

# DS\_Project1

컴퓨터정보공학부 2022202012 손민

## 1. Introduction

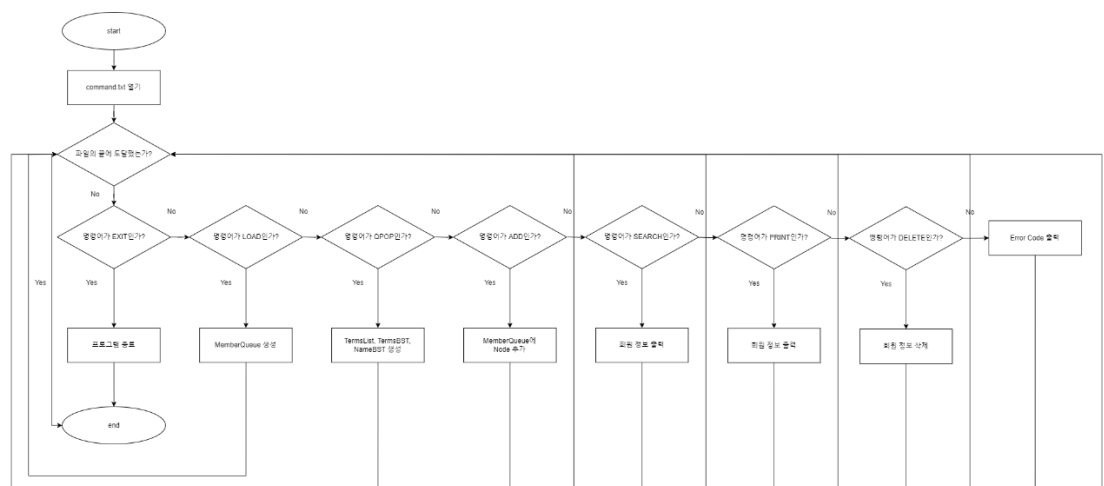
회원 정보를 저장하고, 분류하는 개인정보 관리 프로그램을 구현하는 프로젝트이다. 프로그램은 Queue, Linked List, Binary Search Tree의 자료 구조를 사용해 회원 정보를 저장한다. 그리고, LOAD, ADD, QPOP, SEARCH, PRINT, DELETE, EXIT의 7가지의 명령어들로 프로그램이 실행된다.

회원 정보가 저장되는 자료구조의 형태는 다음과 같다. data.txt 파일에 담긴 회원 정보를 읽어와 Queue(Member\_Queue)를 구축한다. Queue에 저장된 정보를 pop하여 가입약관종류를 기준으로 정렬된 Linked List(Terms\_List)와 BST(Terms\_BST), 회원 이름을 기준으로 정렬된 BST(Name\_BST)를 구축한다.

프로그램의 명령어는 다음과 같다. LOAD 명령어를 실행하면 data.txt 파일에 회원 정보를 읽어 Member\_Queue에 저장한다. ADD 명령어를 실행하면 Member\_Queue에 직접 data를 추가할 수 있다. QPOP 명령어를 실행하면 Member\_Queue에 있던 data들이 pop되며 Terms\_List, Terms\_BST, Name\_BST를 구축한다. SEARCH 명령어를 실행하면 Name\_BST에서 원하는 회원의 정보를 찾아 출력한다. PRINT 명령어를 실행하면 Terms\_BST 혹은 Name\_BST에 저장된 회원 정보를 찾아 출력한다. DELETE 명령어를 실행하면 Terms\_List, Terms\_BST, Name\_BST에 저장된 회원 정보를 삭제한다. 마지막으로, EXIT 명령어를 실행하면 프로그램이 종료된다.

