

데이터 구조 설계

1차 프로젝트

학번: 2018202084

이름: 김민재

- Introduction

자료구조를 활용하여 정보를 관리하는 프로그램을 구현하는 것을 목적으로 둔다. Data가 들어있는 파일의 이름, 나이, 정보 수집 일자, 가입 약관 종류를 읽어서 Member_Queue라고 하는 구조를 형성한다. 약관 종류는 4종류가 존재하며 유효기간은 각기 다르다. Queue에서 Pop 명령어를 실행하는 경우 데이터를 방출하고, 약관 종류와 회원 이름 기준으로 정렬된 내용을 각각 형성한 자료구조에 저장한다. Terms_List는 약관 종류 별로 node가 형성되고, 각각의 node는 약관 종류, 회원 수, 해당 약관의 BST pointer 정보를 가지게 된다. Terms_BST는 가입 약관 종류 별로 구성된다. BST의 node는 이름, 나이, 수집 일자, 만료 일자의 정보를 가진다. 만료 일자는 수집 일자에 유효기간이 더해지며, 각각의 BST는 만료 일자의 정보를 기준으로 정렬되어진다. Name_BST는 이름, 가입 약관 종류, 나이, 정보 수집 일자, 정보 만료일자를 가지는 Node로 구성되며, 이름을 기준으로 정렬된다.

Member_Queue의 형성은 data.txt 파일의 데이터를 활용하며, 저장되는 순서대로 방출된다. 회원들의 정보를 담는 Queue 클래스를 나타낸다. data.txt의 처음 줄부터 마지막까지 순서대로 저장되며, 입력 데이터는 이름, 나이, 정보 수집일자, 약관종류 순서의 형식을 가진다. Queue에 저장되는 data는 MemberQueueNode class로 구현되고 이름, 나이, 정보 수집일자, 약관 종류의 정보를 가지게 된다. Queue는 100의 크기를 일정하게 가져, 비어 있는데 POP을 하거나 가득 차 있는데 PUSH를 하면 프로그램은 종료된다. Queue에서 QPOP으로 방출되는 데이터는 List와 BST의 입력으로 사용하게 된다.

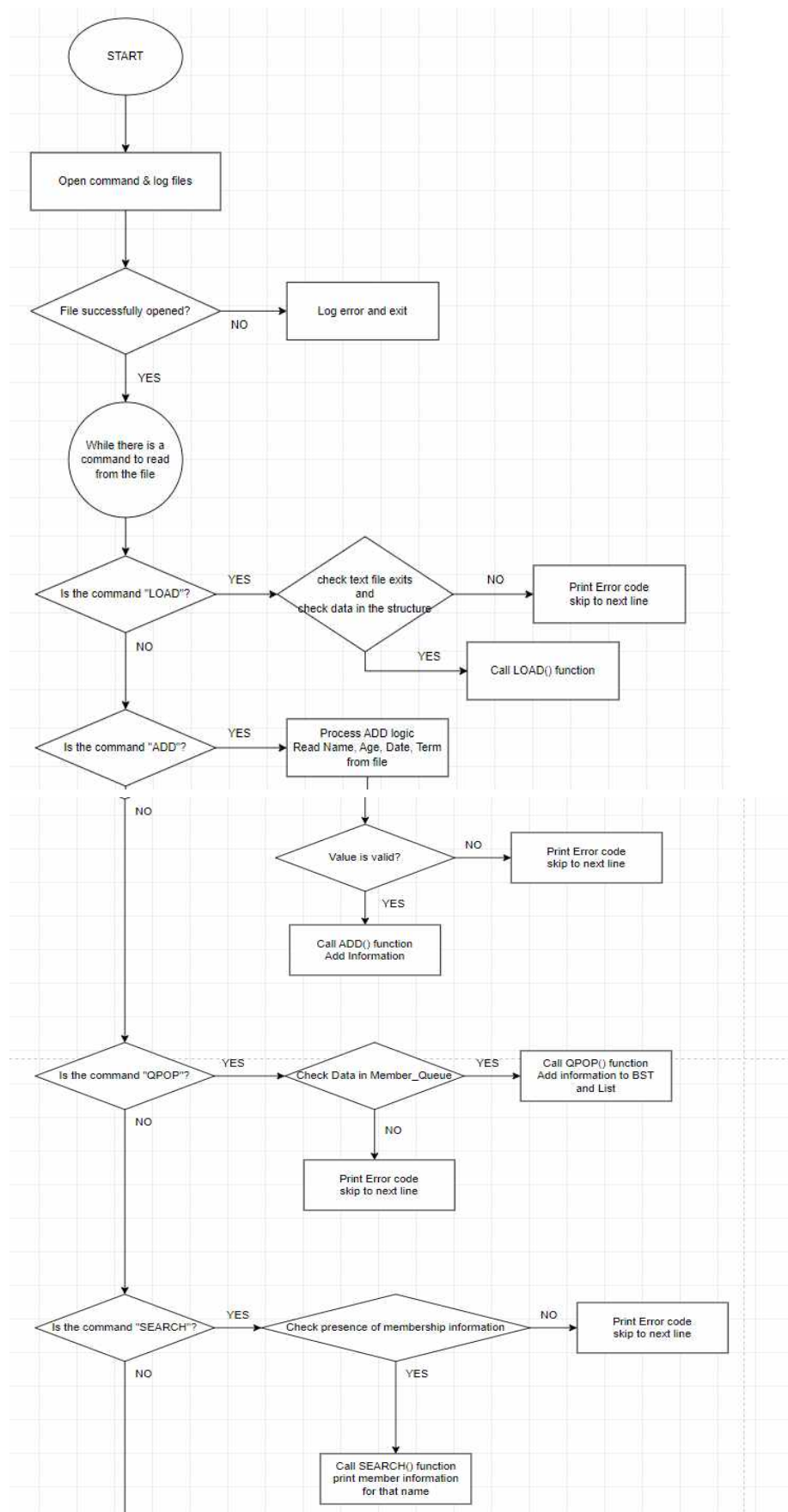
Terms_List는 Queue로부터 나온 data를 이용하여 구축하며, 입력된 약관 순서로 정렬되어진다. 약관 종류에 따라 Node로 연결되어진다. 해당 Node는 TermsListNode class로 구현하고, 약관 종류, 회원 수, 약관 BST pointer의 정보를 가지게 된다. 가입 약관을 확인하여 List에 없는 경우 Node를 추가한다. 반면에 있는 경우 Node의 회원 수를 증가시키도록 한다. DELETE를 통하여 BST Node의 삭제 연산이 진행된다면 Terms_List에서 해당 약관의 회원수를 감소시키고, 회원 수가 0이 된 경우 Node를 삭제하도록 한다.

