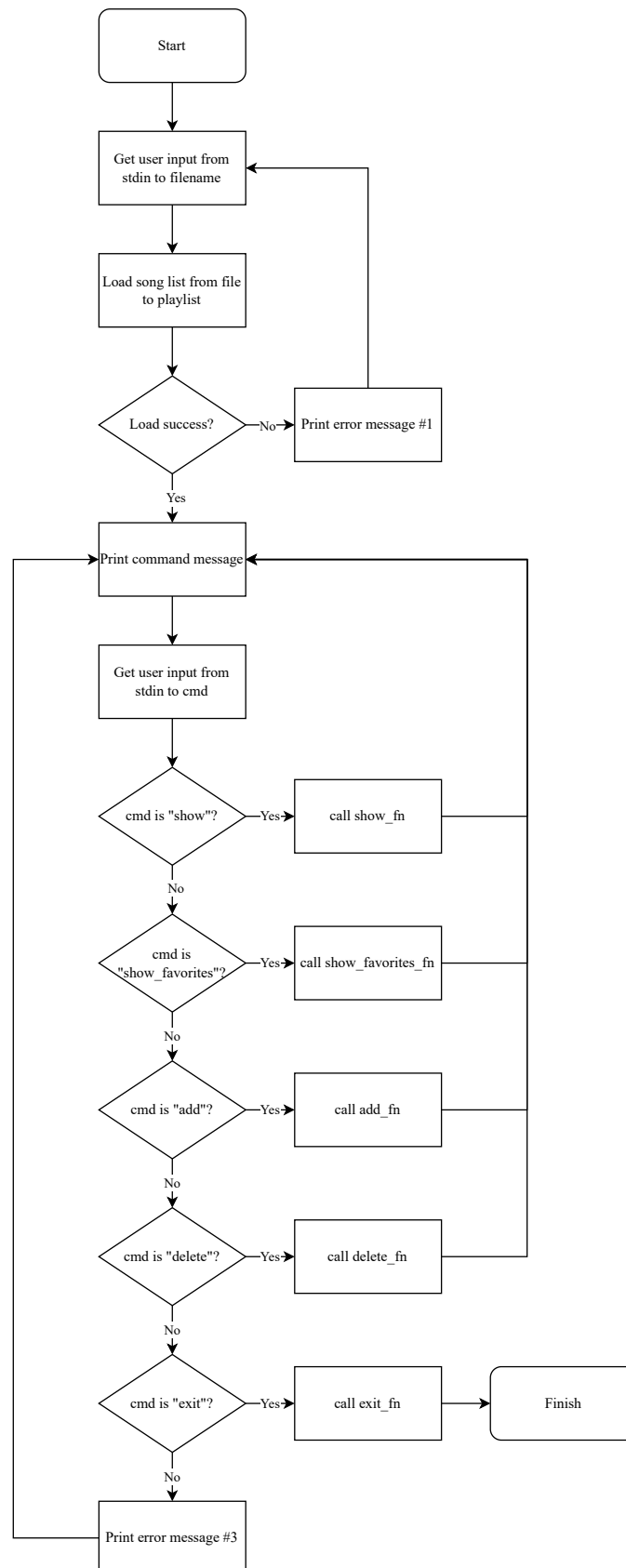
본 프로그램에서 사용한 알고리즘을 pseudocode로 나타내면 다음과 같다.

```

infinite loop:
  get user input from stdin to filename
  load song list from file to playlist
  if load was successful:
    break
  print error message #1
infinite loop:
  print command message
  get user input from stdin to cmd
  if cmd is "show":
    call show_fn
  else if cmd is "show_favorites":
    call show_favorites_fn
  else if cmd is "add":
    call add_fn
  else if cmd is "delete":
    call delete_fn
  else if cmd is "exit":
    call exit_fn
    break
  else
    print error message #3
  
```

Flowchart로 나타내면 다음과 같다.

<sup>1</sup>지면을 아끼기 위해 프로그램의 기능상 중요한 함수들 위주로 structure chart에 넣었다.