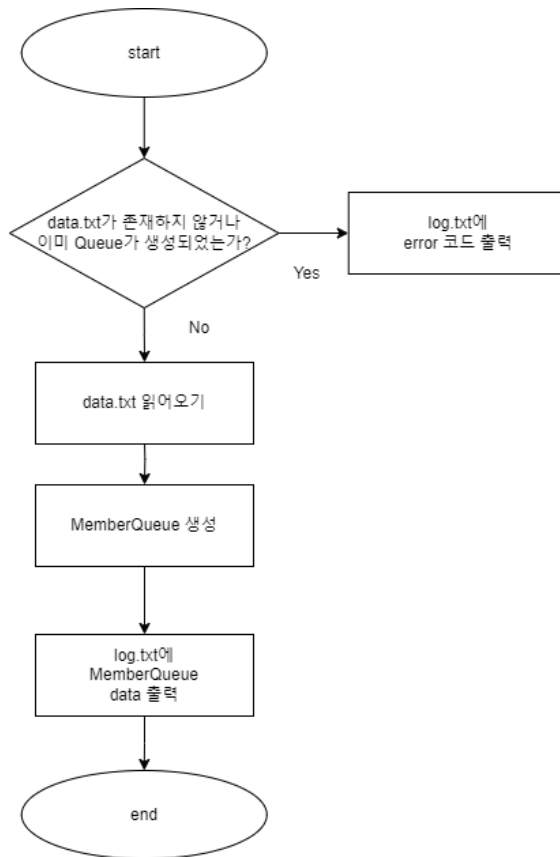
## 2. Flowchart

### A. 프로그램 전체



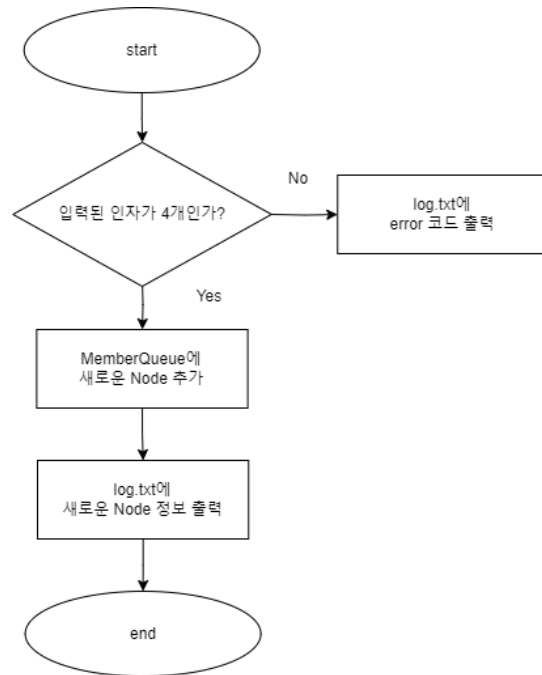
프로그램 전체 flow chart이다. 명령어가 담긴 command.txt 파일을 열어 한 줄 씩 읽어주고, 명령어에 맞게 동작을 수행한다.

#### B. LOAD



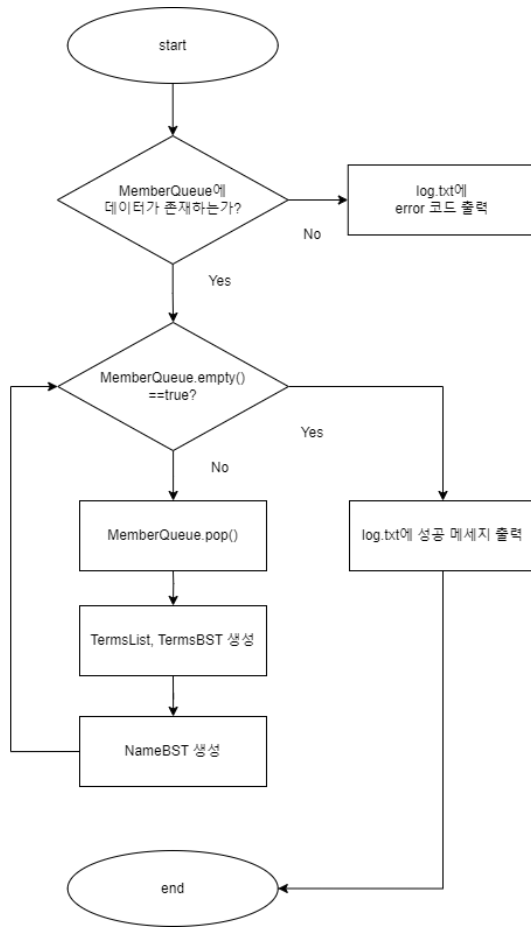
LOAD 명령어의 flow chart이다. data.txt가 존재하지 않거나 이미 Member\_Queue가 존재한다면 error 코드를 출력한다. 아닌 경우 data.txt를 읽어와 Member\_Queue를 생성하고 Member\_Queue에 저장된 정보를 출력하도록 한다.

