

DATA STRUCTURE PROJECT #1

학번: 2021202087 / 이름: 장현웅 / 과제 기간: 20240916~20241013

INTRODUCTION

프로젝트 목표

프로젝트의 목표는 Queue, Binary Search Tree, Linked list 자료구조를 사용해 자막 관리 프로그램을 구현하는 것이다. 각각의 자료구조를 SubtitleQueue, SubtitleBST, SectionList Class 로 구현하고, Manager Class 와의 상호작용을 통해 자막 프로그램이 구동되게 하도록 한다. 다양한 자료구조의 사용 방법을 익히고, 각 자료구조의 동작을 처리하는데 필요한 알고리즘을 이해하도록 한다.

입출력

command.txt 파일에는 구동해야 하는 명령어가 줄 단위로 들어가 있다. subtitle.txt 파일에는 그림과 같은 형식으로 자막 시간, 자막 내용이 한 줄에 하나씩, 자막 시간이 정렬되지 않은 순서대로 들어가 있다. 자막 시간 형식은 XX:XX:XX 이며, 기본적으로는 중복되지 않는다고 가정한다. (프로그램은 중복된 노드에 대해서도 정상 동작한다.) 자막 내용은 200 자(공백 포함)를 넘기지 않는다. 프로그램의 command.txt 에 따라 subtitle.txt 의 자막정보를 자료구조에 저장해 조작하고, 각 command 실행에 따른 로그를 log.txt 에 출력한다. 아래에 차례로 command.txt, subtitle.txt, log.txt 의 예시이다.

```
LOAD
QPOP
PRINT
SECTION 3 00:52:10 00:52:30
SECTION 2 00:53:50 00:54:23
PRINT 3
PRINT 2
DELETE EQUAL 00:52:36
PRINT
DELETE UNDER 00:52:23
PRINT
PRINT 3
EXIT
```

```
01:03:45 You're not listening to me!
00:52:36 You're welcome.
00:54:16 Hey, you! Where's the other mother?
00:54:18 I wanna go home.
00:53:50 Mom! Dad!
00:53:56 Oh, God. I'm still here?
00:54:19 All will be swell, soon as Mother's refre
00:54:23 Her strength is our strength.
00:52:21 Bed? Before dinner?
00:52:20 Right now!
00:52:19 I'm going to bed.
00:21:37 See you soon.
00:52:23 I'm really, really tired. Yeah.
01:09:34 Ok.
```

```
===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:19 - All will be swell, soon as Mother's ref
00:54:23 - Her strength is our strength.
00:52:21 - Bed? Before dinner?
00:52:20 - Right now!
00:52:19 - I'm going to bed.
00:21:37 - See you soon.
00:52:23 - I'm really, really tired. Yeah.
01:09:34 - Ok.
01:09:24 - Challenge her, then.
01:09:29 - She's got a thing for games.
01:09:26 - She may not play fair, but she won't re
01:09:17 - I have to go back. They are my parents.
01:11:36 - A finding things game.
```

미리 알아둘 것

class Time

프로젝트 전체에서 XX:XX:XX 형식의 시간을 수치로 지속적으로 비교하고 사용한다. 따라서 시, 분, 초의 시간을 각각 short 로 저장하는 class 를 만들었으며, <, >, ≥, ≤, ==, !=, >, < 연산자를 오버로딩해 프로그램 구현 전체에서 유용하게 사용하고자 하였다. Time.h 에서 정의를 찾아볼 수 있다.

Time >> operation 오버로딩은 문자를 읽어 들여올 때 시간이 전부 두 자리 정수이며, 시간은 00~99, 분, 초는 00~59 의 정수가 되는 것을 보장한다. 구현에 대한 소개는 생략한다.

Datapair

`std::pair<Time, std::string>`의 형식으로 typedef 된 자료형이다. `baseheader.h` 에서 typedef 를 확인할 수 있다.

search key, node key

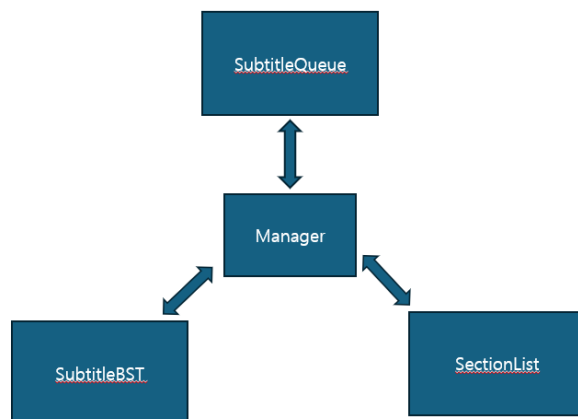
Binary Search Tree 에서 Tree 에 매번 비교해야하는, 자리를 찾아야 하는 동일한 key 를 search key, 그리고 매번 달라지는 node 가 가진 key 를 코드와 문서에서 종종 node key 로 서술한다.

section header

SectionList 에서 subtitleListNode(내용 노드)를 삽입하게 될 SectionListNode(헤더 노드)를 지칭한다.

FLOWCHART

객체 간 관계



각 자료구조의 헤더 파일을 보면, 서로 다른 자료구조에 대한 헤더 파일 include 가 SubtitleBST 에서 SubtitleQueue.h 밖에 없다. SubtitleBST 에서 SubtitleQueue.h 를 참조하는 이유는 프로그램 전체에서 SubtitleQueue 객체를 참조하는 것이 아니라, 잠깐 SubtitleBST 의 데이터를 저장하기 위한 buffer 로, STL queue 대신 사용한 것이다.

프로그램 전체에서는 Manager 객체에서 각 명령어를 실행하면서 heap 에 세 가지의 자료구조 객체를 각각 SQptr, SBSTptr, SLptr 에 동적 할당해서 사용하며, 이 세 가지 객체가 Manager 를 거치지 않고 내부 함수적으로 직접 데이터를 주고받는 경우가 하나도 존재하지 않는다. 따라서 실제로 모든 command 는 Manager 와 여러 차례 자료 구조가 데이터를 주고받으며 진행된다.

대표 CLASS 가 지원하는 OPERATION 소개

Manager

프로그램 전반에서 데이터의 흐름을 제어한다.

- 주로 관련하는 command: 전부

- property

SQptr, SBSTptr, SLptr 3 개의 포인터를 가지고 있으며 Manager::Run() 동안에 자료구조 객체를 동적으로 할당해 관리한다.

- Operation

Run(): 프로그램을 동작시킨다.

Load(): LOAD 명령어를 실행한다.

Qpop(): QPOP 명령어를 실행한다.

Section(): SECTION section_num start_time end_time 명령어를 실행한다.

Print(): PRINT 명령어를 실행한다.

Print(const int&): PRINT section_num 명령어를 실행한다.

DeleteEqual(): DELETE EQUAL delete_time 명령어를 실행한다.

DeleteUnder(): DELETE UNDER delte_time 명령어를 실행한다.

