Data Structure Lab. Project #1

2025년 9월 15일

Due date: 2025년 10월 12일 일요일 23:59:59 까지

본 프로젝트에서는 이진 탐색 트리(Binary Search Tree), 연결 리스트(Linked List), 큐(Queue) 자료 구조를 이용하여 나만의 음악 플레이 리스트 프로그램을 구현한다. 이 프로그램은 음악 데이터가 저장되어 있는 데이터 파일로부터 가수, 노래 제목, 재생 시간 정보를 읽어 Queue를 구축하며, 해당 Queue를 MusicQueue라 부른다. Queue에서 pop 명령어를 실행하면 데이터를 방출하여 가수 이름을 기준으로 정렬된 자료구조(ArtistBST)와 노래 제목을 기준으로 정렬된 자료구조(TitleBST)에 저장한다. 사용자가 플레이 리스트 생성 명령어를 실행하면, 두 BST에 저장된 음악 정보를 기반으로 순환 연결리스트를 구성하여 실제 플레이 리스트를 완성한다.

- 가수 이름을 기준으로 정렬된 자료구조

ArtistBST는 가수 이름, 해당 가수의 노래, 각 노래들의 재생 시간으로 구성되며, 가수 이름을 기준으로 정렬된다. 각 노드에는 해당 가수의 모든 노래 정보와 곡별 재생 시간이 저장된다.

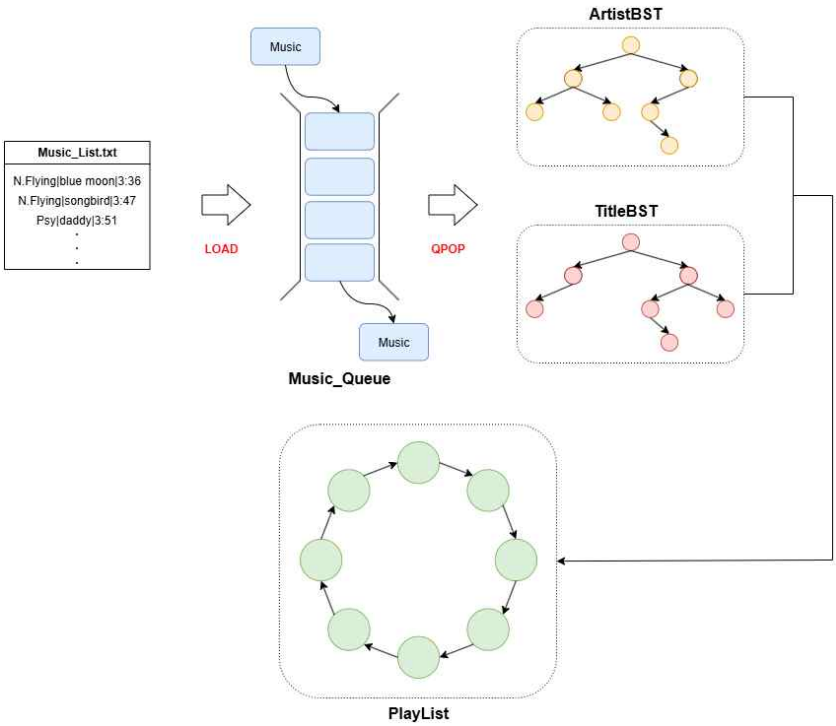
- 노래 제목을 기준으로 정렬된 자료구조

TitleBST는 노래 제목, 해당 제목을 보유한 가수 이름들, 각 노래들의 재생시간으로 구성되며, 노래 제목을 기준으로 정렬된다. 각 노드에는 동일한 제목을 가진 모든 곡 정보와 가수별 재생 시간이 저장된다.

- BST로부터 데이터로 이뤄진 자료구조

PlayList는 BST에 저장된 음악 데이터를 기반으로 구축되며, 순환 연결 리스트 형태로 구성된다. 각 노드는 단일 곡에 대한 정보(가수 이름, 노래제목, 재생 시간)으로 구성되며, 리스트 전체는 반복 재생(Loop) 기능을 지원한다.