#### C. ADD



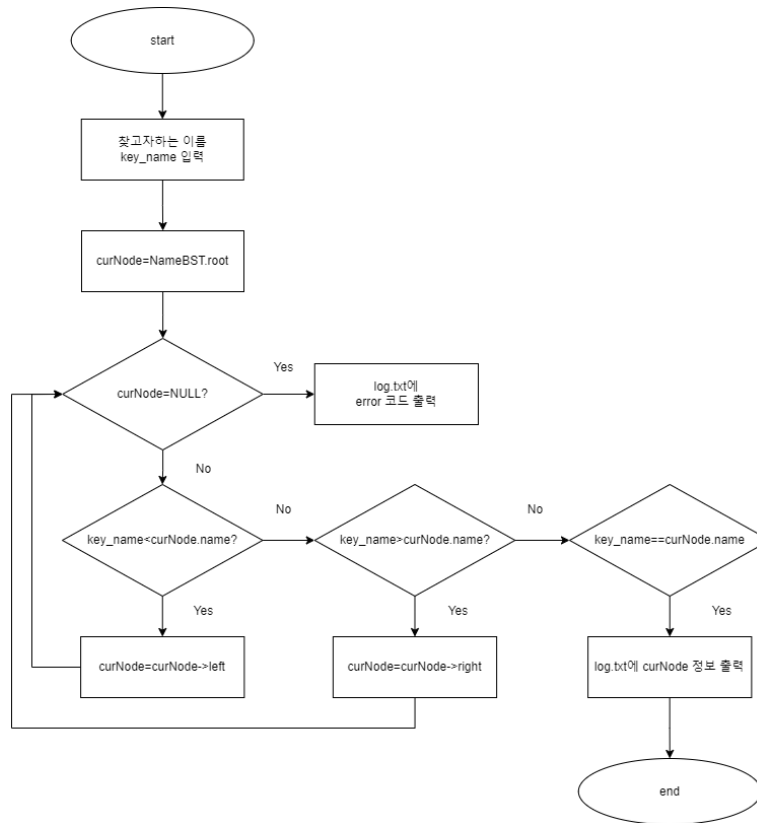
ADD 명령어의 flow chart이다. 입력된 인자가 4개가 아닌 경우(데이터 정보 부족), 에러 코드를 출력한다. 4개의 인자가 모두 입력된 경우, Member\_Queue에 새로운 노드를 추가하고 그 정보를 출력한다.