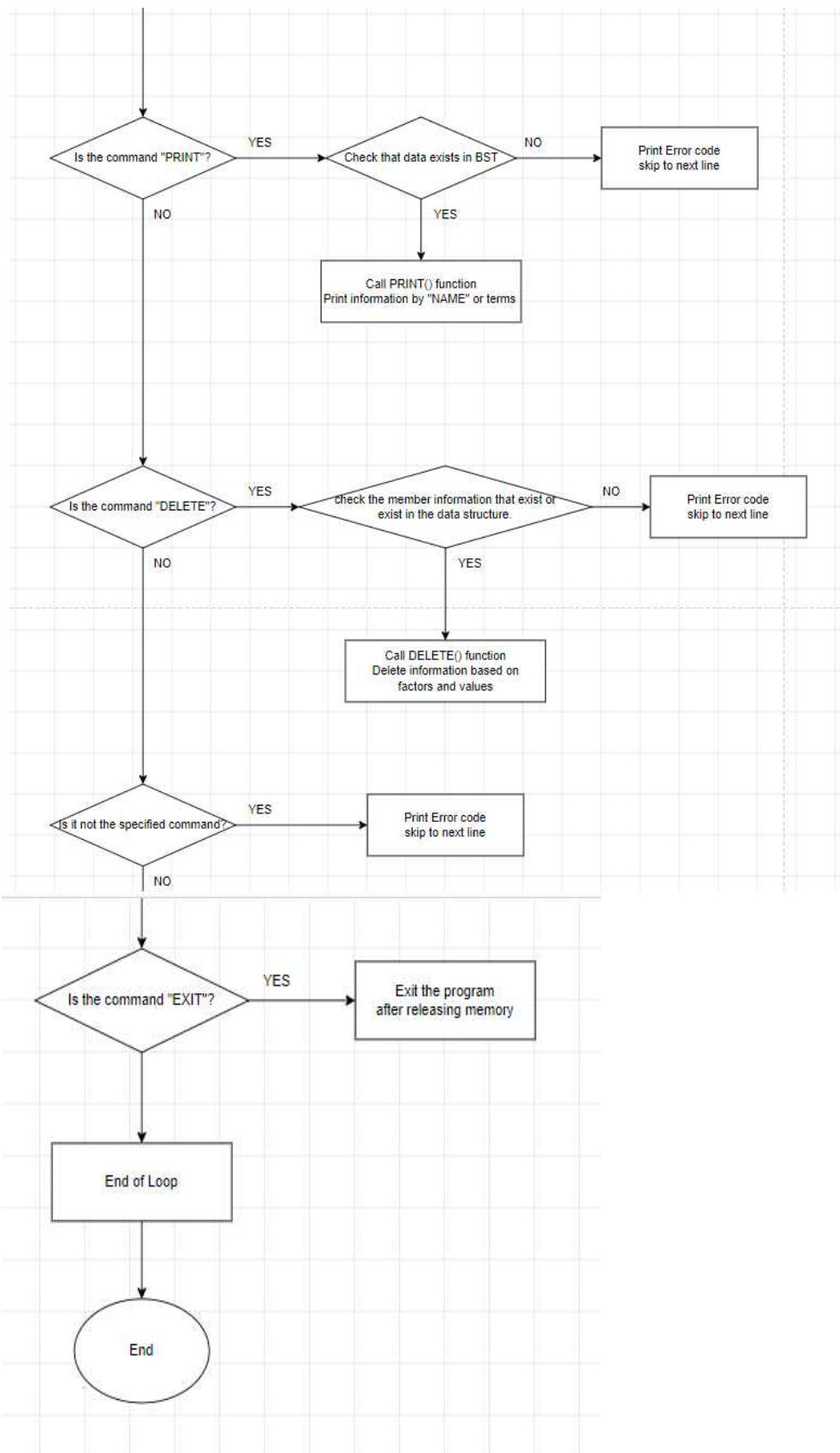
Terms_BST는 List의 가입약관에 따른 Node를 구성하고, Queue로부터 나온 data를 이용하여 형성한다. BST의 Node는 TermsBSTNode class이고, 이름, 나이, 정보 수집일자, 정보 만료일자를 가지게 된다. 정보 만료일자는 정보 수집일자에 가입약관에 따른 유효기간이 더해져서 계산 되어진다. BST의 연결 규칙으로는 부모 Node보다 정보 만료일자가 이전인 경우는 왼쪽이고, 같거나 이후이면 오른쪽 subtree에 위치하게 된다. Node를 제거하는 경우 양쪽 자식 Node가 모두 존재하면 오른쪽 자식 Node 중 가장 작은 Node가 제거 되는 위치로 이동하게 된다.