EXIT 명령어의 경우 Run() 내부에서 명령어를 읽었을 때 자체적으로 각 자료구조 pointer 를 할당해제한다.

Subtitle Queue

subtitle.txt 에서 LOAD 해온 데이터로 구성된 Queue 이다

- 주로 관련하는 command

LOAD / QPOP / EXIT

- property

■ SubtitleQueue:

front, rear 노드 포인터 - queue 의 front node, rear node 를 관리한다.

capacity, nodecnt - queue 의 최대 용량을 관리한다.

■ SubtitleQueueNode:

Datapair 와 nextnode 로 구성된 linkedlist node 이다.

- 지원하는 operation

Push(): SubtitleQueue rear 에 새로운 노드를 추가한다.

Pop(): SubtitleQueue front 를 삭제하고 Datapair 를 반환한다.

Front(): SubtitleQueue front 의 Datapair 를 반환한다.

PrintQueue(): front 에서 rear 까지 순회하면서 모든 노드를 출력한다.

~SubtitleQueue(): 직접적으로 호출되지는 않지만, delete 키워드로 호출된다. front 부터 시작해서 모든 노드를 삭제한다.

print operation 의 경우 queue 에서는 모든 원소를 pop 하면서 확인하지 않고, front 에서

rear 까지 순회하면서 출력한다. (push, pop 기본 연산이 지원되며 중간에 원소의 수정이 없기 때문에 Queue 로 인정한다는 답변을 받았다.)

SubtitleBST

SubtitleQueue 에서 QPOP 한 데이터를 자막 시간을 key 로 하여 구성한 이진 트리이다.

- 주로 관련하는 command

QPOP / PRINT / SECTION section_num start_time end_time / DELETE EQUAL delete_time / DELETE UNDER delete_time / EXIT

- property

- SubtitleBST

root 포인터 - 트리 전체의 root 를 담고 있다.

- SubtitleBSTNode

Datapair 와 left, right ptr 로 구성된 Binary Search Tree node 이다.

- 지원하는 operation

Insert(): BST 에 새로운 node 를 삽입한다.

PrintBST(): BST 전체를 중위 순회하며 출력한다.

SearchRange() : BST 에서 [start_time, end_time] 범위에 있는 모든 데이터를 임시 SubtitleQueue 버퍼에 담는다.

DeleteEveryEqual(): BST 에 존재하는, 모든 delete_time 과 같은 subtitle Time 값을 가진 노드를 삭제한다.

DeleteUnder(): BST 에 존재하는, delete_time 보다 아래의 subtitle Time 값을 가진 노드를 삭제한다.

DeleteBST(): BST 에 존재하는 모든 노드를 삭제한다.

그 외 Debug 용으로 PrintStructure() 함수도 존재한다. 이 함수는 현재 BST 의 전체 시간 키를 BST 전체가 좌로 90 도 돌아간 모습으로 출력하도록 설계했다.

SectionList

SECTION (...) command 를 통해 section_num 인수 값의 section number 를 가지는 섹션 헤더를 만들고, SubtitleBST 에서 일정 범위에 속하는 노드들을 중위 탐색하며 그 Datapair 들을 section header 부터 오름차순으로 이은 이차원 리스트이다. (중간 삽입이 없는)

- 주로 관련하는 command

SECTION section_num start_time end_time / PRINT section_num / EXIT

- property

■ SectionList

head 와 tail 포인터가 존재한다. 새로운 section header 삽입이 오직 list 끝에서만 이뤄진다.
Search 는 head 부터 시작하여 임의의 SectionListNode 을 탐색 가능하다.

■ SectionListNode

section_num: section number 를 저장한다.

subListHead: 이 섹션 헤더에 연결된 자막 리스트의 head 이다. 여기에서부터 출력이 시작된다.

subListTail: 이 섹션 헤더에 연결된 자막 리스트의 tail 이다. 항상 여기에 새 자막노드가 삽입된다.

■ SubtitleListNode

Datapair data 와 next pointer 가 있는, linkedlist 의 노드이다.

● 지원하는 operation

GenerateNewSection(): 헤더노드 리스트의 tail 에 새로운 헤더노드를 추가한다.

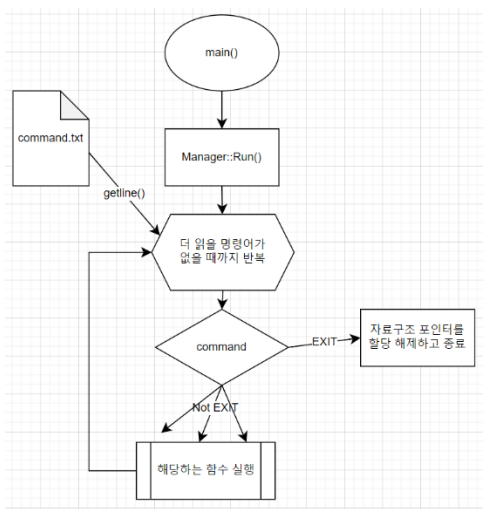
InsertSubtitle(): 새롭게 생성한 섹션 헤더의 tail 에 내용노드를 추가한다.

PrintSection(): 헤더노드 리스트를 head 부터 탐색 시작해서 section_num 과 동일한 섹션
번호를 가진 section 헤더를 찾고, 이후 해당 자막노드 리스트를 출력한다.

프로그램은 실제로 각각의 명령어 수행 과정이 하나의 함수가 아닌 여러 개의 함수와 여러 깊이의
호출로 구성 되어 있고, 여러 클래스 사이의 데이터 흐름으로 구성 되어 있다. 따라서
Manager()에서 불러내는 Operation 까지만 설명하도록 한다.

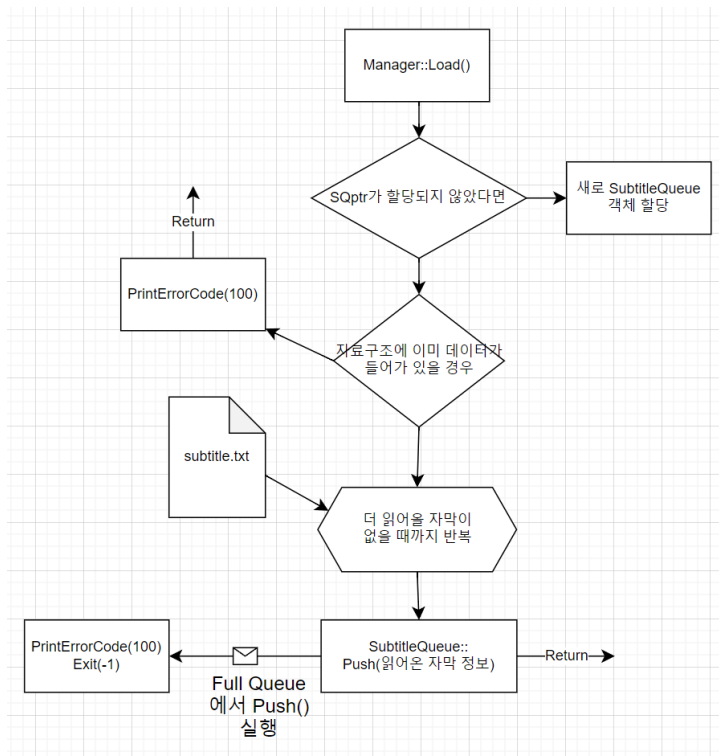
MANAGER::RUN()

다음과 같은 구조도로 실행된다.