#### D. QPOP



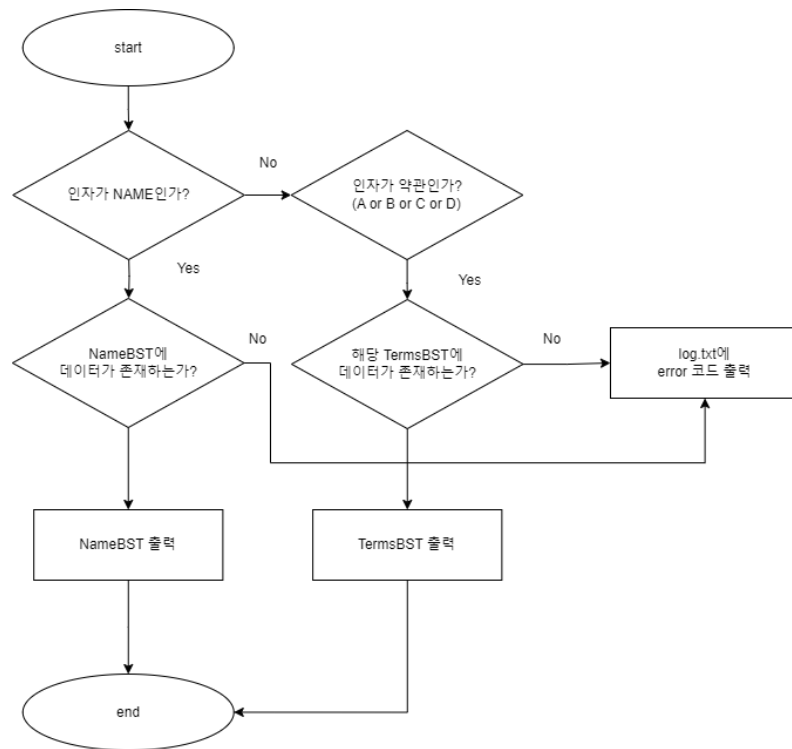
QPOP 명령어의 flow chart이다. Member\_Queue에 데이터가 존재하지 않는다면 error 코드를 출력한다. Member\_Queue가 empty 상태가 될 때까지 Member\_Queue에서 데이터를 pop하여, Terms\_List, Terms\_BST, Name\_BST를 생성한다. Member\_Queue의 데이터가 모두 pop된 경우(empty 상태인 경우), 반복문을 탈출하고 성공 메시지를 출력한다.

#### E. SEARCH



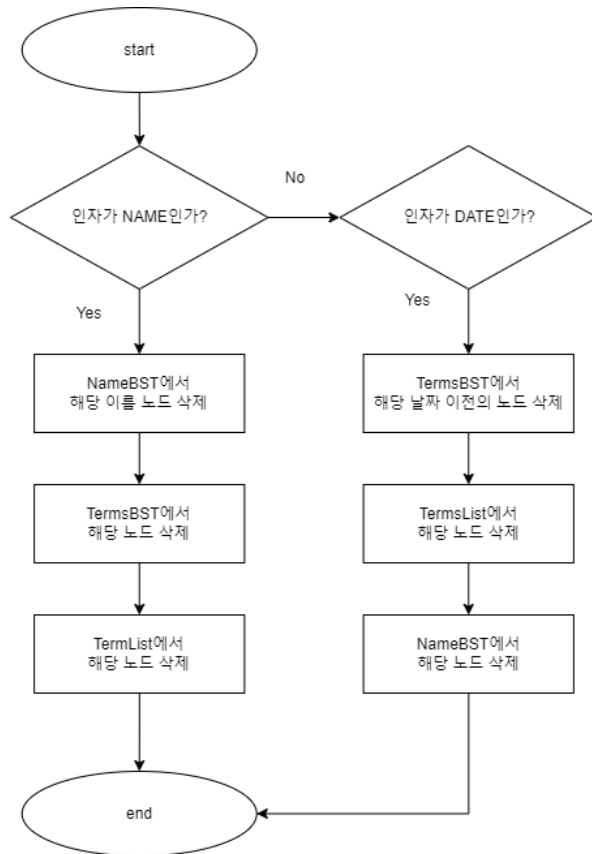
SEARCH 명령어의 flow chart이다. 찾고자 하는 이름(key\_name)을 인자로 입력 받는다. Name\_BST의 root 노드부터 시작하여 현재 노드(curNode)의 이름 정보와 key\_name을 비교한다. curNode가 NULL이라면 search에 실패한 것이므로 error 코드를 출력한다. key\_name이 curNode의 이름 정보보다 작다면, curNode를 curNode의 left child로 바꾼다. key\_name이 curNode의 이름 정보보다 크다면, curNode를 curNode의 right child로 바꾼다. key\_name과 curNode의 이름 정보가 일치할 때까지 해당 과정을 반복한다. key\_name이 curNode의 이름 정보와 같다면, search에 성공하였으므로 성공 메시지를 출력한다.