Name_BST의 경우 Queue로부터 나온 data를 이용하여 NameBSTNode class를 이용한다. 정보는 이름, 나이, 정보 수집일자, 정보 만료일자, 약관의 종류를 가진다. 정보 만료일자는 Terms_BST와 동일하게 정보 수집일자에 약관에 따른 유효기간이 더해져서 계산 된다. DELETE 명령어를 이용하여 Name_BST의 삭제 연산이 진행된다면 Terms_BST 또한 해당 Node를 삭제하게 된다. 연결 규칙은 부모 노드보다 이름이 사전에 따라 이전이면 왼쪽에 위치하고, 같거나 이후인 경우는 오른쪽 subtree에 위치하게 된다. Node를 제거하는 경우 양쪽 자식 Node가 모두 존재하면 오른쪽 자식 Node 중 가장 작은 Node가 제거 되는 위치로 이동하게 된다. 또한 여러 가지 명령어의 사용법이 존재한다. LOAD 명령어의 경우 txt 파일의 정보를 읽어온다. 정보가 존재하는 경우 Member_Queue 구조에 모두 저장하게 된다. 정상적

으로 읽어온 경우는 데이터를 출력하게 되며 이름/나이/정보 수집일자/약관 종류 형태로 출력한다. txt 파일이 존재하지 않거나 이미 데이터가 있는 경우는 에러 코드를 출력한다. ADD 명령어의 경우는 Member_Queue에 data를 직접적으로 추가하며 이름/나이/정보 수집일자/약관 종류 형태로 입력한다. 이 중 하나라도 없으면 에러 코드를 출력한다. QPOP 명령어의 경우 Member_Queue로부터 POP을 수행하여 Name_BST를 구성한다. front로부터 POP을 수행하여 data를 이용하여 자료 구조를 형성한다. Member_Queue에 데이터가 없으면 에러 코드를 출력한다. SEARCH 명령어는 Name_BST에 있는 회원을 찾아서 출력한다. 이름을 입력 받으면, 이름의 Node를 찾아서 정보를 출력한다. PRINT 명령어는 인자로 약관 종류가 입력되는 경우는 Terms_BST에 저장된 데이터를 출력하고, 이름을 입력하면 Name_BST에 저장된 데이터를 출력한다. in order 방식으로 탐색하여 출력을 진행한다. DELETE 명령어와 일자가 입력되면 Terms_BST에서 해당 일자보다 이전인 모든 노드를 제거한다. 노드가 삭제되면 해당 약관의 회원 수를 삭제 노드만큼 감소시켜 회원 수가 0인 노드는 삭제한다. DELETE 명령어와 이름이 같이 입력되면 Name_BST에서 해당 이름을 가진 노드를 제거한다. 삭제 시키고자 하는 정보가 없는 경우는 에러 코드를 출력한다.

- Flowchart





command file과 log file을 열어서 시작하게 된다. 파일이 성공적으로 열린 경우는 파일에 대한 읽기 명령이 있는 동안 명령어를 수행하고, 열리지 못하면 에러 메시지를 출력한다. LOAD 명령어는 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 에러 코드를 출력하고, 정상적으로 작동하면 LOAD 함수를 실행한다. ADD 명령어는 회원이름, 나이, 개인정보수집일자, 가입약관종류를 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력하고, 정상적으로 작동하면 ADD 함수를 실행한다. QPOP 명령어는 Member_Queue에 데이터가 존재하지 않을 시 에러 코드를 출력하고, 정상적으로 작동하면 QPOP 함수를 실행한다. SEARCH 명령어는 찾고자 하는 회원정보가 Name_BST에 존재하지 않는 경우 에러 코드를 출력하고, 정상적으로 작동하면 SEARCH 함수를 실행한다. PRINT 명령어는 BST에 데이터가 존재하지 않을 시 에러 코드를 출력하고, 정상적으로 작동하면 PRINT 함수를 실행한다. DELETE 명령어는 회원 정보가 존재하지 않거나, 자료구조에 데이터가 존재하지 않을 시 에러 코드를 출력하고, 정상적으로 작동하면 DELETE 함수를 실행한다. EXIT 명령어는 메모리를 해제하며, 프로그램을 종료한다.

- Algorithm

Manager class의 핵심 기능과 알고리즘을 보면 run 함수는 파일로부터 명령어를 읽어서, 각 명령어에 따른 작업을 수행한다. 이 후 명령어와 관련된 데이터를 처리하기 위해 문자열 비교와 파일 입출력을 사용한다. LOAD 함수는 파일에서 데이터를 로드하여 큐(Queue)에 저장한다. ADD 함수는 데이터를 큐에 추가한다. QPOP 함수는 큐에서 데이터를 POP하여 BST와 List에 추가한다. SEARCH 함수는 주어진 이름으로 데이터를 BST에서 검색하고, PRINT 함수는 BST 또는 List에서 데이터를 출력한다. DELETE 함수는 주어진 조건에 따라 데이터를 삭제한다.

[Manager]

```
|
|-----> [run] --> Parse Commands
|           |
|           |-----> LOAD() --> Queue
|           |
|           |-----> ADD() --> Queue
|           |
|           |-----> QPOP() --> [TermsBST], [NameBST], [TermsLIST]
|           |
|           |-----> SEARCH() --> [NameBST]
|           |
|           |-----> PRINT() --> [NameBST] or [TermsBST]
|           |
|           |-----> DELETE() --> [NameBST], [TermsBST], [TermsLIST]
|
|
|-----> [fcmd] (Command File Stream)
|
|-----> [flog] (Log File Stream)
```

run(): 주요 함수로, 명령 파일을 읽어 각각의 명령을 실행

LOAD(): "data.txt"에서 데이터를 읽어와 큐에 저장

ADD(): 사용자 정보를 큐에 추가

QPOP(): 큐에서 데이터를 빼와 Binary Search Tree와 List에 추가

SEARCH(): 주어진 이름을 가진 회원을 BST에서 검색

PRINT(): 이름 또는 약관 유형에 따라 BST의 데이터를 출력

DELETE(): 이름 또는 날짜에 따라 BST에서 데이터를 삭제

PrintSuccess(): 성공 메시지를 출력

PrintErrorCode(): 에러 코드에 따라 에러 메시지를 출력

각 함수는 다양한 데이터 구조와 상호 작용한다.

NameBST 클래스는 이름 기반의 이진 탐색 트리(BST)를 표현 하고, 각 기능은 주로 노드의 추가, 탐색, 출력 및 삭제를 위한 BST 연산을 포함한다.