LOAD 명령어

다음과 같은 구조도로 실행된다.



QPOP 명령어

Manager::Qpop()이 실행된 후, SubtiteBST 객체가 없으면 SBSTptr 에 동적 할당하도록 한다. Queue 가 비어있을 때 Pop 되는 예외상황을 subtitleQueue 가 존재하지 않거나 불러온 데이터가 없는 경우로 구현하였다. 따라서 실제로 SubtitleQueue 는 비어있을 때 Pop() 되는 상황은 발생하지 않는다.

이후, SubtitleQueue 객체에서 지속적으로 Pop() 해서 얻어온 Datapair 를 SBSTptr 에 Insert() 한다. 내부적으로 Datapair 를 가지고 새로운 노드를 생성하여 삽입하게 된다.

PRINT 명령어

Manager::Print()가 실행된 후 SubtitleBST 객체가 없으면 에러 코드를 로그에 출력하고 리턴한다. SBSTptr 의 PrintBST() 명령어를 통해 트리 전체를 in-order traversal 하면서 전체 노드를 출력하게 된다.

BST 에 노드가 존재하지 않는 경우 에러 로그를 출력한다.

PRINT SECTION_NUM 명령어

Manager::Print(const int&) 가 실행된 후 SectionList 객체가 없으면 에러 코드를 로그에 출력하고 리턴한다.

SLptr 에 PrintSection() 명령어를 통해 SectionList 객체에 앞서 SubtitleBST 에서 생성한 section_num 의 sectionheader 가 존재하는지 확인하고, 해당 섹션 헤더의 노드들을 출력할 수 있다.

내부적으로, 해당 섹션 넘버가 검색되지 않았을 경우 에러를 출력한다

SECTION SECTION_NUM START_TIME END_TIME 명령어

SLptr 가 할당된 적이 없을 경우 SectionList 객체를 동적 할당하게 된다.

SECTION 명령어가 수행되면 GenerateNewSection()으로 section_num 에 해당하는 섹션 헤더를 먼저 생성하게 된다.

만약 SBSTptr 내부에 데이터가 없거나 SBSTptr 가 할당된 적이 없다면 에러 코드를 출력하고 리턴하게 된다.

SubtitleQueue* 형식의, SearchRange()의 결과를 저장할 임시 버퍼를 생성한다. SubtitleBST 객체에 SearchRange() 함수를 호출하여 start_time~ end_time 에 포함되는 모든 노드를 찾아서, SearchRange 의 결과를 저장한다.

만약 아무 노드도 발견되지 않았다면 에러를 출력하고 리턴한다.

이후 bufferSQ 가 빌 때까지 pop()을 하면서 SLptr()의 섹션 헤더 뒤의 내용 노드 리스트의 Tail 에 지속적으로 노드를 삽입하게 된다. (이차원 리스트의 Tail)

ALGORITHM

각 명령어의 동작에서 주축이 되는 알고리즘을 설명하도록 한다.

RESULT SCREEN

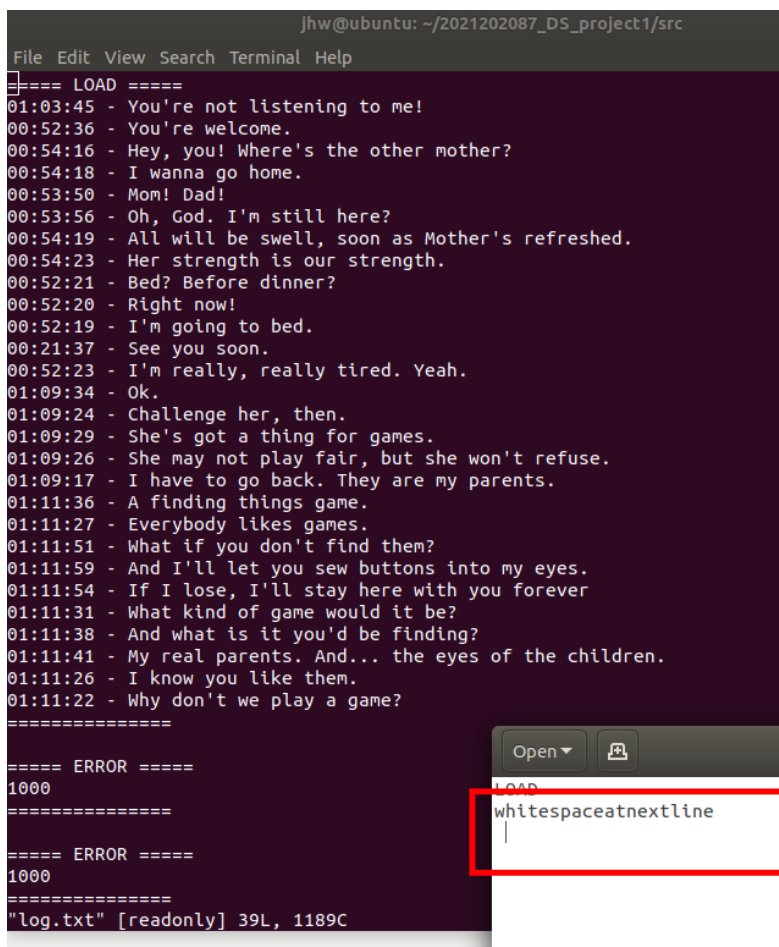
command.txt 사진과 함께 설명한다.

PRINTERRORCODE(1000)

모든 명령어는 줄 단위로 읽기 때문에, \n 으로 다음 줄이 읽을 수 있다면, 명령어가 읽히지 않을 때 PrintErrorCode(1000)을 출력한다.