각 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **Program Implementation**에서 설명한다.



[그림 1] 프로그램 구조

□ Program implementation

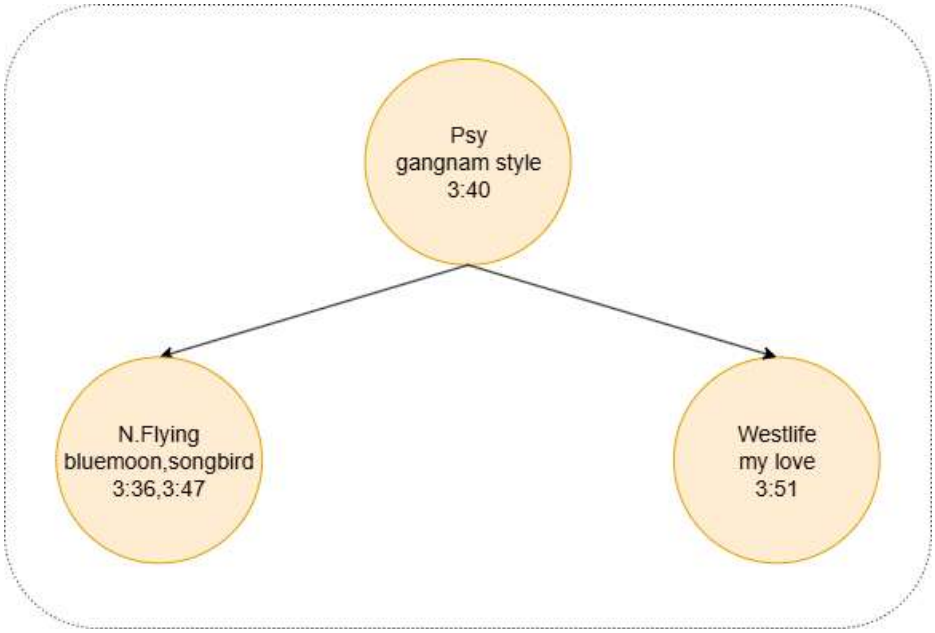
1) MusicQueue

- 주어진 Music_List.txt에 저장된 음악 데이터를 이용하여 구축한다. Queue의 자료구조에 따라 저장되는 순서대로 방출(FIFO)된다. Queue에는 Music_List.txt에 첫 번째 줄부터 마지막 줄까지 순서대로 저장되며, 이때 동일한 음악 데이터의 중복은 존재하지 않는다.
- 입력 데이터는 “가수 이름 | 노래 제목 | 재생 시간”(ex. N.Flying|blue moon|3:36)으로 구성되며, ‘|’문자를 기준으로 데이터가 나뉜다. 이때 노래 제목은 소문자로 구성되며, Music_List.txt에서 불러온 곡이 아닌, 새로 곡을 추가하는 경우, Queue에 이미 존재하는지 확인해야 한다.
- Queue를 구성하는 데이터는 MusicQueueNode class로 구현되며, 각 노드는 단일 곡의 형태이다.
- Queue의 크기는 100으로 초기화되고 이후 크기가 변하지 않으며, Queue가 비어 있을 때 POP 또는 전부 차 있을 때 PUSH를 수행되는 경우 프로그램이 종료된다.
- Queue에서 QPOP 명령어를 통해 방출되는 데이터는 두 BST의 입력으로 사용된다.

