데이터 구조 설계/ 실습

2025_Kwangwoon_DS_Project_1

1. 서론

이번 데이터 구조 설계 및 실습 과제에서는 QUEUE/BST/LINKEDLIST 를 사용하여 음악 관리프로그램을 제작하게 되었습니다.
제안서를 바탕으로 데이터의 흐름은 다음과 같습니다.
Music_List.txt → Music Queue → ArtistBST / TitleBST → PlayList LinkedList
해당 과정은 command.txt에 있는 명령들을 한줄 씩 읽어와 수행되며, 모든 수행은 Manager에서 수행합니다.
데이터는 가수/ 제목/ 시간 으로 구성되어 있습니다.

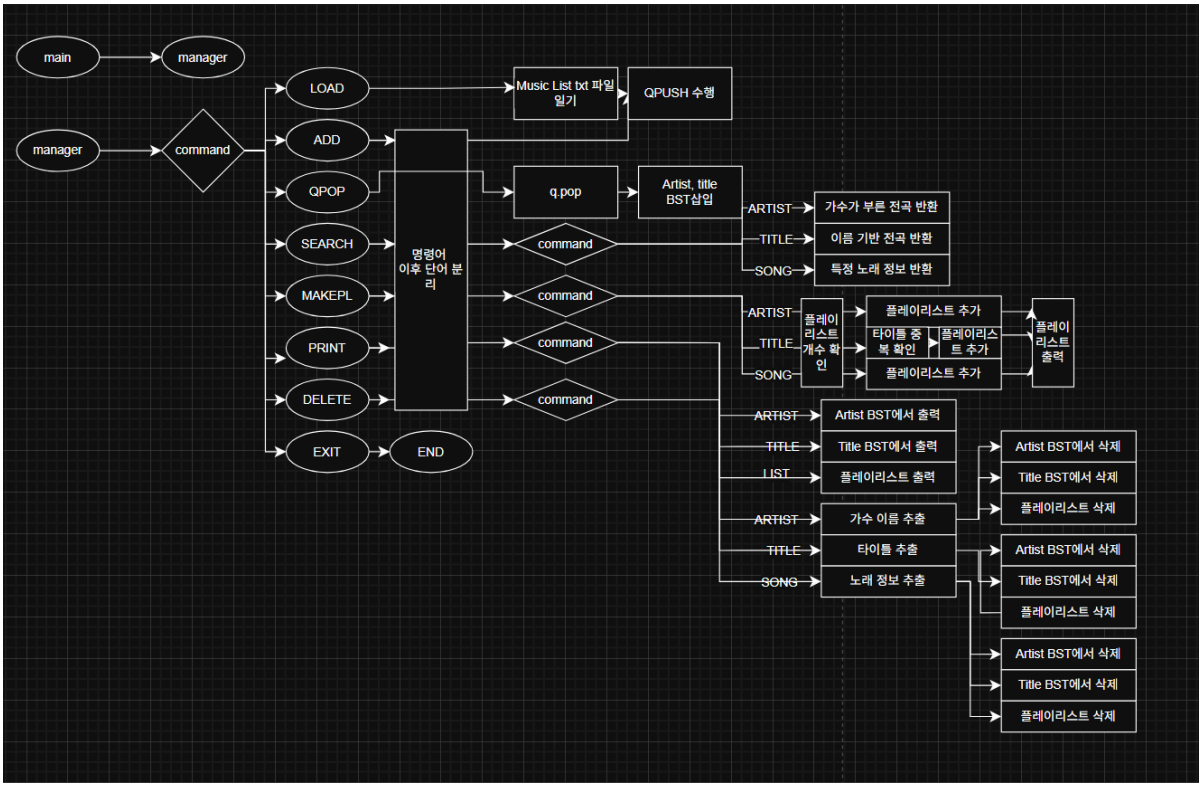
각각은 Manager 파일에서 다음과 같이 연결되어 있습니다.
Music Queue
LOAD와 PUSH를 기반으로한 입력
QPOP을 기반으로한 출력

ArtistBST, TitleBST
QPOP을 기반으로한 입력
PRINT와 MAKEPL를 기반으로한 출력

PlayList LinkedList
MAKEPL를 기반으로한 입력
PRINT를 기반으로한 출력

해당 과제는 1.각 자료구조를 구현하고 2.다음과 같은 연결 관계를 구현하여 일련의 command를 수행하는 것 입니다.

2. 플로우차트



3. 알고리즘

Manager

RUN
커맨드 파일을 읽기 시작 로그파일 작성 시작
명령에 따라서 함수 호출
명령어의 이후 내용을 필요에 따라 함수에 반환하며 호출

명령어는 LOAD, ADD, QPOP, SEARCH, MAKEPL, PRINT, DELETE, EXIT으로 구분되어 있음

LOAD() 함수
fin을 "Music_List.txt" 파일로 열기

파일 끝까지 한 줄(line)씩 반복:
  line에서 '|' 기준으로 artist, title, duration 파싱
  Queue에 Push

ADD(args) 함수
args에서 '|' 기준으로 artist, title, duration 파싱
만약 q.exist(artist, title)가 참이면 (중복이면):

    200에러 반환
아니면:
    Queue에 Push

QPOP() 함수
만약 Queue 가 비어있다면:
    flog에 에러(300) 기록 후 종료

q가 빌 때까지 반복:
    QPOP수행
    ArtistBST에 데이터 입력
    TitleBST에 데이터 입력
    node 메모리 해제

SEARCH(args) 함수
args에서 command와 query 분리
만약 command가 "ARTIST"이면:
    ArtistBST의 search함수 호출
만약 command가 "TITLE"이면:
    TitleBST의 search함수 호출
만약 command가 "SONG"이면:
    query에서 '|' 기준으로 artist와 title 분리
    ARtistBST에서 searchSong호출 및 시간 반환 + 출력

MAKEPL(args) 함수
args에서 type과 나머지 인자 분리

만약 type이 "ARTIST"이면:
    추가할 곡 수 = searchCount로 확인
    추가했을 때 곡이 10개 초과면:
        에러 반환
    아니면, ab.MAKEPL 호출
    플레이 리스트 출력

만약 type이 “Title"이면:
    추가할 곡 수 = searchCount로 확인
    추가했을 때 곡이 10개 초과면:
        에러 반환
    아니면, ab.MAKEPL 호출

만약 type이 “Title"이면:
    pl.getcount로 1곡 추가시 10개 이상인지 확인
    아니면, pl.insert_node 호출