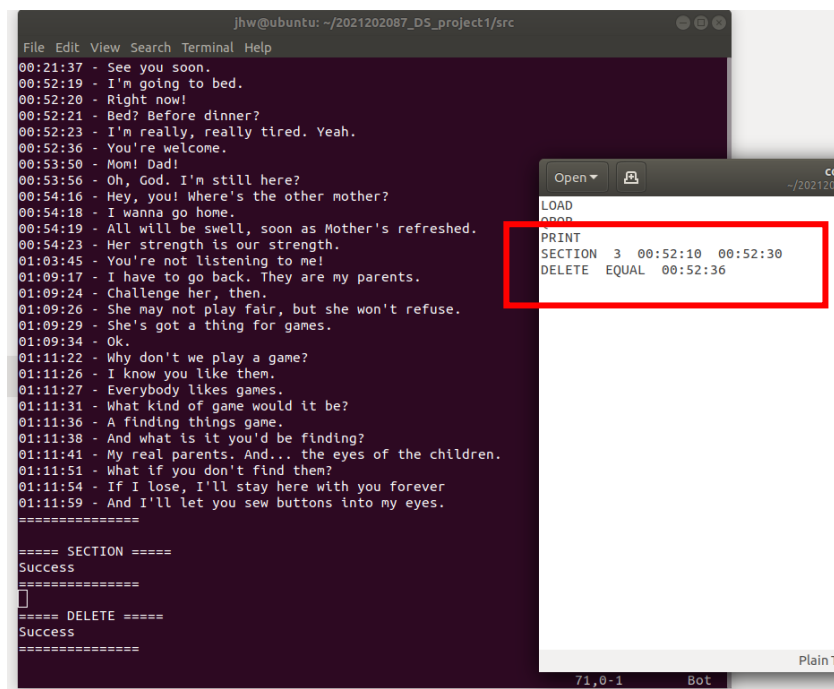
```
jhw@ubuntu: ~/2021202087_DS_project1/src
File Edit View Search Terminal Help
===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:19 - All will be swell, soon as Mother's refreshed.
00:54:23 - Her strength is our strength.
00:52:21 - Bed? Before dinner?
00:52:20 - Right now!
00:52:19 - I'm going to bed.
00:21:37 - See you soon.
00:52:23 - I'm really, really tired. Yeah.
01:09:34 - Ok.
01:09:24 - Challenge her, then.
01:09:29 - She's got a thing for games.
01:09:26 - She may not play fair, but she won't refuse.
01:09:17 - I have to go back. They are my parents.
01:11:36 - A finding things game.
01:11:27 - Everybody likes games.
01:11:51 - What if you don't find them?
01:11:59 - And I'll let you sew buttons into my eyes.
01:11:54 - If I lose, I'll stay here with you forever
01:11:31 - What kind of game would it be?
01:11:38 - And what is it you'd be finding?
01:11:41 - My real parents. And... the eyes of the children.
01:11:26 - I know you like them.
01:11:22 - Why don't we play a game?
=====
===== ERROR =====
1000
=====
'log.txt' [readonly] 35L, 1149C
```


또한, 다음 줄에 공백을 포함한 읽을 수 없는 문자가 존재할 때도 1000 을 출력한다.



```
jhw@ubuntu: ~/2021202087_DS_project1/src
File Edit View Search Terminal Help
===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:19 - All will be swell, soon as Mother's refreshed.
00:54:23 - Her strength is our strength.
00:52:21 - Bed? Before dinner?
00:52:20 - Right now!
00:52:19 - I'm going to bed.
00:21:37 - See you soon.
00:52:23 - I'm really, really tired. Yeah.
01:09:34 - Ok.
01:09:24 - Challenge her, then.
01:09:29 - She's got a thing for games.
01:09:26 - She may not play fair, but she won't refuse.
01:09:17 - I have to go back. They are my parents.
01:11:36 - A finding things game.
01:11:27 - Everybody likes games.
01:11:51 - What if you don't find them?
01:11:59 - And I'll let you sew buttons into my eyes.
01:11:54 - If I lose, I'll stay here with you forever
01:11:31 - What kind of game would it be?
01:11:38 - And what is it you'd be finding?
01:11:41 - My real parents. And... the eyes of the children.
01:11:26 - I know you like them.
01:11:22 - Why don't we play a game?
=====
===== ERROR =====
1000
=====
===== ERROR =====
1000
=====
"log.txt" [readonly] 39L, 1189C
```

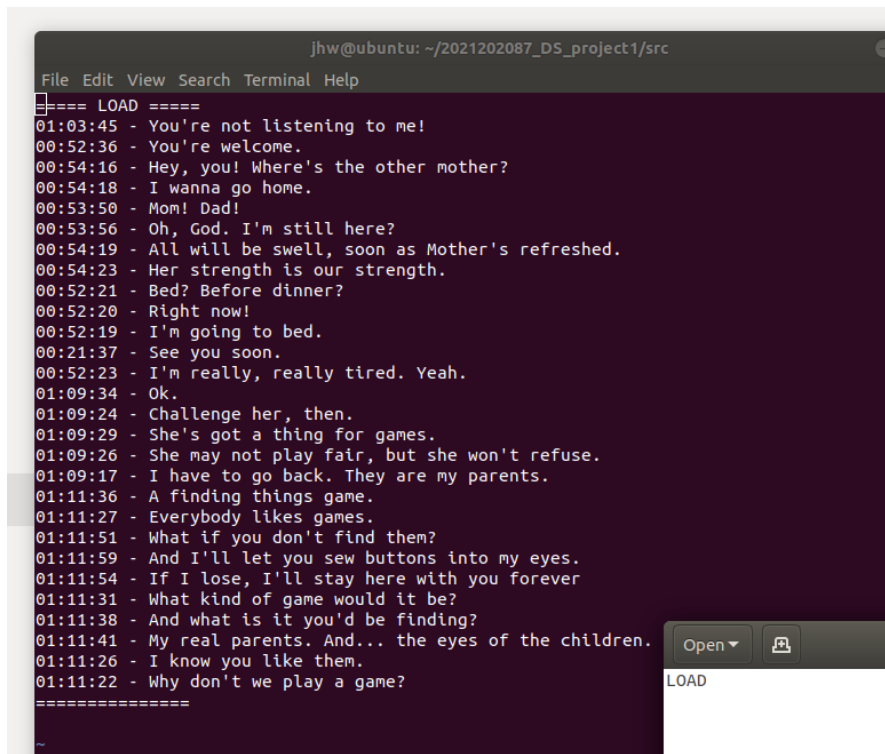
단 명령어와 명령어 사이 그리고 끝에 들어가는 추가 공백은 다음과 같이 정상 처리되도록 제작하였다.



```
jhw@ubuntu: ~/2021202087_DS_project1/src
File Edit View Search Terminal Help
00:21:37 - See you soon.
00:52:19 - I'm going to bed.
00:52:20 - Right now!
00:52:21 - Bed? Before dinner?
00:52:23 - I'm really, really tired. Yeah.
00:52:36 - You're welcome.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:54:19 - All will be swell, soon as Mother's refreshed.
00:54:23 - Her strength is our strength.
01:03:45 - You're not listening to me!
01:09:17 - I have to go back. They are my parents.
01:09:24 - Challenge her, then.
01:09:26 - She may not play fair, but she won't refuse.
01:09:29 - She's got a thing for games.
01:09:34 - Ok.
01:11:22 - Why don't we play a game?
01:11:26 - I know you like them.
01:11:27 - Everybody likes games.
01:11:31 - What kind of game would it be?
01:11:36 - A finding things game.
01:11:38 - And what is it you'd be finding?
01:11:41 - My real parents. And... the eyes of the children.
01:11:51 - What if you don't find them?
01:11:54 - If I lose, I'll stay here with you forever
01:11:59 - And I'll let you sew buttons into my eyes.
=====
===== SECTION =====
Success
=====
===== DELETE =====
Success
=====
```