[그림 2] MusicQueue 예시

2) ArtistBST

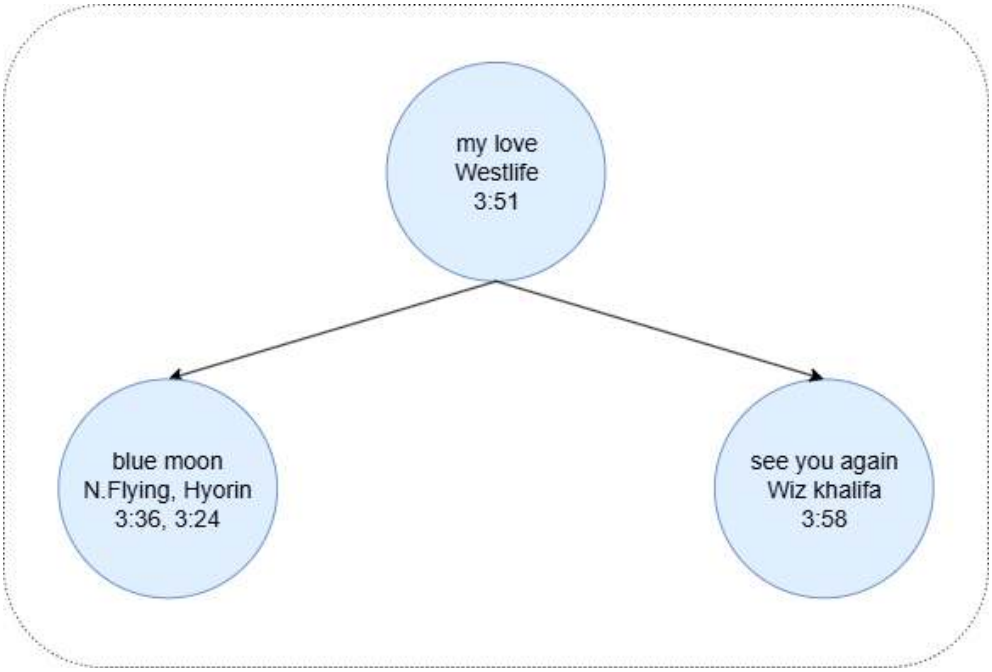
- Queue에서 방출된 데이터를 이용하여 구축된다.
- BST의 노트는 ArtistBSTNode class로 구현되며, 각 노드는 가수 이름, 해당 가수의 노래 목록, 각 노래별 재생 시간 정보를 갖는다.
- BST에서 지원하는 연산은 삽입, 삭제, 탐색, 출력(중위 순환)이며, 각 연산에 대해서는 **Functional Requirements**에서 자세히 설명한다.
- BST 연결 규칙은 다음과 같다.
  ① 부모 노드보다 가수 이름이 사전순으로 앞서는 경우 왼쪽 서브트리에, 사전순으로 뒤에 오는 경우 오른쪽 서브트리에 위치한다.
  ② 노드를 삭제할 때, 양쪽 자식 노드가 모두 존재하는 경우에는 오른쪽 서브트리에서 가장 작은 이름 노드를 삭제되는 노드 위치로 이동한다.