### 3 프로그램 실행 방법과 예제

첨부한 `assn4.c`, `functions.c`, `functions.h` 파일은 Linux, macOS 등의 UNIX 계열 OS에서 실행하는 것을 전제로 작성되었다.<sup>2</sup> gcc 컴파일러가 설치된 환경에서는 다음 명령어를 통해 코드를 컴파일할 수 있다.

```
$ gcc -o assn4 assn4.c functions.c
```

다음과 같은 입력 파일을 사용했다고 가정할 때,

Astronomia	Vicetone	8.2	5
Technologic	DaftPunk	6.4	8.5
EventHorizon	YOUNHA	7.8	6
Panorama	LEECHANHYUK	5.0	7
NoMoney	Galantis	7.6	8
iPad	Chainsmokers	5.5	3
IAintWorried	OneRepublic	7.3	4
LightSwitch	CharliePuth	5.1	3.5
Levels	Avicii	6.0	6
Shivers	EdSheeran	1.9	5.5
Clarity	Zedd	11.3	9

이를 읽어들이는 내용은 다음과 같다.

```

./assn4
1: ./assn4 x +
[~/Documents/csed101/assignments/assn4 main*]
λ ./assn4
음악 리스트 파일 이름을 입력해주세요. >> playlist.txt
윽랑 초과! 음악(LightSwitch)은 추가되지 않았습니다.
윽랑 초과! 음악(Levels)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.

```

Figure 1: 파일을 읽는 상황

파일이 존재하지 않는 경우의 예외 처리도 구현되어 있다.

```

./assn4
1: ./assn4 x +
[~/Documents/csed101/assignments/assn4 main*]
λ ./assn4
음악 리스트 파일 이름을 입력해주세요. >> does-not-exist
윽랑 초과! 음악(LightSwitch)은 추가되지 않았습니다.
윽랑 초과! 음악(Levels)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.
윽랑 초과! 음악(Clarity)은 추가되지 않았습니다.

```

Figure 2: 파일이 존재하지 않을 때의 예외 처리

여기서 `show` 명령어를 실행하면 다음과 같다.

<sup>2</sup>Windows의 경우 WSL이나 Cygwin 등의 환경에서 실행할 수 있을 것이다.



```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> show
PLAYLIST

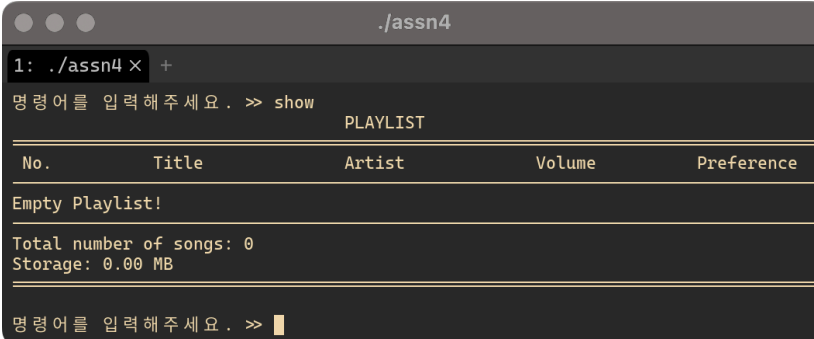

| No. | Title        | Artist       | Volume  | Preference |
|-----|--------------|--------------|---------|------------|
| #1  | Astronomia   | Vicetone     | 8.20 MB | 5.00       |
| #2  | EventHorizon | YOUNHA       | 7.80 MB | 6.00       |
| #3  | IAintWorried | OneRepublic  | 7.30 MB | 4.00       |
| #4  | NoMoney      | Galantis     | 7.60 MB | 8.00       |
| #5  | Panorama     | LEECHANHYUK  | 5.00 MB | 7.00       |
| #6  | Shivers      | EdSheeran    | 1.90 MB | 5.50       |
| #7  | Technologic  | DaftPunk     | 6.40 MB | 8.50       |
| #8  | iPad         | Chainsmokers | 5.50 MB | 3.00       |


Total number of songs: 8
Storage: 49.70 MB
명령어를 입력해주세요. >>