PRINT(args) 함수
type = args
만약 type이 "ARTIST"이면:
  ab.print(flog) 호출
만약 type이 "TITLE"이면:
  tb.print(flog) 호출
만약 type이 "LIST"이면:
  pl_output = pl.print() 호출

DELETE(args) 함수
args에서 type과 query 분리
deleted_flag = false

만약 type이 "ARTIST"이면:
  같은 가수를 가진 노래들을 검색
    노래에 대한 각 가수에 대해
    ArtistBST, TitleBST, PlayList 를 다 삭제함

만약 type이 "ARTIST"이면:
    가수의 각 노래에 대해
    ArtistBST, TitleBST, PlayList 를 다 삭제함
만약 type이 “SONG"이면:
    노래와 가수를 지목하여
    ArtistBST, TitleBST, PlayList 를 다 삭제함

EXIT() 함수
flog에 "Success" 출력
run 함수에서 이 함수 호출 후 break가 실행됨

MUSIC QUEUE, Node

Node 부분
insert(아티스트,타이틀,런타임):
      노드에 정보들을 입력
exist()
      노드에 해당 음악 정보가 존재하는지 확인
MusicQueue::생성자():
    head를 null로 설정
    tail을 null로 설정
    size를 0으로 설정

MusicQueue::소멸자():
    큐가 비어있지 않은 동안 반복:
      temp 변수 = pop() 함수 호출
      temp 변수 메모리에서 해제

MusicQueue::empty():
    만약 size가 0이면:
        참(true)을 반환
    아니면:
        거짓(false)을 반환

MusicQueue::exist(아티스트, 제목):
    current 변수 = head로 설정
    current가 null이 아닌 동안 반복:
        만약 current 노드에 해당 아티스트와 제목이 존재하면:
            참(true)을 반환
        current를 다음 노드로 이동
    반복이 끝나면 거짓(false)을 반환

MusicQueue::push(아티스트, 제목, 재생시간, 로그파일):
    만약 size가 max_size와 같으면:
        로그파일에 "Error: 100" 기록
        프로그램 종료
        거짓(false)을 반환
    아니면:큐의 크기가 0~99일 때
        새로운 노드(newNode)를 생성
        newNode에 아티스트, 제목, 재생시간 정보를 삽입

        만약 큐가 비어있으면:
            head를 newNode로 설정
            tail을 newNode로 설정
        아니면:
            현재 tail의 다음 노드를 newNode로 설정
            newNode의 이전 노드를 현재 tail로 설정
            tail을 newNode로 설정

        size를 1 증가시킴
        참(true)을 반환

MusicQueue::pop():
    만약 큐가 비어있으면:
        에러 300 반환 및 프로그램 종료
    temp 변수 = head
    head = head의 다음 노드
    size를 1 감소시킴

    만약 큐가 비어있지 않다면:
        새로운 head의 이전 노드를 null로 설정

아니면 (큐가 완전히 비었다면):
tail을 null로 설정

temp의 다음 노드를 null로 설정
temp의 이전 노드를 null로 설정

temp를 반환

MusicQueue::front():
head를 반환

ArtistBST, NODE

set(아티스트, 제목, 재생시간):
node 하나에 값을 저장함
run_time의 경우 ", "으로 전달받은 재생시간을 덧붙임
나의 title 리스트에 전달받은 제목을 추가
나의 count를 1 증가시킴

---

delete_title(삭제할_제목):
반복 순회하며 내부에서

만약 song_index가 -1이라면:
함수 종료
나의 title 리스트에서 song_index 위치의 항목을 삭제
기존_재생시간_문자열 = 나의 run_time
새_재생시간_문자열 = "" (빈 문자열로 초기화)
현재_인덱스 = 0
기존_재생시간_문자열을 쉼표(',') 기준으로 분리하여 각 부분(시간)에 대해 반복:
만약 시간의 첫 글자가 공백이면:
시간에서 첫 글자인 공백을 제거

만약 현재_인덱스가 song_index와 다르다면:
만약 새_재생시간_문자열이 비어있지 않다면:
새_재생시간_문자열에 ", "를 덧붙임
새_재생시간_문자열에 시간을 덧붙임
현재_인덱스를 1 증가시킴
나의 run_time = 새_재생시간_문자열
나의 count를 1 감소시킴

---

getSongCount():

나의 count를 반환

ArtistBST

destroy_recursive(파괴할_노드):
    만약 파괴할_노드가 null이면, 함수를 종료

    파괴할_노드의 왼쪽 자식 노드에 대해 재귀적으로 파괴 함수를 호출
    파괴할_노드의 오른쪽 자식 노드에 대해 재귀적으로 파괴 함수를 호출

    파괴할_노드를 메모리에서 해제

---

insert(아티스트, 제목, 재생시간):
    전달받은 정보로 새로운 노드(newNode)를 생성

    만약 root가 null이면,
      root를 newNode로 설정하고 함수를 종료

    탐색용 변수(current)를 root로 설정
    탐색용 변수가 null이 아닐 때까지 반복:
      만약 전달받은 아티스트가 current의 아티스트와 같다면,
        current 노드에 새 노래 정보(제목, 재생시간)를 추가
        불필요해진 newNode를 메모리에서 해제하고 함수를 종료

      만약 전달받은 아티스트가 current의 아티스트보다 작다면,
        만약 current의 왼쪽 자식이 null이면,
          current의 왼쪽 자식을 newNode로 설정하고 함수를 종료
        탐색용 변수를 왼쪽 자식으로 이동
      아니면 (더 크다면),
        만약 current의 오른쪽 자식이 null이면,
          current의 오른쪽 자식을 newNode로 설정하고 함수를 종료
        탐색용 변수를 오른쪽 자식으로 이동

---

search(아티스트, 로그파일):
    탐색용 변수(current)를 root로 설정
    탐색용 변수가 null이 아닐 때까지 반복:
      만약 전달받은 아티스트가 current의 아티스트와 같다면,
        current 노드의 전체 재생시간 문자열을 가져옴

        current 노드의 노래 제목 목록을 하나씩 순회:
          전체 재생시간 문자열에서 현재 노래 제목에 맞는 재생시간 부분을 찾음
          "아티스트/제목/재생시간" 형식으로 로그파일에 기록

        작업을 마쳤으므로 함수를 종료

아니면, 만약 전달받은 아티스트가 current의 아티스트보다 작다면,
탐색용 변수를 왼쪽 자식으로 이동
아니면 (더 크다면),
탐색용 변수를 오른쪽 자식으로 이동