[그림 3] ArtistBST 예시

3) TitleBST

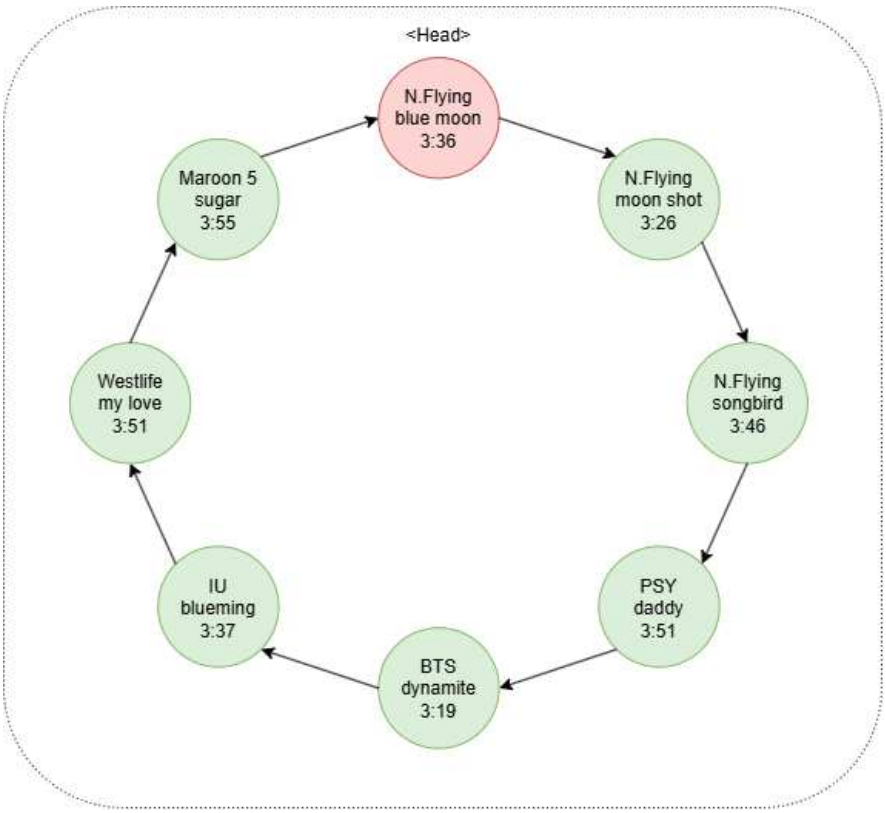
- Queue에서 방출된 데이터를 이용하여 구축된다.
- BST의 노드는 TitleBSTNode class로 구현되며, 각 노드는 노래 제목, 해당 제목을 가진 가수 이름 목록, 각 노래별 재생 시간 정보를 갖는다.
- BST에서 지원하는 연산은 삽입, 삭제, 탐색, 출력(중위 순환)이며, 각 연산에 대해서는 **Functional Requirements**에서 자세히 설명한다.
- BST 연결 규칙은 다음과 같다.
  ① 부모 노드보다 노래 제목이 사전순으로 앞서는 경우 왼쪽 서브트리에, 사전순으로 뒤에 오는 경우 오른쪽 서브트리에 위치한다.
  ② 노드를 삭제할 때, 양쪽 자식 노드가 모두 존재하는 경우에는 오른쪽 서브트리에서 가장 작은 제목 노드를 삭제되는 노드 위치로 이동한다.



[그림 4] TitleBST 예시

4) PlayList

- PlayList는 BST에 저장된 음악 데이터를 기반으로 구축된다.
- Circular Linked List의 노드는 PlayListNode class로 구현되며, 각 노드는 가수 이름, 노래 제목, 재생 시간 정보를 포함하고 순환(Loop) 구조를 유지한다.
- Circular Linked List에서 지원하는 연산은 삽입, 삭제, 탐색, 출력이며, 각 연산에 대해서는 Functional Requirements에서 자세히 설명한다.
- PlayList를 구성하는 각 노드는 단일 곡의 정보를 담고 있으며, 곡을 추가할 때는 BST를 순환하며 데이터를 찾고, 해당 데이터를 바탕으로 새로운 노드를 생성한 뒤 추가한다.
- 곡이 추가될 때 PlayList를 순환하며 동일한 곡이 이미 존재하는지 확인하고, 존재하지 않을 경우에만 새 노드를 삽입한다.
- PlayList는 최대 10곡까지 저장 가능하며, 곡은 추가된 순서에 따라 노드가 생성되어 연결된다.
- PlayList에 곡이 추가되거나 제거될 때마다 Loop 전체의 총 재생 시간이 즉시 갱신된다.



[그림 5] PlayList 예시

□ Functional Requirements

- 출력 포맷은 포맷에 대한 예시일 뿐 실제 출력되는 데이터들과는 차이가 있을 수 있습니다.