```

Figure 3: 로드된 모든 음악 출력

저장된 음악이 없을 때는 다음과 같이 “Empty Playlist!”라고 출력한다.



```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> show
PLAYLIST


| No.             | Title | Artist | Volume | Preference |
|-----------------|-------|--------|--------|------------|
| Empty Playlist! |       |        |        |            |


Total number of songs: 0
Storage: 0.00 MB
명령어를 입력해주세요. >>

```

Figure 4: 음악 리스트가 비어 있는 경우

존재하지 않는 명령어를 입력했을 때의 처리도 다음과 같이 구현되어 있다.



```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> aaa
유효하지 않은 명령어입니다.
명령어를 입력해주세요. >> bbb
유효하지 않은 명령어입니다.
명령어를 입력해주세요. >> ccc
유효하지 않은 명령어입니다.
명령어를 입력해주세요. >> ddd
유효하지 않은 명령어입니다.
명령어를 입력해주세요. >>

```

Figure 5: 존재하지 않는 명령어를 입력했을 경우

`show_favorites` 명령어의 실행 결과는 다음과 같다.

```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> show_favorites
상위 몇 개의 음악을 추출할까요? >> 5
FAVORITES
=====
No.      Title      Artist      Volume      Preference
-----
#1 | Technologic | DaftPunk | 6.40 MB | 8.50
#2 | NoMoney | Galantis | 7.60 MB | 8.00
#3 | Panorama | LEECHANHYUK | 5.00 MB | 7.00
#4 | EventHorizon | YOUNHA | 7.80 MB | 6.00
#5 | Shivers | EdSheeran | 1.90 MB | 5.50
=====
Total number of songs: 5
Storage: 28.70 MB
=====
명령어를 입력해주세요. >>

```

Figure 6: 선호도순 출력

현재 로드된 것보다 더 많은 음악이나, 더 적은 개수의 음악을 로드하려고 하면 오류가 발생한다.

```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> show_favorites
상위 몇 개의 음악을 추출할까요? >> 10
추출하고자 하는 음악의 수가 전체 음악의 수보다 많습니다.

명령어를 입력해주세요. >> show_favorites
상위 몇 개의 음악을 추출할까요? >> 0
추출하고자 하는 음악의 수는 1 이상이어야 합니다.

명령어를 입력해주세요. >> show_favorites
상위 몇 개의 음악을 추출할까요? >> -1
추출하고자 하는 음악의 수는 1 이상이어야 합니다.

명령어를 입력해주세요. >>

```

Figure 7: show\_favorites의 예외 처리

add 명령어의 실행 결과는 다음과 같다.

```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> add
추가할 음악의 타이틀을 입력해주세요. >> NAKKA
추가할 음악의 아티스트를 입력해주세요. >> AKMU
추가할 음악의 용량을 입력해주세요. >> 0.1
추가할 음악의 선호도를 입력해주세요. >> 6

명령어를 입력해주세요. >>

```

Figure 8: 음악을 추가하는 상황

이미 있는 음악을 추가하거나, 공간 제한을 초과하는 경우에는 오류가 발생한다.

```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> add
추가할 음악의 타이틀을 입력해주세요. >> iPad
추가할 음악의 아티스트를 입력해주세요. >> Chainsmokers
추가할 음악의 용량을 입력해주세요. >> 0.1
추가할 음악의 선호도를 입력해주세요. >> 8
해당 음악이 이미 플레이리스트 내에 존재합니다.

명령어를 입력해주세요. >> add
추가할 음악의 타이틀을 입력해주세요. >> Clarity
추가할 음악의 아티스트를 입력해주세요. >> Zedd
추가할 음악의 용량을 입력해주세요. >> 10
추가할 음악의 선호도를 입력해주세요. >> 8
용량 초과! 음악(Clarity)은 추가되지 않았습니다.

명령어를 입력해주세요. >>