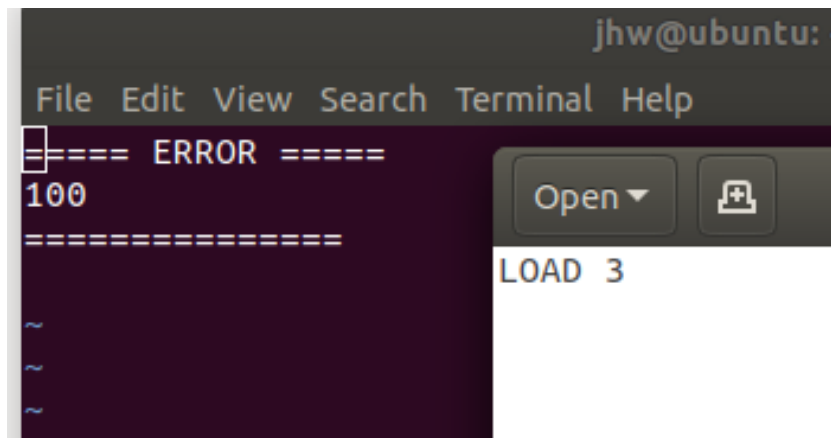
LOAD

LOAD 명령어를 실행할 때 결과는 다음과 같다.



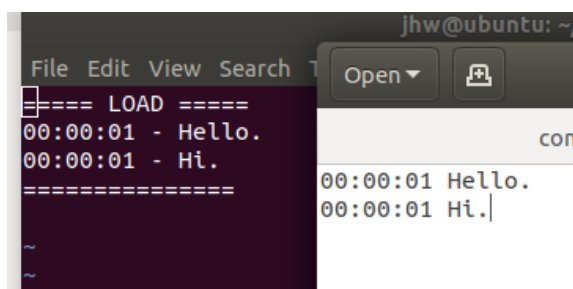
```
jhw@ubuntu: ~/2021202087_DS_project1/src
File Edit View Search Terminal Help
===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:19 - All will be swell, soon as Mother's refreshed.
00:54:23 - Her strength is our strength.
00:52:21 - Bed? Before dinner?
00:52:20 - Right now!
00:52:19 - I'm going to bed.
00:21:37 - See you soon.
00:52:23 - I'm really, really tired. Yeah.
01:09:34 - Ok.
01:09:24 - Challenge her, then.
01:09:29 - She's got a thing for games.
01:09:26 - She may not play fair, but she won't refuse.
01:09:17 - I have to go back. They are my parents.
01:11:36 - A finding things game.
01:11:27 - Everybody likes games.
01:11:51 - What if you don't find them?
01:11:59 - And I'll let you sew buttons into my eyes.
01:11:54 - If I lose, I'll stay here with you forever
01:11:31 - What kind of game would it be?
01:11:38 - And what is it you'd be finding?
01:11:41 - My real parents. And... the eyes of the children.
01:11:26 - I know you like them.
01:11:22 - Why don't we play a game?
=====
~
```

추가 인수가 들어가면 다음과 같이 오류가 발생한다.



```
jhw@ubuntu: ~
File Edit View Search Terminal Help
===== ERROR =====
100
=====
~
~
~
```

subtitle.txt 에 중복된 자막 시간이 있을 경우에도 작동 가능하다.



```
jhw@ubuntu: ~
File Edit View Search Terminal Help
===== LOAD =====
00:00:01 - Hello.
00:00:01 - Hi.
=====
~
~
```

다음과 같은 형식으로 subtitle.txt 에 200 자의 자막내용의 자막정보 10 개가 10 번 반복되는 subtitle.txt 의 입력을 처리하였다. (결과에 subtitleload.txt 로 첨부함) 결과는 다음과 같다.

문제 없이 출력되는 것을 확인할 수 있다.

101 번째 subtitle.txt 줄에 추가 정보를 삽입하는 경우 오류가 뜨고 프로그램이 종료된다.

이미 자료구조에 데이터가 적재되어 비워지지 않은 상태일 경우 다음과 같이 치명적이지 않은 에러가 발생한다.

특히 SectionList 에 들어간 노드를 삭제하는 기능은 EXIT 밖에 없기 때문에, SECTION 명령어를 수행하면 다시 LOAD 는 무조건 실패한다.

BST 에 있는 데이터를 비웠을 때는 정상적으로 LOAD 된다.

```
01:11:22 - Why don't we play a game?
=====

===== QPOP =====
Success
=====

===== DELETE =====
Success
=====

===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:19 - All will be sweet now!
00:54:23 - Her strength is in her love.
00:52:21 - Bed? Before dinner?
00:52:20 - Right now!
00:52:19 - I'm going to bed.
00:21:37 - See you later.
00:52:23 - I'm ready.
01:09:34 - Ok.
01:09:34 - Ok.
```

QPOP

실행한 결과는 다음과 같다.

```
jhw@ubuntu: ~/2021202087_DS_project1/src
File Edit View Search Terminal Help

===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?

(중략)

01:11:31 - What kind of game would it be?
01:11:38 - And what is it you'd be finding?
01:11:41 - My real parents. And... the eyes of the children.
01:11:26 - I know you like them.
01:11:22 - Why don't we play a game?
=====

===== QPOP =====
Success
=====
```

추가 인수가 전달 되었을 시 오류가 출력되지만, 종료되지는 않는다.

```
jhw@ubuntu: ~/2021202087_DS_project1/src
File Edit View Search Terminal Help

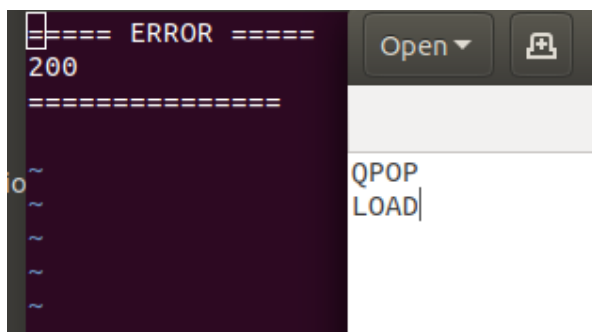
===== ERROR =====
200
=====

===== LOAD =====
01:03:45 - You're not listening to me!
00:52:36 - You're welcome.
00:54:16 - Hey, you! Where's the other mother?
00:54:18 - I wanna go home.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here?
00:54:19 - All will be sweet now!
00:54:23 - Her strength is in her love.
00:52:21 - Bed? Before dinner?
00:52:20 - Right now!

===== QPOP =====
Success
=====

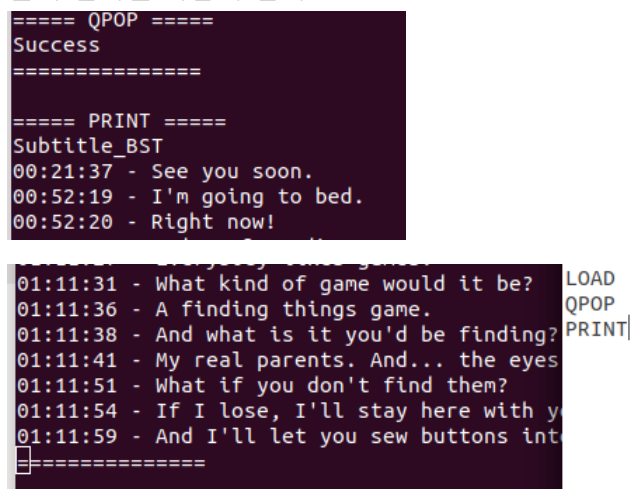
===== QPOP 3 =====
LOAD
QPOP
"log.txt" 39L, 1190C
```

load 하지 않고 QPOP 한 경우 치명적인 에러로 간주하여 프로그램이 강제 종료된다.