[NameBST]

```
|
|-----> [Constructor/Destructor]: 클래스 생성 및 소멸
|
|-----> [insert(name, age, date, term)]: BST에 노드 삽입
|
|-----> [search(name)]: 주어진 이름으로 노드 검색
|
|-----> [deleteByName(name)]: 주어진 이름의 노드 삭제
|      |
|      |-> deleteNode(root, name) - 재귀적으로 노드 삭제
|      |-> minValueNode(node) - 주어진 노드의 서브트리에서
최소값 노드 찾기
|
|-----> [print(flog)]: BST 내용을 로그에 출력
|
|-----> [printMembersByTerm(flog, term)]: 주어진 약관으로 노드를 출력
|      |
|      |-> aa(startDate, term) - 주어진 시작 날짜와 약관으로 만료 날짜
계산
|
|-----> [isEmpty()]: BST가 비어있는지 확인
```

insert(): 주어진 정보로 노드를 생성하고 BST에 삽입

search(): 주어진 이름으로 BST를 검색하여 해당 노드를 반환

deleteByName(): 주어진 이름의 노드를 삭제

print(): BST의 노드를 로그에 중위 순회를 사용하여 노드를 정렬된 순서로 출력
printMembersByTerm(): 주어진 약관 유형에 따라 노드를 로그에 출력
isEmpty(): BST가 비어 있는지 확인

BST의 기본적인 원칙을 따르고 있다. 시스템에서 필요한 추가 기능들을 포함하고 있다. NameBST는 TermsBST 및 TermsLIST와 상호 작용하며 데이터를 동기화 하고 있다.

TermsBST 클래스는 약관 기반의 이진 탐색 트리(BST)를 표현한다. 주된 기능들은 노드의 추가, 탐색, 출력 및 삭제와 관련이 있다.

[TermsBST]

```
|
|-----> [Constructor/Destructor]: 클래스 생성 및 소멸
|
|-----> [insert(name, age, collectionDate, term)]: BST에 노드 삽입
|      |
|      |-> insert(node, name, age, date, term) - 재귀적인 삽입
|
|-----> [search(term)]: 주어진 약관으로 노드 검색
|
|-----> [deleteMember(name)]: 주어진 이름의 노드 삭제
|      |
|      |-> deleteMemberRecursive(node, name, wasDeleted) -
재귀적으로 노드 삭제
|      |-> minValueNode(node) - 주어진 노드의 서브트리에서 최소값
노드 찾기
|
|-----> [deleteBeforeDate(expirationDate)]: 주어진 만료 날짜 이전의 노드
삭제
|      |
|      |-> deleteBeforeDateRecursive(node, givenDate, count) - 재귀적인
삭제
|
```

```

|-----> [print()]: BST 내용을 콘솔에 출력
|
|      |-> inOrderTraversal(node, flog, term) - 중위 순회로 출력
|
|-----> [isEmpty(term)]: 주어진 약관의 노드가 비어있는지 확인
|
|-----> [calculateExpirationDate(collectionDate, term)]: 약관의 만료 날짜
계산

```

insert(): 주어진 정보로 노드를 생성하고 BST에 삽입
 search(): 주어진 약관으로 BST를 검색하여 해당 노드를 반환
 deleteMember(): 주어진 이름의 노드를 삭제
 deleteBeforeDate(): 주어진 만료 날짜 이전의 노드를 삭제
 print(): BST의 노드를 콘솔에 출력
 isEmpty(): 주어진 약관의 노드가 비어있는지 확인

TermsLIST 클래스는 연결 리스트(linked list)를 사용하여 약관 데이터를 관리 한다.

[TermsLIST]

```

|
|-----> [Constructor] --> 초기화
|
|-----> [Destructor] --> 리스트의 모든 노드 메모리 해제
|
|-----> [getHead] --> head 노드 반환
|
|-----> [insert] --> 리스트의 끝에 새로운 노드 삽입
|
|-----> [search] --> 주어진 이름으로 연결 리스트에서 노드 검색
|
|-----> [deleteMember] --> 주어진 이름을 가진 회원 삭제

```

TermsLIST는 단일 연결 리스트를 사용한다. 각 노드는 약관 유형, 이름, 나이, 수집 날짜 등의 데이터를 포함하고, 데이터는 리스트의 끝에 추가된다. 노드 검색 시 처음 발견된 노드를 반환하고, deleteMember 함수는 주어진 이름을 가진 노드를 찾아 삭제한다.

- Result Screen

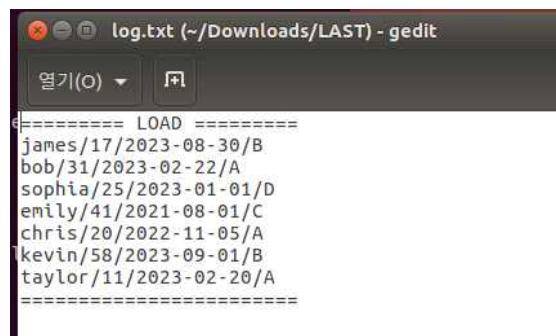
명령어를 수행하기 전에 compile을 하기 위하여 make를 입력을 하였는데, #pragma once 경고가 발생하였다. run을 실행하기에는 문제가 없었다.

```
os2018202084@ubuntu: ~/Downloads/LAST
os2018202084@ubuntu:~/Downloads/LAST$ make
make: 경고: 'TermsBST.cpp' 파일의 변경 시각(3334)이 미래입니다
g++ -std=c++11 -g -o run NameBST.cpp MemberQueue.cpp TermsBST.cpp TermsList.cpp
Manager.cpp main.cpp NameBST.h MemberQueue.h TermsBST.h NameBSTNode.h MemberQueu
eNode.h TermsBSTNode.h TermsList.h Manager.h TermsListNode.h
NameBST.h:1:9: warning: #pragma once in main file
#pragma once
^
MemberQueue.h:1:9: warning: #pragma once in main file
#pragma once
^
TermsBST.h:1:9: warning: #pragma once in main file
#pragma once
^
NameBSTNode.h:1:9: warning: #pragma once in main file
#pragma once
^
MemberQueueNode.h:1:9: warning: #pragma once in main file
#pragma once
^
TermsBSTNode.h:1:9: warning: #pragma once in main file
#pragma once
^
TermsList.h:1:9: warning: #pragma once in main file
#pragma once
^
Manager.h:1:9: warning: #pragma once in main file
#pragma once
^
TermsListNode.h:1:9: warning: #pragma once in main file
#pragma once
^
make: 경고: 시계가 잘못되었음이 발견되었습니다. 빌드가 불안전할 수 있습니다.
os2018202084@ubuntu:~/Downloads/LAST$ ./run
os2018202084@ubuntu:~/Downloads/LAST$
```