명령어	명령어 사용 예 및 기능
LOAD	사용 예) LOAD 텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 MusicQueue 자료구조에 모두 저장한다. 성공적으로 데이터를 불러온 경우 읽은 데이터를 출력하며, 출력 형태는 “가수 이름/노래 제목/재생 시간”이다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. 텍스트 파일: Music_List.txt ***데이터 조건** - 가수 이름과 제목 모두 영문으로 표기해야 한다. - 같은 곡이 중복되어 있는 경우는 없다. - “|”를 구분자로 하여 정보가 저장되어 있다. - 재생 시간은 “x:xx”의 형태로 ‘:’ 문자로 분 초를 구분한다. 출력 포맷 예시) ========LOAD======== PSY/ gangnam style/3:40 … ======================== ========ERROR======== 100 ========================
ADD	사용 예) ADD KDA|pop star|3:40 MusicQueue에 데이터를 직접 추가하기 위한 명령어로, 총 3개의 인자를 추가로 입력한다. 첫 번째 인자부터 가수 이름, 노래 제목, 재생시간을 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력한다. 이때 LOAD 명령어로 Queue에 추가한 데이터들과 비교해 같은 곡이 들어올 경우 에러 코드를 출력한다. 성공적으로 데이터를 추가하였다면 해당 데이터를 출력하며, 출력 형태는 “가수이름/노래제목/재생시간”이다. 출력 포맷 예시)

	========ADD======== KDA/pop star/3:40 ==================== ========ERROR======== 200 ======================
QPOP	사용 예) QPOP MusicQueue에서 데이터를 POP하여 ArtistBST & TitleBST를 구성하는 명령어이다. MusicQueue의 front부터 모든 노드를 POP하여 해당 노드의 데이터로 자료구조를 구축한다. MusicQueue에 데이터가 존재하지 않을 시 에러 코드를 출력한다. BST는 다음과 같은 기준이 적용된다. 1. ArtistBST : 가수의 이름을 기준으로 정렬 - 같은 가수의 작품이 들어오면, 해당 가수 노드의 노래 개수를 count하는 변수를 +1하고, 해당 노드에 데이터를 추가 저장한다. - 이미 존재하는 곡이 입력으로 들어오면 에러 코드를 출력한다. 2. TitleBST : 노래 제목을 기준으로 정렬 - 같은 제목이 들어오면, 해당 제목 노드의 가수 인원수를 count를 하는 변수를 +1하고, 해당 노드에 데이터를 추가 저장한다. - 이미 존재하는 곡이 입력으로 들어오면 에러 코드를 출력한다. 출력 포맷 예시) ========QPOP======== Success ==================== ========ERROR======== 300 ======================
SEARCH	사용 예1) SEARCH ARTIST N.Flying 사용 예2) SEARCH TITLE blue moon 사용 예3) SEARCH SONG N.Flying|blue moon BST에서 노래를 검색하는 명령어이다. 해당 명령어에는 3개의 옵션이 존재한다. 1. ARTIST : 가수 이름으로 ArtistBST에서 검색을 한다. 이때 해당 가수의 모든 곡이 출력 되어야 한다. 2. TITLE : 노래 제목으로 TitleBST에서 검색을 한다. 이때 해당 제목