#### F. PRINT



PRINT 명령어의 flow chart이다. 인자로 NAME 혹은 약관(A/B/C/D)를 받는다. 인자가 NAME인 경우 NameBST에 데이터가 존재하는지 확인한다. 인자가 약관인 경우 해당 TermsBST에 데이터가 존재하는지 확인한다. NameBST 혹은 TermsBST에 데이터가 존재하지 않는 경우 error 코드를 출력한다. 데이터가 존재한다면 BST의 데이터를 출력한다.

## G. DELETE



DELETE 명령어의 flow chart이다. 인자로 NAME 혹은 DATE를 받는다. 인자가 NAME인 경우 NameBST에 데이터가 존재하는지 확인한다. 존재하지 않는 경우 에러 코드를 출력한다. 존재하는 경우, 해당 NAME을 가지는 노드를 NameBST에서 삭제한다. TermsBST와 TermsList에서도 해당 노드를 삭제한다. 인자가 DATE인 경우, TermsBST에서 해당 DATE 값보다 작은 개인정보 만료일자를 가지는 노드들을 삭제한다. TermsList와 NameBST에서도 해당 노드를 삭제한다.

### 3. Algorithm

#### A. 개인정보 만료일자 계산

```

개인정보 수집일자 정보를 year, month에 복사
인자로 약관의 종류(A-6개월, B-12개월, C-24개월, D-36개월)를 입력 받음
if (약관==A)
    month=month+6
    if (month-12>0)
        month=month-12
        year=year+1
else if (약관==B)

```

```

        year=year+1
else if (약관==C)
        year=year+2
else if (약관==D)
        year=year+3

```

Terms\_BST와 Name\_BST의 정보인 개인정보 만료일자를 계산하는 pseudo code이다. 개인정보 수집일자와 약관의 종류를 가지고 개인정보 만료일자를 계산한다. 우선, 개인정보 수집일자의 년도와 달 정보를 year, month에 복사한다. 약관의 종류에 따라 year, month 값을 증감시킨다. A는 6개월로, month의 값에 따라 year를 증가시킬지 말지가 달라진다. B, C, D는 각각 1년, 2년, 3년이므로 year 값만 증가시킨다.

## B. QPOP 명령어

```

명령어로 QPOP을 입력 받음
if (MemberQueue.empty()==true)
    error code 300 출력

```

```

while (MemberQueue.empty()!=false)
    popNode = MemberQueue.pop()
    TermsList.insert(popNode)
    NameBST.insert(popNode)

```

```

function TermsList::insert(MemberQueueNode* popNode)
    MemberQueue에서 pop한 노드를 인자로 받아옴
    if (TermsList가 생성되지 않았다면)
        새로운 TermsList 노드 생성
        TermsList 노드의 회원 수 증가
        새로운 TermsBST 생성

    새로운 TermsList 노드를 추가해줄 위치 찾기
    if (popNode의 약관의 TermsListNode가 생성되어 있다면)
        TermsList 노드의 회원 수 증가
        TermsBST 노드 추가
    else
        새로운 TermsList 노드 생성

```



TermsList 노드의 회원 수 증가  
새로운 TermsBST 생성

QPOP 명령어를 입력 받았을 때, TermsList, TermsBST, NameBST를 생성하는 pseudo code이다. 위는 QPOP 명령어를 입력 받았을 때 프로그램의 동작이고, 아래는 TermsList와 TermsBST를 생성하는 프로그램의 동작이다.