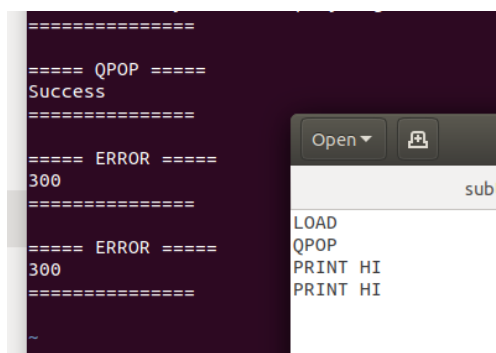


PRINT

출력 결과는 다음과 같다.

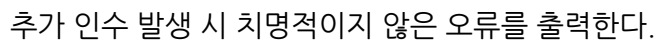


추가 인수 전달 시 치명적이지 않은 에러를 발생시킨다.

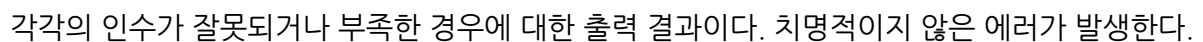


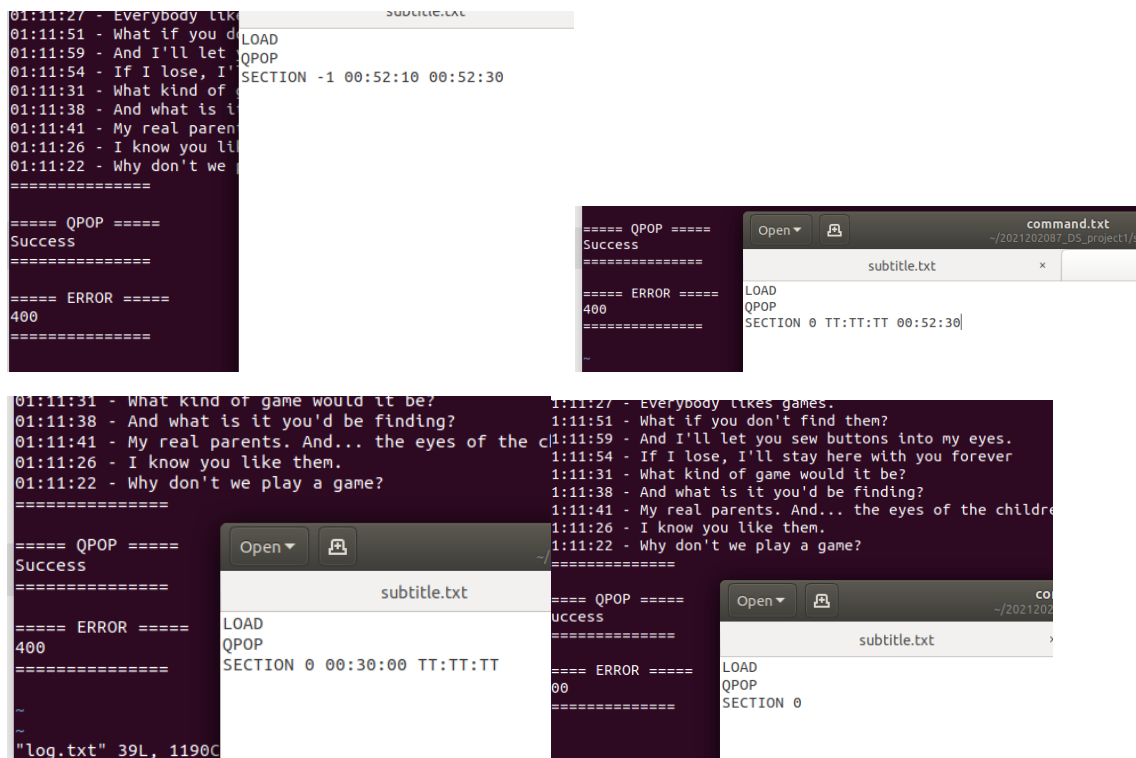
PRINT SECTION_NUM

출력 결과는 다음과 같다.

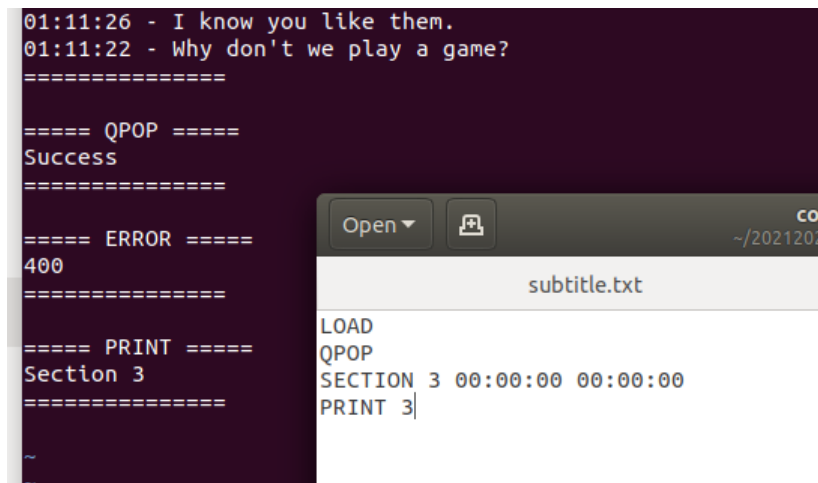


정상적으로 인수를 입력한 결과는 다음과 같다.