* LOAD 명령어

```
data.txt (~/.Downloads/LAST) - gedit
열기(O)  [icon]
data.txt x
james 17 2023-08-30 B
bob 31 2023-02-22 A
sophia 25 2023-01-01 D
emily 41 2021-08-01 C
chris 20 2022-11-05 A
kevin 58 2023-09-01 B
taylor 11 2023-02-20 A
```

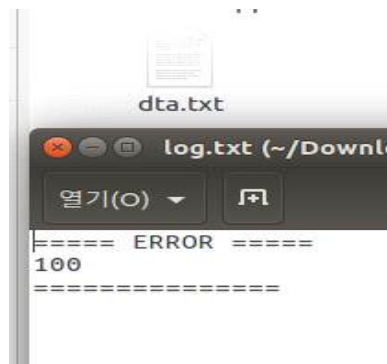
```
command.txt (~/.Downloads/LAST) - gedit
열기(O)  [icon]
LOAD
```

A screenshot of a gedit text editor window titled 'log.txt (~/Downloads/LAST) - gedit'. The window contains the following text:

```
===== LOAD =====  
james/17/2023-08-30/B  
bob/31/2023-02-22/A  
sophia/25/2023-01-01/D  
emily/41/2021-08-01/C  
chris/20/2022-11-05/A  
kevin/58/2023-09-01/B  
taylor/11/2023-02-20/A  
=====
```

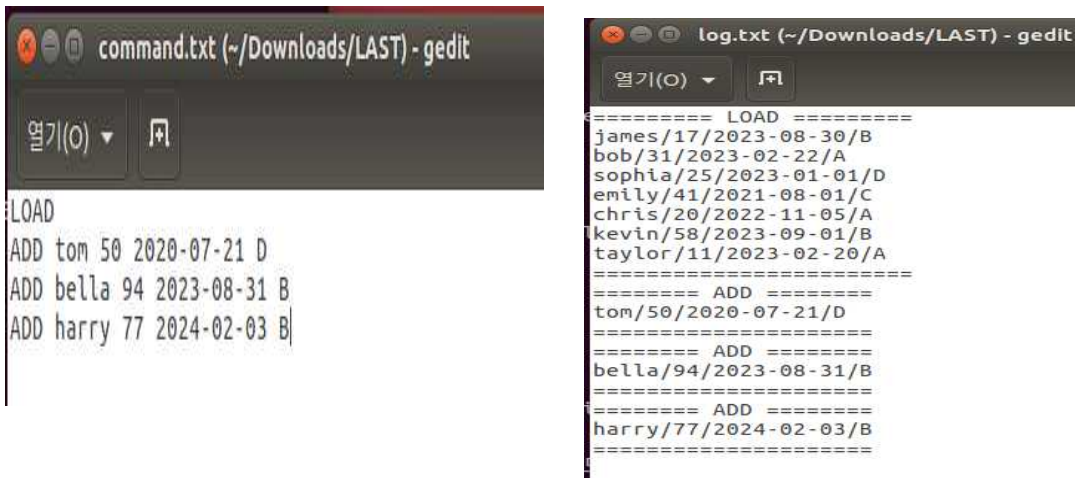
data.txt 파일에 데이터 정보가 존재하며, command.txt에서 LOAD 명령어가 작성되어 있다. run을 수행하면 log.txt에 결과가 정상적으로 출력된다.

- 예외 처리



data.txt 파일을 dta.txt 파일로 이름을 바꾼 후 실행을 시키니 에러코드를 출력하도록 하였다.

* ADD 명령어



The first screenshot shows a gedit window titled 'command.txt (~Downloads/LAST) - gedit'. The content of the file is:

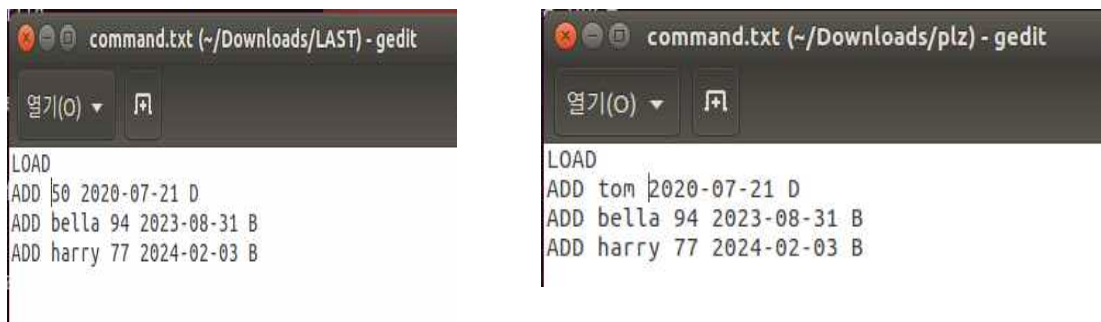
```
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
```

The second screenshot shows a gedit window titled 'log.txt (~Downloads/LAST) - gedit'. The content of the file is:

```
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
===== ADD =====
tom/50/2020-07-21/D
===== ADD =====
bella/94/2023-08-31/B
===== ADD =====
harry/77/2024-02-03/B
=====
```

command.txt 파일에 추가하고자 하는 데이터의 정보를 입력 시킨 후 run을 실행하여 log.txt 파일을 확인한 결과 정상적으로 출력이 되고 있는 것을 확인할 수 있었다.