함수를 종료

---

searchCount(아티스트):
탐색용 변수(current)를 root로 설정
탐색용 변수가 null이 아닐 때까지 반복:
만약 전달받은 아티스트가 current의 아티스트와 같다면,
current 노드의 노래 개수를 반환

아니면, 만약 전달받은 아티스트가 current의 아티스트보다 작다면,
탐색용 변수를 왼쪽 자식으로 이동
아니면 (더 크다면),
탐색용 변수를 오른쪽 자식으로 이동

0을 반환

MAKEPL(아티스트, 플레이리스트):
탐색용 변수(current)를 root로 설정
탐색용 변수가 null이 아닐 때까지 반복:
만약 전달받은 아티스트가 current의 아티스트와 같다면,
current 노드의 노래 제목 목록을 하나씩 순회:
전체 재생시간 문자열에서 현재 노래 제목에 맞는 재생시간 부분을 찾음
찾아낸 정보(아티스트, 제목, 재생시간)를 플레이리스트에 삽입

작업을 마쳤으므로 함수를 종료

아니면, 만약 전달받은 아티스트가 current의 아티스트보다 작다면,
탐색용 변수를 왼쪽 자식으로 이동
아니면 (더 크다면),
탐색용 변수를 오른쪽 자식으로 이동

(반복이 끝났다면 가수를 못 찾은 것이므로) 함수를 종료

---

print_recursive(출력할_노드, 로그파일):
만약 출력할 노드가 null이면, 함수를 종료

출력할_노드의 왼쪽 자식에 대해 재귀적으로 출력 함수를 호출

출력할 노드의 노래 제목 목록을 하나씩 순회:
전체 재생시간 문자열에서 현재 노래 제목에 맞는 재생시간 부분을 찾음
"아티스트/제목/재생시간" 형식으로 로그파일에 기록

    출력할_노드의 오른쪽 자식에 대해 재귀적으로 출력 함수를 호출

  print(로그파일):
    root 노드를 시작으로 재귀적 출력 함수를 호출

---

  delete_node(아티스트, 제목, 로그파일):
    만약 아티스트가 빈 문자열이면,
      제목으로 삭제하는 함수(delete_title)를 호출
    아니면, 만약 제목이 빈 문자열이면,
      아티스트로 삭제하는 함수(delete_artist)를 호출
    아니면,
      특정 노래로 삭제하는 함수(delete_song)를 호출

---

  delete_title(제목, 로그파일):
    root 노드를 시작으로 재귀적 제목 삭제 함수를 호출

  delete_title_recursive(현재_노드, 제목, 로그파일):
    만약 현재_노드가 null이면, 함수를 종료
    현재_노드에서 해당 제목의 노래를 삭제
    현재_노드의 왼쪽 자식에 대해 재귀적 제목 삭제 함수를 호출
    현재_노드의 오른쪽 자식에 대해 재귀적 제목 삭제 함수를 호출

  delete_artist(아티스트, 로그파일):
    만약 root가 null이거나 아티스트가 빈 문자열이면,
      로그파일에 에러를 기록하고 함수를 종료

    삭제할 노드(current)를 root로, 그 부모 노드(parent)를 null로 설정
    current가 null이 아니면서, current의 아티스트가 전달받은 아티스트와 다를 때까지 반복:
      parent를 current로 설정
      만약 전달받은 아티스트가 current의 아티스트보다 작다면,
        current를 왼쪽 자식으로 이동
      아니면,
        current를 오른쪽 자식으로 이동

    만약 current가 null이면 (아티스트를 찾지 못했으면),
      로그파일에 에러를 기록하고 함수를 종료

    만약 current의 자식이 둘 다 없으면 (리프 노드),
      만약 current가 root이면, root를 null로 설정
      아니면, 만약 parent의 왼쪽 자식이 current이면, parent의 왼쪽 자식을 null로 설정
      아니면, parent의 오른쪽 자식을 null로 설정
      current를 메모리에서 해제

아니면, 만약 current의 자식이 하나만 있으면,
자식 노드(child)를 찾음 (왼쪽 또는 오른쪽)
만약 current가 root이면, root를 child로 설정
아니면, 만약 parent의 왼쪽 자식이 current이면, parent의 왼쪽 자식을 child로 설정
아니면, parent의 오른쪽 자식을 child로 설정
current를 메모리에서 해제

아니면 (current의 자식이 둘 다 있으면),
오른쪽 서브트리에서 가장 작은 노드를 후계자(successor)로 찾음
current의 데이터를 후계자의 데이터로 덮어씀

delete_song(아티스트, 제목, 로그파일):
만약 root가 null이거나 아티스트 또는 제목이 빈 문자열이면,
로그파일에 에러를 기록하고 함수를 종료

삭제할 노드가 포함된 아티스트 노드(current)와 그 부모(parent)를 찾음

만약 아티스트 노드(current)를 찾지 못했으면,
로그파일에 에러를 기록하고 함수를 종료

current 노드의 노래 목록에서 삭제할 제목의 위치(song_index)를 찾음

만약 노래 제목을 찾지 못했으면,
로그파일에 에러를 기록하고 함수를 종료

만약 current 노드의 노래 개수가 1보다 크면,
current의 제목 리스트에서 song_index 위치의 항목을 제거
전체 재생시간 문자열을 분리하여, song_index 위치의 시간을 제외하고 다시 합침
current의 노래 개수를 1 감소

아니면 (마지막 남은 한 곡을 삭제하는 경우),
만약 current의 자식이 둘 다 없으면 (리프 노드),
부모 노드와의 연결을 끊고 current를 메모리에서 해제
아니면, 만약 current의 자식이 하나만 있으면,
부모 노드를 current의 자식 노드와 연결하고 current를 메모리에서 해제
아니면 (current의 자식이 둘 다 있으면),
오른쪽 서브트리에서 가장 작은 노드를 후계자(successor)로 찾음
current의 데이터를 후계자의 데이터로 덮어쓰고, 후계자 노드를 삭제

---

searchSong(아티스트, 제목):
탐색용 변수(current)를 root로 설정
탐색용 변수가 null이 아닐 때까지 반복:
만약 전달받은 아티스트가 current의 아티스트와 같다면,
current의 제목 리스트를 순회하며 전달받은 제목의 위치(i)를 찾음

만약 제목을 찾았다면,
전체 재생시간 문자열을 분리하여 i번째 재생시간을 찾아 반환
못 찾았으면 빈 문자열을 반환

아니면, 만-약 전달받은 아티스트가 current의 아티스트보다 작다면:
탐색용 변수를 왼쪽 자식으로 이동
아니면:
탐색용 변수를 오른쪽 자식으로 이동