	의 노래를 음원으로 가지고 있는 가수 정보까지 출력 되어야 한다. 3. SONG : 찾고자 하는 가수의 음원의 정보를 “가수이름|노래제목”의 형식으로 입력하여 검색 후 출력한다. 찾고자 하는 데이터가 검색되지 않는다면 에러 코드를 출력한다. 출력 포맷 예시1) **입력: SEARCH ARTIST Psy** ========SEARCH======== Psy/gangnam style/3:25 Psy/daddy/3:32 ==================== 출력 포맷 예시2) **입력: SEARCH TITLE blue moon** ========SEARCH======== N.Flying/blue moon/3:36 Hyorin/blue moon/3:12 ==================== 출력 포맷 예시3) **입력: SEARCH SONG N.Flying|moonshot** ========SEARCH======== N.Flying/moonshot/3:26 ==================== ========ERROR======== 400 ======================
MAKEPL	사용 예1) MAKEPL ARTIST N.Flying 사용 예2) MAKEPL TITLE blue moon 사용 예3) MAKEPL SONG N.Flying|blue moon MAKEPL 명령어는 BST에 저장된 데이터를 바탕으로 PlayList에 노드를 추가하는 기능을 수행한다. 이 명령어는 총 3가지 옵션을 제공한다. 1. ARTIST : 지정된 가수의 모든 곡을 PlayList에 추가한다. 2. TITLE : 특정 제목을 가진 모든 곡을 추가하며, 동일한 제목을 가진 다른 가수의 곡도 포함한다. 3. SONG : “가수이름|노래제목” 형식으로 단일 곡을 지정하여 PlayList에 추가한다. 추가 과정에서 지정한 곡의 존재 여부는 먼저 BST에 검색되며, 곡이

	존재하지 않을 경우 에러 코드를 출력한다. 반대로 성공적으로 추가되면 현재 PlayList의 총 재생 시간(한 Loop 기준)과 등록된 곡 수가 함께 출력된다. PlayList는 최대 10곡까지만 저장 가능하며, ARTIST 또는 TITLE 옵션으로 여러 곡을 동시에 추가하는 경우에는 추가 전에 PlayList의 남은 공간을 확인해야 한다. 만약 공간이 부족하다면 해당 요청은 처리되지 않고 에러 코드가 출력된다. 출력 형태는 “가수이름/노래제목/재생시간”이며, 등록된 곡 수와, 총 재생 시간 포맷은 아래 출력 포맷에서 확인할 수 있다, 출력 포맷 예시1) **입력: MAKEPL ARTIST Psy** ========MAKEPL======== Psy/gangnam style/3:25 Psy/daddy/3:32 Count : 2 / 10 Time : 6min 57sec ==================== 출력 포맷 예시2) **입력: MAKEPL TITLE blue moon** ========MAKEPL======== Psy/gangnam style/3:25 Psy/daddy/3:32 N.Flying/blue moon/3:36 Hyorin/blue moon/3:12 Count : 4 / 10 Time : 13min 45sec ==================== 출력 포맷 예시3) **입력: MAKEPL SONG KDA|pop star** ========MAKEPL======== Psy/gangnam style/3:25 Psy/daddy/3:32 N.Flying/blue moon/3:36 Hyorin/blue moon/3:12 KDA/pop star/3:40 Count : 5 / 10 Time : 17min 25sec ====================

	========ERROR======== 500 ======================
PRINT	사용 예1) PRINT ARTIST 사용 예2) PRINT TITLE 사용 예3) PRINT LIST PRINT명령어는 BST, LIST에 저장된 데이터를 출력하는 명령어이다. 이 명령어는 총 3개의 옵션이 존재한다. 1. ARTIST : ArtistBST를 중위 순회(in-order) 방식으로 BST를 탐색하여 데이터를 출력한다. 2. TITLE : TitleBST를 중위 순회(in-order) 방식으로 BST를 탐색하여 데이터를 출력한다. 3. LIST : PlayList를 한바퀴를 순회하며 모든 노드의 정보를 출력한다. 출력되는 데이터는 출력 포맷 예시에서 확인할 수 있다. 출력 포맷 예시1) **입력: PRINT ARTIST** ========PRINT======== ArtistBST Hyorin/blue moon/3:12 KDA/pop star/3:40 N.Flying/blue moon/3:36 N.Flying/moonshot/3:26 N.Flying/songbird/3:46 N.Flying/rooftop/3:19 N.Flying/everlasting/4:05 N.Flying/oh really./3:10 N.Flying/into you/3:45 N.Flying/star/3:55 ==================== 출력 포맷 예시2) **입력: PRINT TITLE** ========Print======== TitleBST N.Flying/blue moon/3:36 Hyorin/blue moon/3:12 N.Flying/everlasting/4:05 N.Flying/into you/3:45 N.Flying/moonshot/3:26