- 예외 처리



The first screenshot shows a gedit window titled 'command.txt (~Downloads/LAST) - gedit'. The content of the file is:

```
LOAD
ADD 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
```

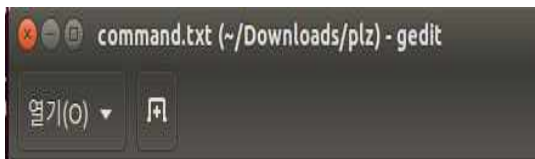
The second screenshot shows a gedit window titled 'command.txt (~Downloads/plz) - gedit'. The content of the file is:

```
LOAD
ADD tom 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
```

이름이 누락 된 경우와 나이가 누락 된 경우

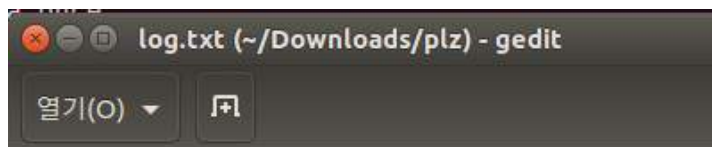


```
LOAD
ADD tom 50 b
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
```



```
LOAD
ADD tom 50 2020-07-21
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
```

개인정보 수집 일자가 누락된 경우와 약관이 누락된 경우



```
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
=====
===== ERROR =====
200
=====

===== ADD =====
bella/94/2023-08-31/B
=====
===== ADD =====
harry/77/2024-02-03/B
=====
```

1가지가 누락된다면 위와 같이 에러코드를 출력하게 된다.

* QPOP 명령어



```

command.txt (~/Downloads/plz) - gedit
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP

log.txt (~/Downloads/plz) - gedit
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
===== ADD =====
tom/50/2020-07-21/D
===== ADD =====
bella/94/2023-08-31/B
===== ADD =====
harry/77/2024-02-03/B
===== QPOP =====
Success
=====
  
```

command.txt에 QPOP 명령어를 작성하여, Member_Queue에서 POP하여 Terms_List & Terms_BST와 Name_BST를 구성하도록 하였다.

- 예외 처리



```

command.txt (~/Downloads/plz) - gedit
LOAD
QPOP

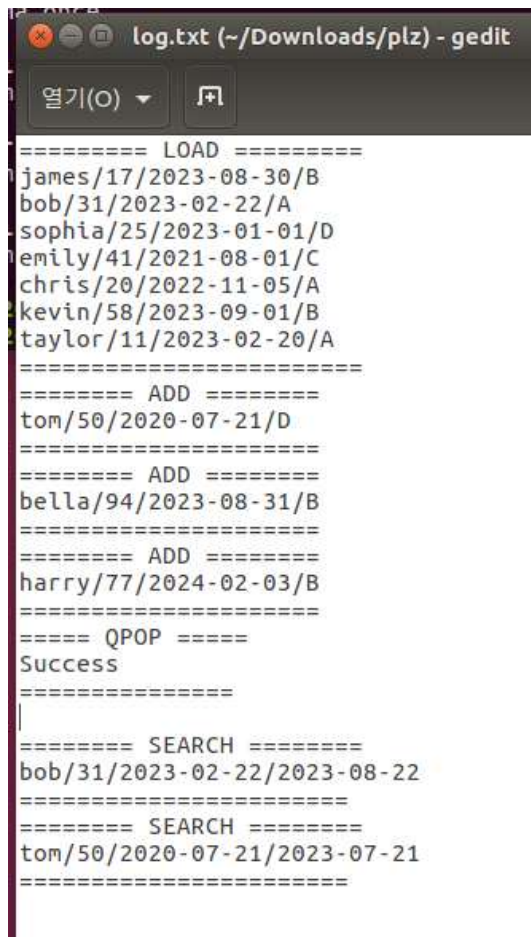
log.txt (~/Downloads/plz) - gedit
===== LOAD =====
===== ERROR =====
300
=====
  
```

Member_Queue에 데이터가 없는데, QPOP을 수행하면 에러 코드를 출력하도록 하였다.

* SEARCH 명령어



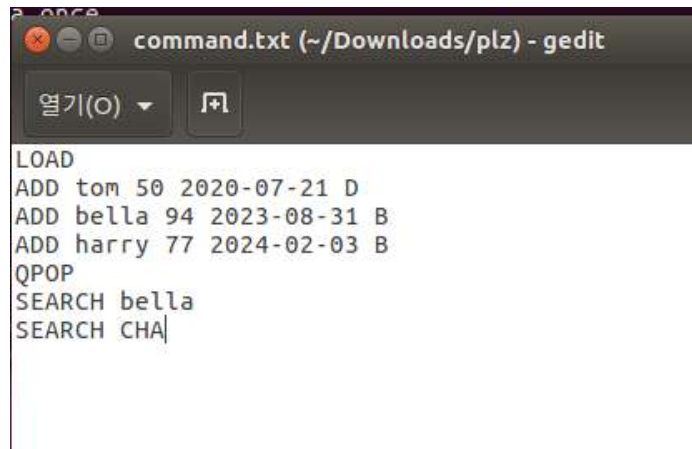
```
command.txt (~Downloads/plz) - gedit
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP
SEARCH bob
SEARCH tom
```



```
log.txt (~Downloads/plz) - gedit
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
=====
===== ADD =====
tom/50/2020-07-21/D
=====
===== ADD =====
bella/94/2023-08-31/B
=====
===== ADD =====
harry/77/2024-02-03/B
=====
===== QPOP =====
Success
=====
|
===== SEARCH =====
bob/31/2023-02-22/2023-08-22
=====
===== SEARCH =====
tom/50/2020-07-21/2023-07-21
=====
```

인자로 탐색하고자 하는 회원의 이름을 받아서 결과를 출력하도록 한다. bob과 tom은 존재하기에 정상적으로 출력되는 것을 확인할 수 있다.