빈 문자열을 반환

TitleBST,NODE

set(제목, 아티스트, 재생시간):
만약 나의 title 멤버 변수가 비어있다면,
나의 title = 전달받은 제목
나의 run_time = 전달받은 재생시간
아니면,
나의 run_time에 ","와 전달받은 재생시간을 덧붙임

나의 artist 리스트에 전달받은 아티스트를 추가
나의 count를 1 증가시킴

---

delete_artist(삭제할_아티스트):
삭제할 아티스트의 위치(artist_index)를 찾기 위해 나의 artist 리스트를 순회
만약 아티스트를 찾지 못했으면 함수를 종료

artist 리스트에서 찾은 위치(artist_index)의 아티스트를 삭제
run_time 문자열을 쉼표 기준으로 분해하여, artist_index 위치의 재생 시간을 제외하고 다시 하나의 문자열로 합침
나의 run_time을 새로 합쳐진 문자열로 갱신
count를 1 감소

함수 getSongCount():
나의 count를 반환

TitleBST

destroy_recursive(파괴할_노드):
만약 파괴할_노드가 null이면 함수를 종료
파괴할_노드의 왼쪽 자식에 대해 재귀적으로 파괴 함수를 호출
파괴할_노드의 오른쪽 자식에 대해 재귀적으로 파괴 함수를 호출
파괴할_노드를 메모리에서 해제

---

insert(제목, 아티스트, 재생시간):
  전달받은 정보로 새로운 노드(newNode)를 생성
  만약 root가 null이면, root를 newNode로 설정하고 종료

  탐색용 변수(current)를 root로 설정하여 반복:
    만약 전달받은 제목이 current의 제목과 같다면,
      current 노드에 새 아티스트 정보(아티스트, 재생시간)를 추가
      불필요해진 newNode를 메모리에서 해제하고 종료

    만약 전달받은 제목이 current의 제목보다 작다면,
      만약 current의 왼쪽 자식이 비어있으면, 왼쪽 자식으로 newNode를 연결하고 종료
      아니면, 탐색용 변수를 왼쪽 자식으로 이동
    아니면 (더 크다면),
      만약 current의 오른쪽 자식이 비어있으면, 오른쪽 자식으로 newNode를 연결하고 종료
      아니면, 탐색용 변수를 오른쪽 자식으로 이동

---

search(제목, 로그파일):
  탐색용 변수(current)를 root로 설정
  current가 null이 아닐 때까지 반복:
    만약 전달받은 제목이 current의 제목과 같다면,
      current 노드의 가수 목록을 순회하며, 각 가수에 맞는 재생시간을 파싱하여 찾음
      찾아낸 "아티스트/제목/재생시간" 정보를 로그파일에 기록
      작업을 마쳤으므로 함수를 종료

    아니면, 만약 전달받은 제목이 current의 제목보다 작다면,
      탐색용 변수를 왼쪽 자식으로 이동
    아니면,
      탐색용 변수를 오른쪽 자식으로 이동
  (반복이 끝났다면 노래를 못 찾은 것이므로 함수 종료)

---

searchCount(제목):
  탐색용 변수(current)를 root로 설정
  current가 null이 아닐 때까지 반복:
    만약 전달받은 제목이 current의 제목과 같다면,
      current 노드의 가수 수(count)를 반환

    아니면, 만약 전달받은 제목이 current의 제목보다 작다면,
      탐색용 변수를 왼쪽 자식으로 이동
    아니면,
      탐색용 변수를 오른쪽 자식으로 이동
  (반복이 끝났다면 노래를 못 찾은 것이므로) 0을 반환

MAKEPL(제목, 플레이리스트):
  탐색용 변수(current)를 root로 설정
  current가 null이 아닐 때까지 반복:
    만약 전달받은 제목이 current의 제목과 같다면,
      current 노드의 가수 목록을 순회하며, 각 가수에 맞는 재생시간을 파싱하여 찾음
      찾아낸 정보를 플레이리스트에 삽입
      작업을 마쳤으므로 함수를 종료

    아니면, 만약 전달받은 제목이 current의 제목보다 작다면,
      탐색용 변수를 왼쪽 자식으로 이동
    아니면,
      탐색용 변수를 오른쪽 자식으로 이동
  (반복이 끝났다면 노래를 못 찾은 것이므로 함수 종료)

---

print_recursive(출력할_노드, 로그파일):
  만약 출력할_노드가 null이면 함수 종료
  출력할_노드의 왼쪽 자식에 대해 재귀적으로 출력 함수 호출
  현재 노드의 모든 가수 정보를 재생시간과 함께 로그파일에 기록
  출력할_노드의 오른쪽 자식에 대해 재귀적으로 출력 함수 호출

print(로그파일):
  root 노드를 시작으로 재귀적 출력 함수를 호출

---

delete_node(아티스트, 제목, 로그파일):
  만약 제목이 비어있으면, 아티스트로 삭제하는 함수(delete_artist) 호출
  아니면, 만약 아티스트가 비어있으면, 제목으로 삭제하는 함수(delete_title) 호출
  아니면, 특정 노래로 삭제하는 함수(delete_song) 호출

delete_title(제목, 로그파일):
  삭제할 노드(current)를 root로, 그 부모(parent)를 null로 설정
  current가 null이 아니면서 current의 제목이 전달받은 제목과 다를 때까지 반복:
    parent를 current로 갱신
    만약 전달받은 제목이 current의 제목보다 작다면, current를 왼쪽 자식으로 이동
    아니면, current를 오른쪽 자식으로 이동

  만약 current가 null이면 (노드를 찾지 못했으면), 로그파일에 에러를 기록하고 종료

  만약 current의 자식이 둘 다 없으면 (리프 노드),
    만약 current가 root이면, root를 null로 설정
    아니면, 만약 parent의 왼쪽 자식이 current이면, parent의 왼쪽 자식을 null로 설정
    아니면, parent의 오른쪽 자식을 null로 설정
    current를 메모리에서 해제

  아니면, 만약 current의 자식이 하나만 있으면,
    자식 노드(child)를 찾아서, parent와 child를 연결하고 current를 메모리에서 해제