	N.Flying/oh really./3:10 KDA/pop star/3:40 N.Flying/rooftop/3:19 N.Flying/songbird/3:46 N.Flying/star/3:55 ========== 출력 포맷 예시3) **입력: PRINT LIST** ========Print======== N.Flying/blue moon/3:36 Hyorin/blue moon/3:12 KDA/pop star/3:40 N.Flying/songbird/3:46 Count : 4 / 10 Time : 14min 14sec ========== ========ERROR======== 600 ==========
DELETE	사용 예1) DELETE ARTIST Psy 사용 예2) DELETE TITLE blue moon 사용 예3) DELETE LIST KDA|pop star 사용 예4) DELETE SONG N.Flying|songbird DELETE 명령어는 입력된 데이터를 기반으로 해당 곡 또는 가수의 정보를 다양한 자료구조에서 삭제하는 기능을 수행한다. 이 명령어는 총 4개의 옵션을 제공한다. 1. ARTIST : 지정된 가수의 모든 데이터를 삭제한다. 이 경우 해당 가수의 모든 곡이 ArtistBST, TitleBST, 그리고 PlayList에서 제거된다. 2. TITLE : 입력된 노래 제목과 일치하는 모든 곡 데이터를 삭제한다. 동일한 제목을 보유한 다른 가수의 곡 역시 함께 삭제되며, ArtistBST, TitleBST, PlayList에서 모두 제거된다. 3. LIST : 입력된 곡을 PlayList 내에서만 삭제한다. 이때 입력 데이터는 “가수이름|노래제목” 형식을 따른다. BST에는 영향이 없으며, PlayList에서만 해당 노드사 제거된다. 이때 List의 등록된 곡 수와 총 재생시간이 업데이트된다. 4. SONG : 입력된 곡을 단일 데이터로 간주하여 BST(ArtistBST, TitleBST)와 PlayList에서 모두 삭제한다.

	delete 작업이 성공적으로 수행되면 다음과 같이 출력 형식을 따른다. 출력 포맷 예시) ========DELETE======== Success ====================== ========ERROR======== 700 ======================
EXIT	사용 예) EXIT 프로그램상의 메모리를 해제하며, 프로그램을 종료한다. 출력 포맷 예시) =======EXIT======== Success ==================

□ 동작 별 에러 코드

동작	에러 코드
LOAD	100
ADD	200
QPOP	300
SEARCH	400
MAKEPL	500
PRINT	600
DELETE	700
잘못된 명령어	1000

□ Requirements in implementation

- ✓ 모든 명령어는 command.txt에 저장하여 순차적으로 읽고 처리한다.
- ✓ 모든 명령어는 반드시 대문자로 입력한다.
- ✓ 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력받으면 에러 코드를 출력한다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.
- ✓ 출력은 “출력 포맷”을 반드시 따라 한다.
- ✓ log.txt 파일에 출력 결과를 반드시 저장한다.
  - log.txt가 이미 존재할 경우 텍스트 파일 가장 뒤에 이어서 추가로 저장한다.

□ 구현 시 반드시 정의해야하는 Class

- ✓ MusicQueueNode - 음원 정보 Queue 노드 클래스
- ✓ MusicQueue - 음원 정보 Queue 클래스
- ✓ ArtistBSTNode - 가수 이름을 기준으로 정렬되는 BST의 노드 클래스
- ✓ ArtistBST - 가수 이름을 기준으로 정렬되는 BST 클래스
- ✓ TitleBSTNode - 노래 제목을 기준으로 정렬되는 BST의 노드 클래스
- ✓ TitleBST - 노래 제목을 기준으로 정렬되는 BST 클래스
- ✓ PlayListNode - 순환 연결 리스트를 구성하는 노드 클래스
- ✓ PlayList - 순환 연결 리스트 클래스
- ✓ Manager - Manager 클래스
  - 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