QPOP을 입력 받으면, MemberQueue가 비어있는지 아닌지를 확인한다. 비어있다면, 에러 코드를 출력해준다. 비어있지 않다면, MemberQueue가 Empty가 아닌 동안 MemberQueue에서 노드를 pop해주고 pop한 노드를 이용해 TermsList, TermsBST, NameBST를 생성해준다.

이 때, TermsBST는 TermsList를 생성할 때 함께 생성된다. MemberQueue에서 pop한 노드를 이용하는데, 만약 TermsList가 아예 생성되지 않았다면 새로운 TermsList 노드를 생성해 TermsList에 추가해준다. 새롭게 TermsBST를 생성하고, TermsList 노드의 TermsBST 포인터가 생성한 TermsBST를 가리키도록 한다. TermsList가 생성이 되어 있는 경우, pop한 노드의 약관 값을 가진 TermsList 노드가 존재하냐 아니냐로 경우를 나눌 수 있다. pop한 노드의 약관의 노드가 존재한다면 해당 TermsList의 회원 수를 증가해주고 TermsBST에 노드를 추가해준다. 존재하지 않는다면 TermsList와 TermsBST를 새롭게 생성해준다.

### C. DELETE 명령어

명령어로 DELETE를 입력 받음

인자를 입력 받음

if (인자 == "NAME")

    삭제할 노드의 이름을 입력 받음

    NameBST에서 해당 이름을 가진 노드를 삭제

    TermsBST에서 해당 이름을 가진 노드를 삭제

    TermsList에서 삭제할 노드의 약관의 회원 수 감소

if (인자 == "DATE")

    삭제할 노드의 날짜를 입력 받음

    TermsBST에서 해당 날짜 이전의 만료 일자를 가진 노드를 삭제

    TermsList에서 삭제할 노드의 약관의 회원 수 감소

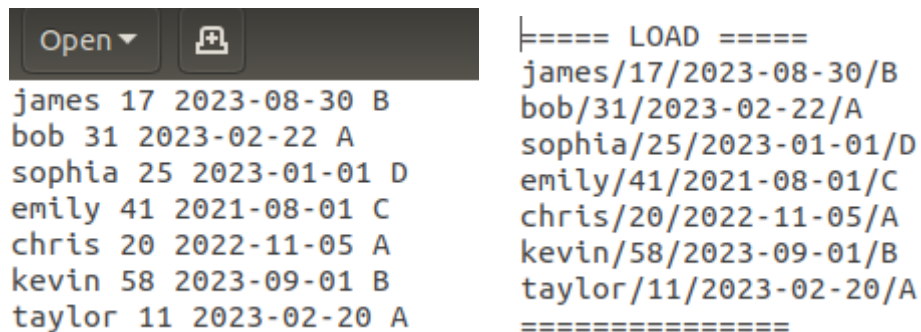
    NameBST에서 삭제할 노드의 이름을 가진 노드를 삭제

DELETE 명령어를 입력 받았을 때, 노드를 TermsList, TermsBST, NameBST에서 삭제

해주는 pseudo code이다. 인자가 "NAME"인 경우, 삭제할 노드의 이름을 입력 받아 해당 이름을 가진 노드를 NameBST에서 삭제해준다. 각 노드들에서 이름 정보는 중복되지 않기 때문에, TermsBST에서도 해당 이름을 가진 노드를 삭제해준다. 삭제할 노드의 약관 정보를 이용해 TermsList에서 해당 약관 노드의 회원 수를 감소해준다. 인자가 "DATE"인 경우, 삭제할 날짜를 입력 받아 TermsBST에서 해당 날짜보다 이전의 개인정보 만료일자를 가지는 노드들을 삭제해준다. TermsBST에서 삭제할 노드의 이름 정보를 이용해, 인자가 "NAME"인 경우에 TermsBST에서 노드를 삭제한 방법으로 노드를 삭제해준다. 삭제할 노드의 약관 정보를 이용해 TermsList에서 해당 약관 노드의 회원 수를 감소해준다. 이름 정보를 이용해 NameBST에서도 노드를 삭제해준다.