아니면 (자식이 둘 다 있으면),
오른쪽 서브트리에서 가장 작은 노드를 후계자(successor)로 찾음
current의 데이터를 후계자의 데이터로 덮어쓰고, 원래의 후계자 노드를 삭제하는 절차를 수행

delete_artist(아티스트, 로그파일):
root 노드를 시작으로 재귀적 아티스트 삭제 함수를 호출

delete_artist_recursive(현재_노드, 아티스트, 로그파일):
만약 현재_노드가 null이면 함수 종료
왼쪽과 오른쪽 자식에 대해 재귀적으로 아티스트 삭제 함수를 호출
현재_노드에서 해당 아티스트 정보를 삭제

delete_song(아티스트, 제목, 로그파일):
삭제할 곡이 있는 제목 노드(current)와 그 부모(parent)를 찾음
만약 노드를 찾지 못했으면, 에러를 기록하고 종료

찾은 노드 내에서 삭제할 아티스트의 위치(index)를 찾음
만약 아티스트를 찾지 못했으면, 에러를 기록하고 종료

만약 노드에 여러 가수가 있다면,
가수 리스트와 재생시간 정보에서 해당 아티스트 부분만 삭제
아니면 (마지막 남은 가수라면),
delete_title 함수와 동일한 절차를 통해 노드 전체를 트리에서 삭제

PlayList, Node

PlayListNode(아티스트,제목, 재생시간):
노드에 해당 값 초기화 생성자

PlayList

소멸자():
만약 리스트가 비어있으면 함수를 종료
탐색용 변수(current)를 head의 다음 노드에서 시작
current가 head가 아닐 때까지 반복:
삭제할 노드(temp)를 current로 지정
current를 다음 노드로 이동
temp를 메모리에서 해제
마지막으로 남은 head 노드를 메모리에서 해제

---

insert_node(아티스트, 제목, 재생시간):

만약 리스트가 가득 찼거나, 동일한 노래가 이미 존재하면 함수를 종료

재생시간 문자열을 초 단위 정수로 변환
전달받은 정보로 새로운 노드(newNode)를 생성

만약 리스트가 비어있으면,
head를 newNode로 설정하고, head의 이전/다음 노드가 자기 자신을 가리키도록 설정
아니면 (리스트에 노드가 있으면),
리스트의 마지막 노드(tail)를 찾음 (head의 이전 노드)
tail의 다음을 newNode로, newNode의 이전을 tail로 연결
newNode의 다음을 head로, head의 이전을 newNode로 연결

count를 1 증가시키고, time에 변환된 재생시간(초)을 더함

---

delete_node(아티스트, 제목, 로그파일):
만약 리스트가 비어있거나, 삭제 조건(아티스트, 제목)이 모두 비어있으면 함수를 종료

삭제 작업 전의 노드 개수(initial_count)를 기억
탐색용 변수(current)를 head에서 시작

기억해둔 개수(initial_count)만큼 반복:
순회가 끊기지 않도록, 다음 방문할 노드(next_node)를 미리 저장

현재 노드가 주어진 아티스트/제목 조건과 일치하는지 확인

만약 일치한다면 (삭제 대상이라면),
time에서 현재 노드의 재생시간을 빼고, count를 1 감소

만약 count가 0이 되면 (마지막 노드를 삭제한 경우),
current를 메모리에서 해제하고, head를 null로 설정한 뒤, 반복을 즉시 중단
아니면,
current의 이전 노드와 다음 노드를 서로 직접 연결하여 리스트에서 current를 제외
만약 삭제한 노드가 head였다면, 미리 저장해둔 next_node를 새로운 head로 지정
current를 메모리에서 해제

탐색용 변수(current)를 미리 저장해둔 next_node로 이동하여 순회를 계속

---

empty():
count가 0이면 참(true)을 반환

---

full():
count가 10 이상이면 참(true)을 반환

---

getCount():
    count를 반환

---

exist(아티스트, 제목):
    만약 리스트가 비어있으면 거짓(false)을 반환

    탐색용 변수(current)를 head에서 시작
    do-while 루프를 사용하여 리스트를 한 바퀴 순회:
        현재 노드의 정보가 주어진 아티스트/제목 조건과 모두 일치하면, 참(true)을 반환
        current를 다음 노드로 이동

    (순회가 끝났다면 일치하는 노래가 없는 것이므로) 거짓(false)을 반환

---

print(로그파일):
    만약 리스트가 비어있으면, 로그파일에 에러를 기록하고 종료

    결과 문자열(result_str)을 초기화
    do-while 루프를 사용하여 리스트를 한 바퀴 순회:
        현재 노드의 정보를 "아티스트/제목/재생시간" 형식으로 변환하여 결과 문자열에 추가