```

Figure 9: add의 예외 처리

delete 명령어의 실행 결과는 다음과 같다.

```

./asn4
1: ./asn4 x +
명령어를 입력해주세요. >> delete
삭제할 음악의 타이틀을 입력해주세요. >> iPad



| No. | Title | Artist       | Volume  | Preference |
|-----|-------|--------------|---------|------------|
| #9  | iPad  | Chainsmokers | 5.50 MB | 3.00       |



위 음악이 삭제되었습니다.

명령어를 입력해주세요. >> delete
삭제할 음악의 타이틀을 입력해주세요. >> Clarity
플레이리스트에 해당 음악(Clarity)이 없습니다.

명령어를 입력해주세요. >>

```

Figure 10: delete의 동작과 예외 처리

exit 명령어의 실행 결과는 다음과 같다.

```

..gnments/asn4
1: ..gnments/asn4 x +
명령어를 입력해주세요. >> delete
삭제할 음악의 타이틀을 입력해주세요. >> Clarity
플레이리스트에 해당 음악(Clarity)이 없습니다.

명령어를 입력해주세요. >> exit
저장할 파일명을 입력해주세요. >> /tmp/playlist.txt
프로그램을 종료합니다.

[~/Documents/csed101/assignments/asn4 main*]
λ

```

Figure 11: exit의 동작과 예외 처리

## 4 토론

### 4.1 파일 분할

요구조건에도 나와 있듯 이번 어싸인에서는 코드를 `assn4.c`, `functions.c`, `functions.h` 파일에 분할하여 작성하였다. 파일을 분할하면서 `functions.h`의 맨 위에 `#pragma` 지시문을 사용하였다. 이를 사용하면 여러 번 같은 헤더 파일을 `include`했을 때에도 문제 없이, 첫 번째 `include`만 컴파일하게 해준다. `#pragma` 지시문은 큰 규모의 코드에서는 빌드 시간을 단축하는 효과도 있다고 한다.

### 4.2 연결 리스트의 할당과 할당 해제

이 프로그램에서는 연결 리스트의 노드를 메모리의 힙 공간에 할당한다. 동적 할당한 연결 리스트의 노드를 할당 해제할 필요가 있고, 이를 위해서 리스트의 노드를 할당 해제하는 함수 `free_list`를 작성하였다. `free_list` 함수에서는 리스트의 각 노드에 대해 반복하여, 노드를 할당 해제하고 다음 노드로 넘어가는 과정을 리스트의 맨 뒤까지 반복한다.

## 5 결론과 개선 방향

C언어를 활용하여 음악 리스트 관리 프로그램을 작성하였다. 프로그램은 잘 동작하지만, 몇 가지 개선할 만한 점을 여기 적어 보았다.

### 5.1 성능 최적화

ASSN4에서 작성한 코드는 음악 리스트의 크기가  $N$ 일 때, 리스트 정렬에는  $O(N^2)$ 회의 문자열 비교를 해야 한다. 이는 음악 리스트를 연결 리스트로 구현하였기 때문에 일어나는 일으로, 이진 트리를 사용하면 이 시간 복잡도를 줄일 수 있다. 이진 트리를 두 개 만들어서 하나는 곡명 순서대로 정렬하고, 나머지 하나는 선호도 순으로 정렬되도록 구성하면 된다. 원래 구현과 이진 트리를 사용한 구현의 시간 복잡도를 계산하면 다음과 같다. 제목 길이가  $L$ 인 상황을 가정했다.

	원래 구현	이진 트리를 사용한 구현
add	$O(LN)$	$O(L \lg N)$
delete	$O(LN)$	$O(L \lg N)$
show	$O(N)$	$O(N)$
show_favorites	$O(N^2)$	$O(\lg N)$

### 5.2 공백 지원

이 프로그램은 `scanf`를 통해 입력받기 때문에, 곡명 혹은 아티스트 이름에 공백이 있을 경우 정상적으로 작동하지 않는다. 입력 파일 파싱을 개선하여 문자열에 공백이 있더라도 문제 없이 동작하도록 개선하면 좋을 것이다.