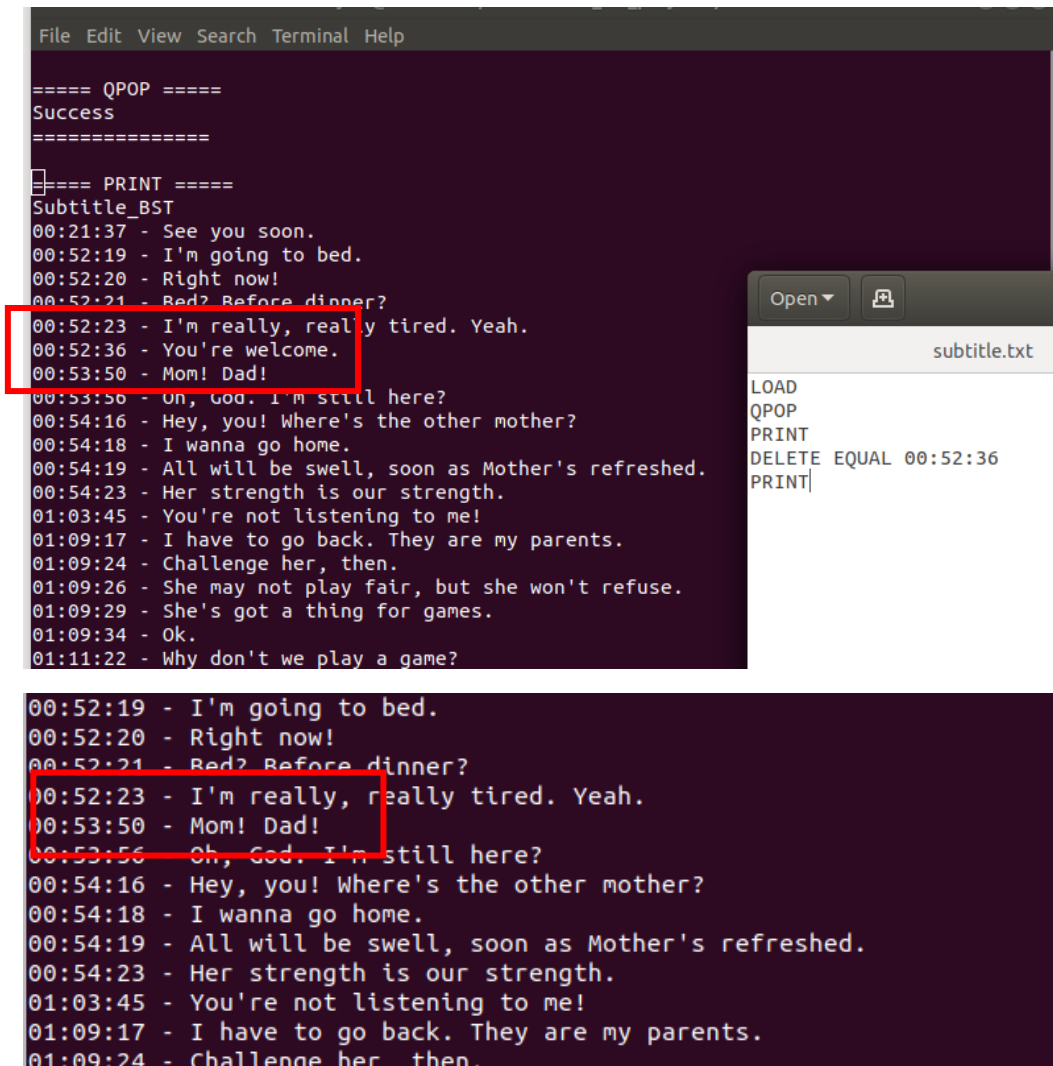


SECTION 명령어의 프로젝트 요구사항에서는 “SECTION 명령어 실행 시” 섹션 헤더를 먼저 생성한다고 서술하고 있다. 따라서 탐색된 노드가 없을 경우에도 SECITON 명령어로 다음과 같이 빈 로그가 print 결과로 출력되도록 하였다.

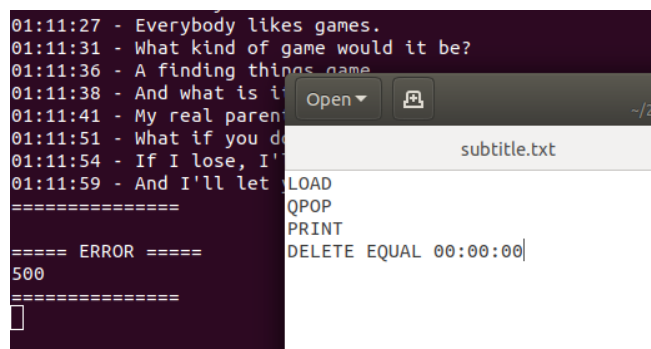


DELETE EQUAL DELETE_TIME

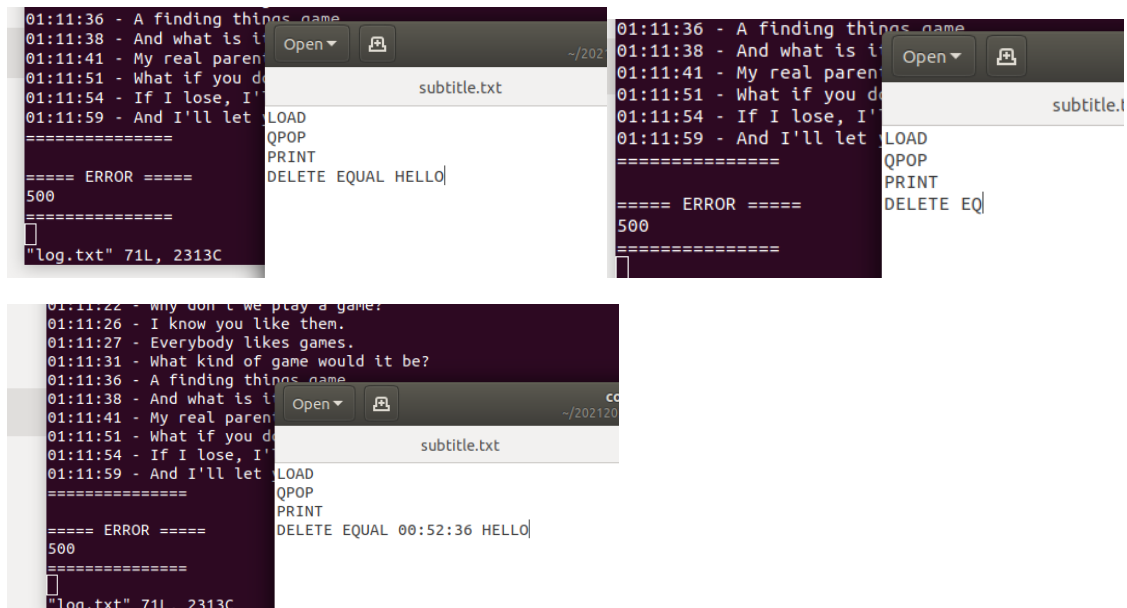
출력 결과는 다음과 같다. 이전 PRINT 출력에 대비해 검색된 노드가 사라진 출력이 되었음을 알 수 있다.



대상 노드가 검색되지 않았을 시 치명적이지 않은 에러를 출력한다.