    전체 곡 수와 총 재생 시간을 형식에 맞게 문자열로 만들어 결과 문자열에 덧붙임
    최종 결과 문자열을 반환

---

run_time():
    총 재생 시간(time)을 반환

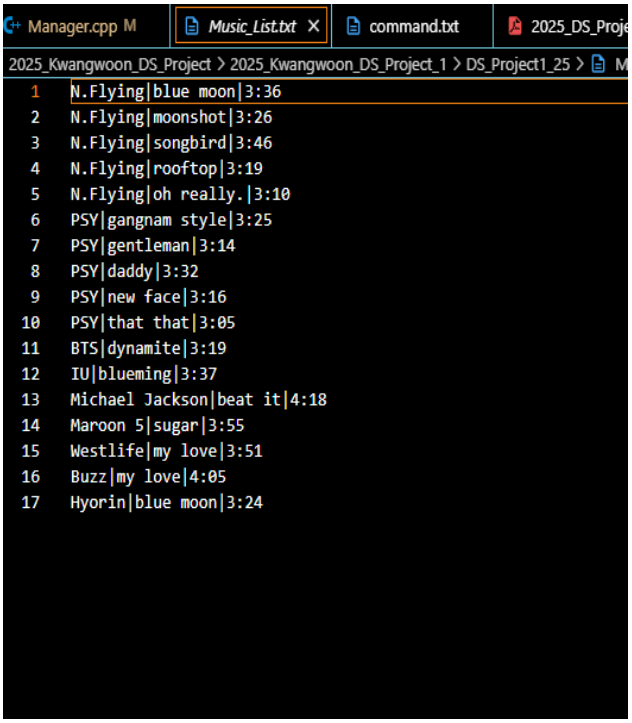
---

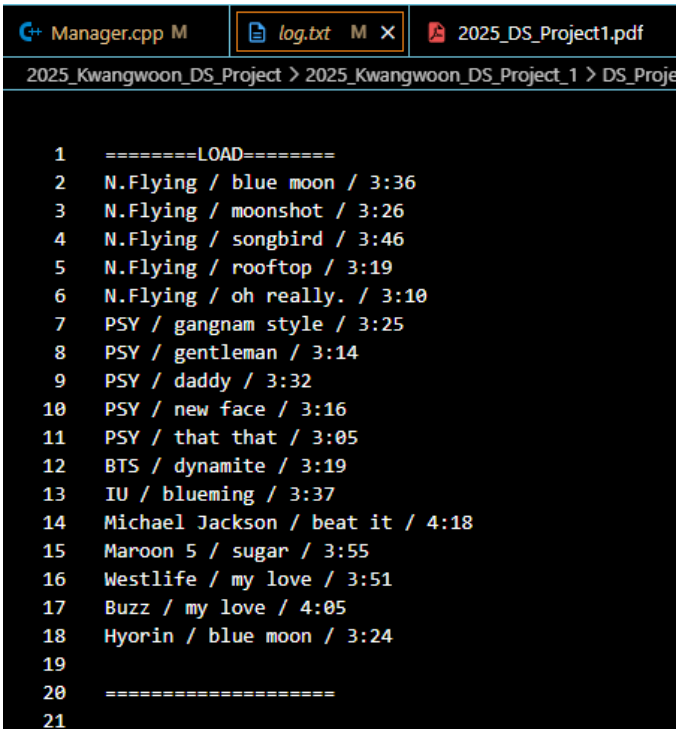
stringToSec(재생시간_문자열):
    문자열에서 ':'의 위치를 찾음
    ':' 앞부분을 분으로, 뒷부분을 초로 변환
    분을 초로 환산하여 초와 더한 총 시간을 반환

---

secToString(총_초):
    총 초를 60으로 나눈 몫을 분으로, 나머지를 초로 계산
    만약 초가 한 자리 수이면, 앞에 "0"을 붙여줌
    "분:초" 형식의 문자열을 만들어 반환

4. 결과 화면





LOAD를 통해서 기존 텍스트 파일에서 노래들을 불러왔다.

```
LOAD
ADD KDA|pop stars|3:23
ADD N.Flying|songbird|3:46
QPOP
SEARCH ARTIST N.Flying
SEARCH TITLE blue moon
SEARCH TITLE my love
SEARCH SONG N.Flying|blue moon
MAKEPL ARTIST N.Flying
MAKEPL TITLE blue moon
MAKEPL TITLE my love
MAKEPL SONG Maroon 5|sugar
MAKEPL ARTIST PSY
PRINT ARTIST
PRINT TITLE
PRINT LIST
DELETE ARTIST PSY
DELETE ARTIST PSY
DELETE TITLE blue moon
DELETE TITLE blue moon
DELETE LIST N.Flying|songbird
DELETE LIST N.Flying|songbird
DELETE SONG N.Flying|rooftop
DELETE SONG N.Flying|rooftop
PRINT ARTIST
DELETE SONG N.Flying|songbird
PRINT ARTIST
PRINT TITLE
PRINT LIST
EXIT
```

```
========LOAD========
N.Flying / blue moon / 3:36
N.Flying / moonshot / 3:26
N.Flying / songbird / 3:46
N.Flying / rooftop / 3:19
N.Flying / oh really. / 3:10
PSY / gangnam style / 3:25
PSY / gentleman / 3:14
PSY / daddy / 3:32
PSY / new face / 3:16
PSY / that that / 3:05
BTS / dynamite / 3:19
IU / blueming / 3:37
Michael Jackson / beat it / 4:18
Maroon 5 / sugar / 3:55
Westlife / my love / 3:51
Buzz / my love / 4:05
Hyorin / blue moon / 3:24

====================

========ADD========
KDA / pop stars / 3:23
====================

========ADD========
Error: 200
====================

========QPOP========
SUCCESS
====================

```

ADD기반으로 KDA|pop stars|3:23 , N.Flying|songbird|3:46 를 시도했다. KDA경우 성공했지만 NFlying 의 경우 이미 있던 노래라서 에러를 반환한다.
이후 QPOP을 통해 두개의 트리에 데이터를 제공한다.

```
LOAD
ADD KDA|pop stars|3:23
ADD N.Flying|songbird|3:46
QPOP
SEARCH ARTIST N.Flying
SEARCH TITLE blue moon
SEARCH TITLE my love
SEARCH SONG N.Flying|blue moon
MAKEPL ARTIST N.Flying
MAKEPL TITLE blue moon
MAKEPL TITLE my love
MAKEPL SONG Maroon 5|sugar
MAKEPL ARTIST PSY
PRINT ARTIST
PRINT TITLE
PRINT LIST
DELETE ARTIST PSY
DELETE ARTIST PSY
DELETE TITLE blue moon
DELETE TITLE blue moon
DELETE LIST N.Flying|songbird
DELETE LIST N.Flying|songbird
DELETE SONG N.Flying|rooftop
```

```

========SEARCH========
N.Flying/blue moon/3:36
N.Flying/moonshot/3:26
N.Flying/songbird/3:46
N.Flying/rooftop/3:19
N.Flying/oh really./3:10
====================

========SEARCH========
N.Flying/blue moon/3:36
Hyorin/blue moon/3:24
====================

========SEARCH========
Westlife/my love/3:51
Buzz/my love/4:05
====================

========SEARCH========
N.Flying/blue moon/3:36
====================

```

NFlying가수로 찾은목록과
blue moon,my love노래로 출력한 결과
노래 이름과 가수를 각각 제공하여 찾은결과

```
LOAD
ADD KDA|pop stars|3:23
ADD N.Flying|songbird|3:46
QPOP
SEARCH ARTIST N.Flying
SEARCH TITLE blue moon
SEARCH TITLE my love
SEARCH SONG N.Flying|blue moon
MAKEPL ARTIST N.Flying
MAKEPL TITLE blue moon
MAKEPL TITLE my love
MAKEPL SONG Maroon 5|sugar
MAKEPL ARTIST PSY
PRINT ARTIST
PRINT TITLE
PRINT LIST
DELETE ARTIST PSY
DELETE ARTIST PSY
DELETE TITLE blue moon
DELETE TITLE blue moon
DELETE LIST N.Flying|songbird
DELETE LIST N.Flying|songbird
DELETE SONG N.Flying|rooftop
DELETE SONG N.Flying|rooftop
PRINT ARTIST
DELETE SONG N.Flying|songbird
PRINT ARTIST
PRINT TITLE
```

```
54  ====================
55
56  ========MAKEPL========
57  N.Flying/blue moon/3:36
58  N.Flying/moonshot/3:26
59  N.Flying/songbird/3:46
60  N.Flying/rooftop/3:19
61  N.Flying/oh really./3:10
62  Count: 5/10
63  Time: 17min 17sec
64  ====================
65
66  ========MAKEPL========
67  ERROR: 500
68  ====================
69
70  ========MAKEPL========
71  N.Flying/blue moon/3:36
72  N.Flying/moonshot/3:26
73  N.Flying/songbird/3:46
74  N.Flying/rooftop/3:19
75  N.Flying/oh really./3:10
76  Westlife/my love/3:51
77  Buzz/my love/4:05
78  Count: 7/10
79  Time: 25min 13sec
80  ====================
```

```
========MAKEPL========
N.Flying/blue moon/3:36
N.Flying/moonshot/3:26
N.Flying/songbird/3:46
N.Flying/rooftop/3:19
N.Flying/oh really./3:10
Westlife/my love/3:51
Buzz/my love/4:05
Maroon 5/sugar/3:55
Count: 8/10
Time: 29min 8sec
====================

========MAKEPL========
ERROR: 500
====================
```