- 예외 처리



```
command.txt (~/Downloads/plz) - gedit
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP
SEARCH bella
SEARCH CHA
```



```
log.txt (~/Downloads/plz) - gedit
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
=====
===== ADD =====
tom/50/2020-07-21/D
=====
===== ADD =====
bella/94/2023-08-31/B
=====
===== ADD =====
harry/77/2024-02-03/B
=====
===== QPOP =====
Success
=====
===== SEARCH =====
bella/94/2023-08-31/2024-08-31
=====
===== ERROR =====
400
=====
```

bella는 Name_BST에 존재하기에 출력되며, CHA는 없기에 에러 코드가 발생한다.

* PRINT 명령어

```
command.txt (~/Downloads/plz) - gedit
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP
SEARCH bob
SEARCH tom
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
```

```
log.txt (~/Downloads/plz) - gedit
=====
===== QPOP =====
Success
=====

===== SEARCH =====
bob/31/2023-02-22/2023-08-22
=====
===== SEARCH =====
tom/50/2020-07-21/2023-07-21
=====
===== PRINT =====
Name_BST
bella/94/2023-08-31/2024-08-31
bob/31/2023-02-22/2023-08-22
chris/20/2022-11-05/2023-05-05
emily/41/2021-08-01/2023-08-01
harry/77/2024-02-03/2025-02-03
james/17/2023-08-30/2024-08-30
kevin/58/2023-09-01/2024-09-01
sophia/25/2023-01-01/2026-01-01
taylor/11/2023-02-20/2023-08-20
tom/50/2020-07-21/2023-07-21
=====
===== PRINT =====
Terms_BST A
bob/31/2023-02-22
chris/20/2022-11-05
taylor/11/2023-02-20
=====
===== PRINT =====
Terms_BST B
bella/94/2023-08-31
harry/77/2024-02-03
james/17/2023-08-30
kevin/58/2023-09-01
=====
===== PRINT =====
Terms_BST C
emily/41/2021-08-01
=====
===== PRINT =====
Terms_BST D
sophia/25/2023-01-01
tom/50/2020-07-21
=====
```


인자로 약관이 들어오면 가입약관 Terms_BST의 저장된 데이터들을 출력하고, NAME이 입력되면 Name_BST의 저장된 데이터들을 출력한다. 위는 정상적으로 모든 결과가 출력되는 것을 확인할 수 있다.

- 예외 처리



```
command.txt (~/Downloads/plz) - gedit
열기(O)
LOAD
QPOP
SEARCH bob
SEARCH tom
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
```

```
LOAD
QPOP
SEARCH bob
SEARCH tom
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
```



```
log.txt (~/Downloads/plz) - gedit
열기(O)
===== LOAD =====
===== ERROR =====
300
=====
===== ERROR =====
400
=====
===== ERROR =====
400
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
```

```
===== LOAD =====
===== ERROR =====
300
=====
===== ERROR =====
400
=====
===== ERROR =====
400
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
```

data.txt에는 회원 정보가 없고, 추가적으로 정보를 추가하지도 않았기에, BST에는 정보가 없다. 따라서 PRINT 명령어를 수행하니 에러코드가 발생했다.

* DELETE 명령어

```
command.txt (~/Downloads/plz) - gedit
열기(O)  [icon]

LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP
SEARCH bob
SEARCH tom
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
DELETE NAME emily
PRINT NAME
PRINT C
DELETE DATE 2024-09-01
PRINT NAME
PRINT A
PRINT B
PRINT D
EXIT
```

```
(~/Downloads/plz) - gedit
열기(O)  [icon]

===== PRINT =====
Terms_BST D
sophia/25/2023-01-01
tom/50/2020-07-21
=====
===== DELETE =====
Success
=====
===== PRINT =====
Name_BST
bella/94/2023-08-31/2024-08-31
bob/31/2023-02-22/2023-08-22
chris/20/2022-11-05/2023-05-05
harry/77/2024-02-03/2025-02-03
james/17/2023-08-30/2024-08-30
kevin/58/2023-09-01/2024-09-01
sophia/25/2023-01-01/2026-01-01
taylor/11/2023-02-20/2023-08-20
tom/50/2020-07-21/2023-07-21
=====
===== ERROR =====
600
=====
===== DELETE =====
Success
=====

===== PRINT =====
Name_BST
harry/77/2024-02-03/2025-02-03
kevin/58/2023-09-01/2024-09-01
sophia/25/2023-01-01/2026-01-01
=====
===== ERROR =====
600
=====

===== PRINT =====
Terms_BST B
harry/77/2024-02-03/2025-02-03
kevin/58/2023-09-01/2024-09-01
=====

===== PRINT =====
Terms_BST D
sophia/25/2023-01-01/2026-01-01
=====

===== EXIT =====
Success
=====
```

주어진 command.txt 파일을 이용하여 DELETE 명령어를 수행하였다. 그리하여 결과에 emily가 적힌 노드를 삭제하고, 2024-09-01 이전인 만료일자를 포함하는 노드를 삭제하였다.