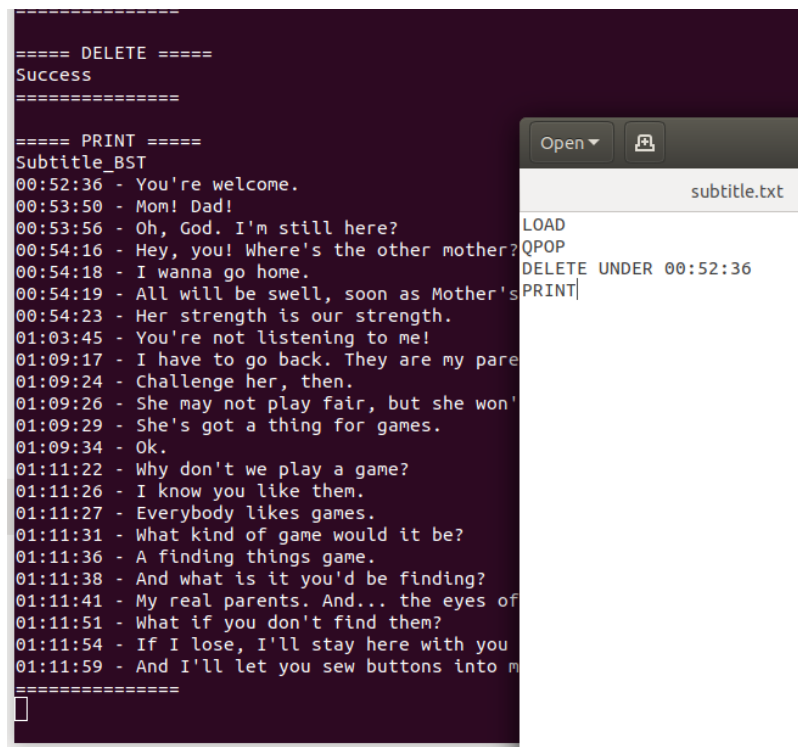


올바르지 않은 인수나 추가 인수를 전달 시 치명적이지 않은 에러를 출력한다.



DELETE UNDER DELETE_TIME

결과는 다음과 같다. 00:52:36 의 시간 미만의 BST node 삭제가 이뤄진 걸 확인할 수 있다.



삭제 대상 노드가 검색되지 않았을 시 치명적이지 않은 오류를 출력한다.

```

Success
=====
===== ERROR =====
500
=====
===== PRINT =====
Subtitle_BST
00:21:37 - See you soon.
00:52:19 - I'm going to bed.
00:52:20 - Right now!
00:52:21 - Bed? Before dinner?
00:52:23 - I'm really, really tired.
00:52:36 - You're welcome.
00:53:50 - Mom! Dad!
00:53:56 - Oh, God. I'm still here.
00:54:16 - Hey, you! Where's your room?
00:54:18 - I wanna go home.
00:54:19 - All will be sweet.
00:54:23 - Her strength is coming back.
01:03:45 - You're not listening.
01:09:17 - I have to go back.
01:09:24 - Challenge her, then.
01:09:26 - She may not play.
01:09:29 - She's got a thing.
01:09:34 - Ok.
01:11:22 - Why don't we play a game?

```

입출력 오류를 다루는 부분이 DELETE EQUAL 과 동일하므로, 잘못된 입력에 대한 사진은 생략하도록 한다.

EXIT

출력 결과는 다음과 같다. 결과는 log.txt 에 저장해 두었다.

```

00:54:23 - Her strength is coming back.
01:03:45 - You're not listening.
01:09:17 - I have to go back.
01:09:24 - Challenge her, then.
01:09:26 - She may not play.
01:09:29 - She's got a thing.
01:09:34 - Ok.
01:11:22 - Why don't we play a game?
01:11:26 - I know you're tired.
01:11:27 - Everybody's tired.
01:11:31 - What kind of game?
01:11:36 - A finding game.
01:11:38 - And what's the prize?
01:11:41 - My real life.
01:11:51 - What if I lose?
01:11:54 - If I lose, I'll be yours.
01:11:59 - And I'll be yours.
=====
===== PRINT =====
Section 3
00:52:19 - I'm going to bed.
00:52:20 - Right now!
00:52:21 - Bed? Before dinner?
00:52:23 - I'm really, really tired.
=====
[ ]
===== EXIT =====
Success
=====
"log.txt" 170L, 5153C

```

Visual Studio 의 디버그 툴에서 메모리 스냅샷 확인 방법에 대해서 이해하지 못했기 때문에, 프로그램 동작 시 메모리 누수 문제가 존재하는지는 확인하지 못하였다.

CONCLUSION

코드 구조를 개선하기 위해 작업한 내용

Flowchart 부분에서 앞서 서술했듯, 각 자료구조가 Manager 를 통하지 않고 직접적으로 데이터를 서로 주고받는 것을 피하고자 했는데, SECTION 명령어 작업 시 특히 그 부분이 어려웠다.

SECTION 명령어 수행 시, 우선적으로 SubtitleBST 를 In-Order Range Search (SubtitleBST :: SearchRange())를 수행하며, 재귀적으로 함수를 호출해야 하는데, 하나의 노드가 찾아지자마자 그것이 Manager 를 통해서 전달될 수가 없어 SubtitleBST 에서 SectionList 객체에 재귀를 진행하면서 직접 노드를 넘겨줘야하는 상황이 발생했다.

따라서 이 과정 중에서 생기는 노드들을 데이터구조실습 시간에 설명한 것과 다르게 중간에 하나의 버퍼를 두어야겠다고 생각했고, SubtitleBST 에서 찾아진 노드를 임시로 버퍼에 저장해 SectionList 에 저장하는 형식을 취하게 되었다. 이로써, Manager 에서 할당한 세 객체 사이에서는 Run()을 실행하며 데이터를 직접적으로 주고받는 일이 없도록 하였다. 이렇게 하면 세 객체 사이의 의존성이 상당히 줄어들게 설계되므로, 각각의 모듈을 이후에 쉽게 재사용할 수 있었다.

성능을 개선하기 위해서 기본적으로 상당수의 함수들을 inline 으로 처리하였으며, 입출력 부분에서 바뀌지 않는 변수들에 대해 const 키워드를 사용하려고 하였다.

코드관리를 위해 작업한 내용

코드 관리를 위해 Visual Studio 에서 Github 를 연동시켜 지속적으로 이전 버전의 개발 상황을 commit push 해가며 진행한 것이 크게 도움이 되었다.

또한 Debug 용도로 트리의 구조를 출력하는 PrintStrucure()함수를 중간중간에 사용해가면서 작업한 것이 개인적으로 상당히 도움이 되었다.

프로젝트를 진행하며 새로이 알게 된 내용

처음에 Time 의 $\gg \ll$ 연산자에 의해, LNK 2019, LNK 4042 에러가 지속적으로 발생하였는데, 모든 파일에 지속적으로 포함되는 헤더 파일 부분을 baseheader.h 로 분리하고 명확하게 헤더 파일의 구조를 분리한 이후로 에러가 발생하지 않게 되었다.

Time 이라는 자료형을 프로젝트 전반에 걸쳐 Class 로 만들어서 사용하고, $\gg \ll$ 연산자를 friend 함수로 클래스 외부에서 오버로딩하는 법에 대해 배워, 프로젝트 전체에서 Key 값을 비교하는 데 상당히 용이하게 사용하였다.