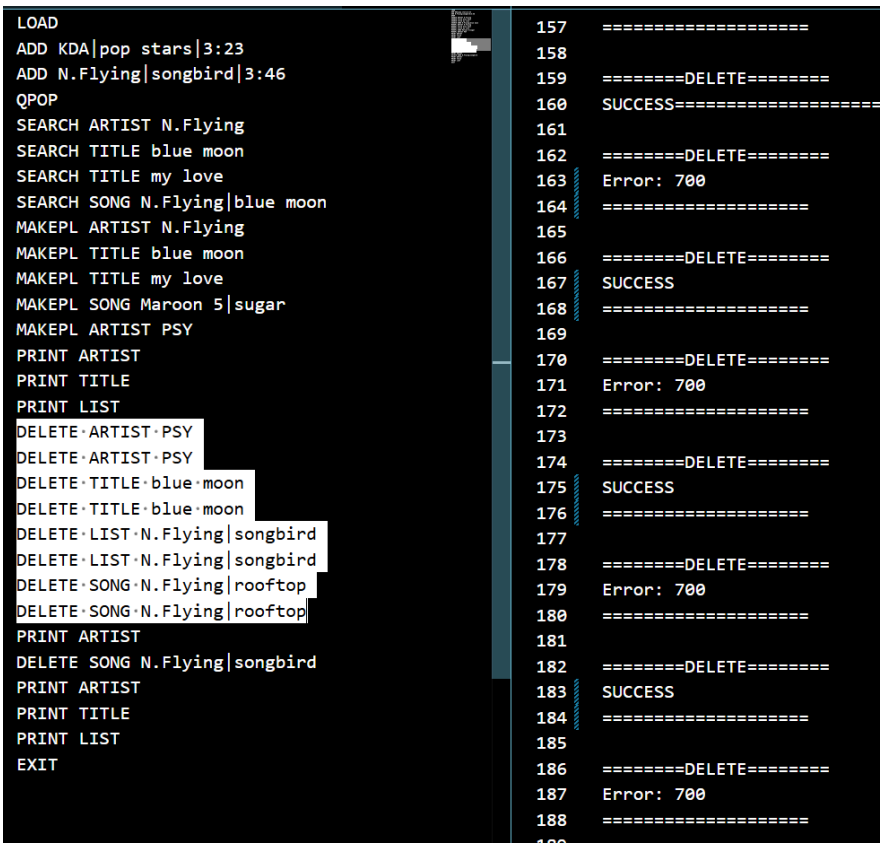
blue moon 가수로 먼저 플레이 리스트를 만들고
다음 명령어에서 blue moon 제목으로 플리를 만들지만 기존의 blue moon이라는 노래가 있어서 에러를 반환
이후 my love 노래들은 겹치지 않아서 추가가 올바르게 된 모습이다.
이후 노래 marron 5 - suger는 개별로 추가 하였고
이미 플리에 노래가 8개인데 psy의 노래는 3개 이상이어서 10개 이상을 담으려고 함으로 에러를 반환한다.

```
ADD KDA|pop stars|3:23
ADD N.Flying|songbird|3:46
QPOP
SEARCH ARTIST N.Flying
SEARCH TITLE blue moon
SEARCH TITLE my love
SEARCH SONG N.Flying|blue moon
MAKEPL ARTIST N.Flying
MAKEPL TITLE blue moon
MAKEPL TITLE my love
MAKEPL SONG Maroon 5|sugar
MAKEPL ARTIST PSY
PRINT ARTIST
PRINT TITLE
PRINT LIST
DELETE ARTIST PSY
DELETE ARTIST PSY
DELETE TITLE blue moon
DELETE TITLE blue moon
DELETE LIST N.Flying|songbird
DELETE LIST N.Flying|songbird
DELETE SONG N.Flying|rooftop
DELETE SONG N.Flying|rooftop
PRINT ARTIST
DELETE SONG N.Flying|songbird
PRINT ARTIST
PRINT TITLE
PRINT LIST
EXIT
```