- 예외 처리

```
command.txt (~/Downloads/plz) - ged
열기(O)  [F2]
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP
SEARCH bob
SEARCH tom
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
DELETE NAME ABC
PRINT NAME
PRINT C
DELETE DATE 2024-09-01
PRINT NAME
PRINT A
PRINT B
PRINT D
EXIT
```

```
=====
===== PRINT =====
Terms_BST D
sophia/25/2023-01-01
tom/50/2020-07-21
=====
===== ERROR =====
600
=====
===== PRINT =====
Name_BST
bella/94/2023-08-31/2024-08-31
bob/31/2023-02-22/2023-08-22
chris/20/2022-11-05/2023-05-05
emily/41/2021-08-01/2023-08-01
harry/77/2024-02-03/2025-02-03
james/17/2023-08-30/2024-08-30
kevin/58/2023-09-01/2024-09-01
sophia/25/2023-01-01/2026-01-01
taylor/11/2023-02-20/2023-08-20
tom/50/2020-07-21/2023-07-21
=====
===== PRINT =====
Terms_BST C
emily/41/2021-08-01
=====
===== DELETE =====
Success
=====
===== PRINT =====
Name_BST
harry/77/2024-02-03/2025-02-03
kevin/58/2023-09-01/2024-09-01
sophia/25/2023-01-01/2026-01-01
=====
===== ERROR =====
600
=====
===== PRINT =====
Terms_BST B
harry/77/2024-02-03/2025-02-03
kevin/58/2023-09-01/2024-09-01
=====
===== PRINT =====
Terms_BST D
sophia/25/2023-01-01/2026-01-01
=====
```

DELETE NAME ABC를 하였지만, ABC라는 이름의 노드는 없기에 에러코드를 출력하고, 이후의 명령어를 실행시켜 예외처리를 진행한다.

* EXIT 명령어

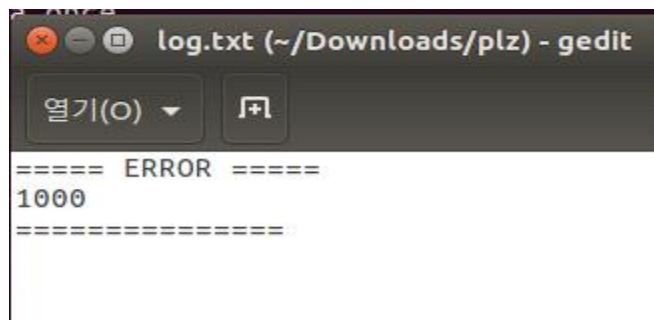
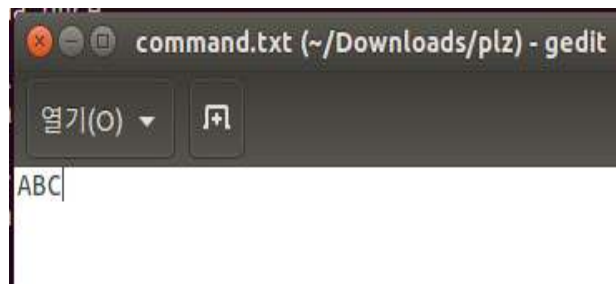
```
LOAD
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
QPOP
SEARCH bob
SEARCH tom
PRINT NAME
EXIT|
```



```
log.txt
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
=====
===== ADD =====
tom/50/2020-07-21/D
=====
===== ADD =====
bella/94/2023-08-31/B
=====
===== ADD =====
harry/77/2024-02-03/B
=====
===== QPOP =====
Success
=====
===== SEARCH =====
bob/31/2023-02-22/2023-08-22
=====
===== SEARCH =====
tom/50/2020-07-21/2023-07-21
=====
===== PRINT =====
Name_BST
bella/94/2023-08-31/2024-08-31
bob/31/2023-02-22/2023-08-22
chris/20/2022-11-05/2023-05-05
emily/41/2021-08-01/2023-08-01
harry/77/2024-02-03/2025-02-03
james/17/2023-08-30/2024-08-30
kevin/58/2023-09-01/2024-09-01
sophia/25/2023-01-01/2026-01-01
taylor/11/2023-02-20/2023-08-20
tom/50/2020-07-21/2023-07-21
=====
===== EXIT =====
Success
=====
```

EXIT 명령어 수행 시 프로그램 종료를 확인할 수 있다.

* 잘못된 명령어



정의된 명령어가 아닌 잘못된 명령어를 입력한 경우에는 에러코드(1000)가 출력이 되는 것을 확인할 수 있다.

- Consideration

본 프로젝트를 구현하며 여러 자료구조를 이용할 수 있었다. 조건에 맞도록 명령어를 구현해야 했으며, 진행하면서 다양한 문제점으로 인해 오랜 시간이 걸리게 되었다. BST, Linked list, queue 구조를 활용하며 프로그램 구현을 진행했지만, 중간 중간 부족한 점이 생겨서 이를 컴파일 하고, 수정하는 것에 있어서 모르는 점 또한 알게 된 부분이 있었다. LOAD 명령어 구현을 하며 100개의 정보 이상이 들어 있는 경우 101번째부터는 출력하지 않고, 에러 메시지를 발생하게 하여 진행하지 못하도록 하는 것에 어려움이 있었지만, Member_Queue의 구조를 다시 한 번 확인하여 수정을 진행하였다. ADD 명령어를 구현하며, 입력 되는 이름, 나이, 수집일자, 만료 일자 중에 1개라도 누락이 있는 경우 에러 메시지를 발생시켜야 하기에 모든 경우에 대하여 예외처리를 진행하는데, 꽤 시간이 걸렸다. 이후 QPOP 명령어는 Member_Queue에 데이터가 없으면 에러 코드를 출력해야 하기에 Queue에 값이 있는지 확인을 하고, POP을 수행하면 Terms_List와 Terms_BST에 POP 한 값을 구성해야 하기에 이중적인 코드의 수정이 필요하였다. SEARCH와 PRINT 명령어의 경우는 Name_BST에서 입력 받은 명령어를 출력을 하고, PRINT 명령어로 NAME이 입력 되면 해당 내용을 출력하는 프로그램을 구현하는 것에 예외 처리 및 작동을 같이 진행하도록 하였다. DELETE 명령어의 경우 List와 BST에 저장된 데이터를 제거하는 명령어 이기에 해당 일자보다 개인정보만료일자가 이전인 노드 들을 확인하여 제거를 수행하는 것에 비교적 시간이 걸렸으며, Terms_BST 노드 삭제가 되면 Name_BST에서도 해당 되어지는 회원 노드를 동시 삭제에 문제가 있었기에 장시간을 투자하였다. 본 프로젝트를 하며 자료구조의 개념을 이해하고, 수행할 수 있었기에 프로젝트에 접근할 수 있었던 기회가 될 수 있었던 것 같다.