□ Files

- ✓ Music_List.txt : 프로그램에 추가할 음원 데이터가 저장된 파일
- ✓ command.txt : 프로그램을 동작시키는 명령어들을 저장하고 있는 파일
- ✓ log.txt : 프로그램 출력 결과를 모두 저장하고 있는 파일

□ 제한사항 및 구현 시 유의사항

✓ 반드시 제공되는 코드(github 주소 참고)를 이용하여 구현하며 작성된 소스 파일의 이름과 클래스와 함수 이름 및 형태를 임의로 변경하지 않는다.
✓ 클래스의 함수 및 변수는 자유롭게 추가 구현이 가능하다.
✓ 제시된 Class를 각 기능에 알맞게 모두 사용한다.
✓ 프로그램 구조에 대한 디자인이 최대한 간결하도록 고려하여 설계한다.
✓ 채점 시 코드를 수정해야 하는 일이 없도록 한다.
✓ 주석은 반드시 영어로 작성한다. (한글로 작성하거나 없으면 감점)
✓ 프로그램은 반드시 리눅스(Ubuntu 22.04)에서 동작해야 한다. (컴파일 에러 발생 시 감점)
- 제공되는 Makefile을 사용하여 테스트하도록 한다.
✓ **인터넷에서 공유된 코드나 다른 학생이 작성한 코드를 절대 카피하지 않도록 하며, 적발시 전체 프로젝트 0점 처리됨을 명시한다.**

□ 채점 기준

✓ 코드

채점 기준	점수
LOAD 명령어가 정상 동작하는가?	1
ADD 명령어가 정상 동작하는가?	1
QPOP 명령어가 정상 동작하는가?	2
SEARCH 명령어가 정상 동작하는가?	1
MAKEPL 명령어가 정상 동작하는가?	1
PRINT 명령어가 정상 동작하는가?	1
DELETE 명령어가 정상 작동하는가?	3
총합	**10**

– 채점 기준 이외에도 조건 미달 시 감점 (linux 컴파일 에러, 파일 입출력 X, 주석 미흡 등)

✓ 보고서

채점 기준	점수
Introduction을 잘 작성하였는가?	1
Flowchart를 잘 작성하였는가?	2.5
Algorithm을 잘 작성하였는가?	3
Result Screen을 잘 작성하였는가?	2.5
Consideration을 잘 작성하였는가?	1
총합	**10**

✓ 최종 점수는 (코드 점수 x 보고서 점수)로 계산됩니다.

□ 제출기한 및 제출방법

✓ 제출기한

- 2025년 10월 12일 일요일 23:59:59까지 클라스(KLAS)에 제출
  (추가 제출: 2025년 10월 13일 월요일 00:59:59 까지, 10% 감점)

✓ 제출방법

- 소스코드(Makefile과 텍스트 파일 제외)와 보고서 파일(pdf)을 함께 압축하여 제출
  - 확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음 (제출시 감점)
  - 보고서 파일 확장자가 pdf가 아닐 시 감점

✓ 제출 형식

- 학번_DS_project1.tar.gz (ex. 2020XXXXXX_DS_project1.tar.gz)
  - 제출 형식을 지키지 않은 경우 감점

✓ 보고서 작성 형식 및 제출 방법

- Introduction : 프로젝트 내용에 대한 설명
- Flowchart : 설계한 프로젝트의 플로우차트를 그리고 설명
- Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명
- Result Screen : 모든 명령어에 대해 결과화면을 캡처하고 동작을 설명
- Consideration : 고찰 작성
  - 위의 각 항목을 모두 포함하여 작성
  - 보고서 내용은 한글로 작성
  - 보고서에는 소스코드를 포함하지 않음