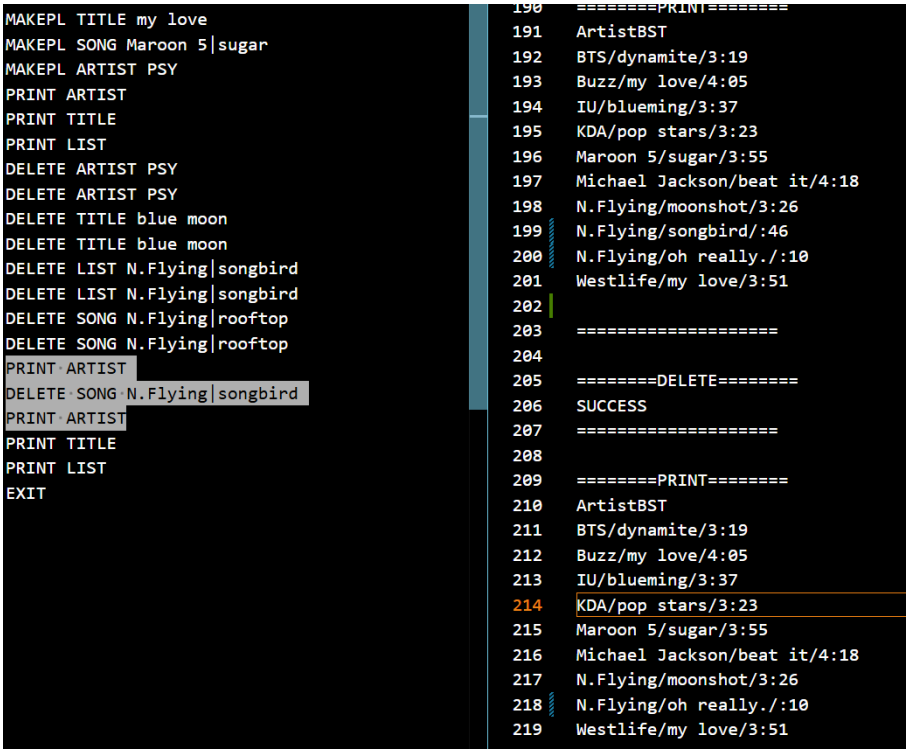
```
100   ArtistBST
101   BTS/dynamite/3:19
102   Buzz/my love/4:05
103   Hyorin/blue moon/3:24
104   IU/blueming/3:37
105   KDA/pop stars/3:23
106   Maroon 5/sugar/3:55
107   Michael Jackson/beat it/4:18
108   N.Flying/blue moon/3:36
109   N.Flying/moonshot/:26
110   N.Flying/songbird/:46
111   N.Flying/rooftop/:19
112   N.Flying/oh really./:10
113   PSY/gangnam style/3:25
114   PSY/gentleman/:14
115   PSY/daddy/:32
116   PSY/new face/:16
117   PSY/that that/:05
118   Westlife/my love/3:51
119
120   ====================
121
122   ========PRINT========
123   TitleBST
124   Michael Jackson/beat it/4:18
125   N.Flying/blue moon/3:36
126   Hyorin/blue moon/3:24
127   IU/blueming/3:37
128   PSY/daddy/3:32
129   BTS/dynamite/3:19
130   PSY/gangnam style/3:25
131   PSY/gentleman/3:14
132   N.Flying/moonshot/3:26
133   Westlife/my love/3:51
134   Buzz/my love/4:05
135   PSY/new face/3:16
136   N.Flying/oh really./3:10
137   KDA/pop stars/3:23
138   N.Flying/rooftop/3:19
139   N.Flying/songbird/3:46
140   Maroon 5/sugar/3:55
141   PSY/that that/3:05
```

```
========PRINT========
N.Flying/blue moon/3:36
N.Flying/moonshot/3:26
N.Flying/songbird/3:46
N.Flying/rooftop/3:19
N.Flying/oh really./3:10
Westlife/my love/3:51
Buzz/my love/4:05
Maroon 5/sugar/3:55
Count: 8/10
Time: 29min 8sec
```

각각의 ARTList와 TitleList PlayList대로 각각 출력하는 모습이다. 특히 플레이리스트에서는 카운트와 총 시간이 출력되는 모습또한 확인 가능하다.



아티스트 기반 삭제, 이름기반 삭제, 제목과 가수 지목 삭제를 각각 연속으로 수행한 모습이다.
처음에는 삭제를 하지만 두번째에는 삭제할 대상이 없어 에러를 반환하는 모습이다.



위에 결과에서 4번의 삭제가 성공하여 노래들이 삭제된 것을 확인 가능하다.
또한 이후 노래를 지목하여 삭제하였고 이후 없어진 모습 또한 확인 가능하다.

```
SEARCH ARTIST N.Flying
SEARCH TITLE blue moon
SEARCH TITLE my love
SEARCH SONG N.Flying|blue moon
MAKEPL ARTIST N.Flying
MAKEPL TITLE blue moon
MAKEPL TITLE my love
MAKEPL SONG Maroon 5|sugar
MAKEPL ARTIST PSY
PRINT ARTIST
PRINT TITLE
PRINT LIST
DELETE ARTIST PSY
DELETE ARTIST PSY
DELETE TITLE blue moon
DELETE TITLE blue moon
DELETE LIST N.Flying|songbird
DELETE LIST N.Flying|songbird
DELETE SONG N.Flying|rooftop
DELETE SONG N.Flying|rooftop
PRINT ARTIST
DELETE SONG N.Flying|songbird
PRINT ARTIST
PRINT TITLE
PRINT LIST
EXIT
```

```
209   ========PRINT========
210   ArtistBST
211   BTS/dynamite/3:19
212   Buzz/my love/4:05
213   IU/blueming/3:37
214   KDA/pop stars/3:23
215   Maroon 5/sugar/3:55
216   Michael Jackson/beat it/4:18
217   N.Flying/moonshot/3:26
218   N.Flying/oh really./:10
219   Westlife/my love/3:51
220
221   ====================
222
223   ========PRINT========
224   TitleBST
225   Michael Jackson/beat it/4:18
226   IU/blueming/3:37
227   BTS/dynamite/3:19
228   N.Flying/moonshot/3:26
229   Westlife/my love/3:51
230   Buzz/my love/4:05
231   N.Flying/oh really./3:10
232   KDA/pop stars/3:23
233   Maroon 5/sugar/3:55
234
235   ====================
236
237   ========PRINT========
238   N.Flying/moonshot/3:26
239   N.Flying/oh really./3:10
240   Westlife/my love/3:51
241   Buzz/my love/4:05
242   Maroon 5/sugar/3:55
243   Count: 5/10
244   Time: 18min 27sec
245
246   ====================
247
```

삭제는 두가지 BST와 플레이 리스트 링크드 리스트에서 모두 사라져 줄어든 모습을 확인 가능하다.

```
PRINT ARTIST
PRINT TITLE
PRINT LIST
EXIT
```

```
246   ====================
247
248   ========EXIT========
249   Success
250   ====================
251
252
```

이후 엑싯을 통해서 프로그램이 종료되었다.

```
==5607== Memcheck, a memory error detector
==5607== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5607== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==5607== Command: ./run
==5607==
==5607==
==5607== HEAP SUMMARY:
==5607==     in use at exit: 0 bytes in 0 blocks
==5607==   total heap usage: 201 allocs, 201 frees, 112,680 bytes allocated
==5607==
==5607== All heap blocks were freed -- no leaks are possible
==5607==
==5607== For lists of detected and suppressed errors, rerun with: -s
==5607== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

명령어로 run할때 누수를 점검하였고 no leaks are possible이라는 대답을 받았다.

5. 고찰

프로잭트를 구현하는 과정에서 플레이 리스트를 만들 때 제목이 겹쳐도 다른 노래가 있다면 넣어야 하는것이 아닐까?라는 생각이 들기도 했습니다. 이번 과제에서는 명세서 및 묻고 답하기 과젱에서 에러 반환으로 수정하게 되었습니다.
내 생각관으로 코드 작성하다가 명세서를 잘 못봐서 돌아가는 경험을 하기도 했습니다.