#### 4. Result Screen

##### A. LOAD



```
===== 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
=====
```

LOAD 명령어 수행 결과이다. 왼쪽 사진은 회원 정보가 들어있는 data.txt 파일이고, 오른쪽 사진은 LOAD 명령어가 성공적으로 수행되었을 때 Member\_Queue에 저장한 회원 정보를 log.txt에 출력한 결과화면이다.

##### B. ADD

```
===== ADD =====
tom/50/2020-07-21/D
=====

===== ADD =====
bella/94/2023-08-31/B
=====

===== ADD =====
harry/77/2024-02-03/B
=====
```

ADD 명령어 수행 결과이다. ADD 명령어로 기존의 Member\_Queue에 3개의 데이터를 추가하였다. 위 사진은 ADD 명령어가 성공적으로 수행되었을 때 Member\_Queue에 새롭게 저장한 3명의 회원 정보를 log.txt에 출력한 결과 화면이다.

#### C. QPOP

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

QPOP 명령어 수행 결과이다. Member\_Queue에 있던 데이터를 pop하여 Terms\_List, Terms\_BST, Name\_BST를 구성하였다. 위 사진은 QPOP 명령어가 성공적으로 수행되었을 때 log.txt에 성공 메시지를 출력한 결과 화면이다.

#### D. SEARCH

```
===== SEARCH =====  
bob/31/2023-02-22/2023-08-22  
=====
```

```
===== SEARCH =====  
tom/50/2020-07-21/2023-07-21  
=====
```

SEARCH 명령어 수행 결과이다. 찾고자 하는 이름을 입력 받아, Name\_BST에서 해당 이름을 찾아 회원 정보를 출력해준다. 이름이 bob, tom인 회원을 찾으려 하였다. 위 사진은 SEARCH 명령어가 성공적으로 수행되었을 때, 찾은 2명의 회원 정보를 출력한 결과 화면이다.

#### E. PRINT

```
===== 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  
chris/20/2022-11-05/2023-05-05  
taylor/11/2023-02-20/2023-08-20  
bob/31/2023-02-22/2023-08-22  
=====
```

```
===== PRINT =====  
Terms_BST B  
james/17/2023-08-30/2024-08-30  
bella/94/2023-08-31/2024-08-31  
kevin/58/2023-09-01/2024-09-01  
harry/77/2024-02-03/2025-02-03  
=====
```

PRINT 명령어 수행 결과이다. 인자로 NAME이 입력된 경우, Name\_BST에 있는 데

이터를 중위 순회 방식으로 출력한다. 인자로 약관(A, B, C, D)이 입력된 경우, Terms\_BST에 있는 데이터를 중위 순회 방식으로 출력한다. 왼쪽 사진은 인자로 NAME이 입력된 경우로 Name\_BST의 데이터를 성공적으로 출력한 결과 화면이다. 오른쪽 사진은 인자로 약관이 입력된 경우로 약관이 A, B인 Terms\_BST의 데이터를 성공적으로 출력한 결과 화면이다.

#### F. DELETE

```

===== 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/41/2021-08-01/2023-08-01
=====

===== 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/41/2024-02-03/2025-02-03
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 =====
500
=====

```

DELETE 명령어 수행 결과이다. 인자로 NAME과 회원 이름을 입력 받은 경우, Name\_BST에서 해당 회원의 노드를 삭제하고, Terms\_List와 Terms\_BST에서도 정보를 삭제한다. 인자로 DATE와 날짜를 입력 받은 경우, Terms\_BST에서 해당 날짜 이전의 개인정보 만료일자를 갖는 노드들을 삭제해준다. Terms\_List와 Name\_BST에서도 정보를 삭제한다.

사진은 DELETE NAME emily의 수행 결과로, 왼쪽 사진은 DELETE 명령어 수행 전에 Name\_BST와 약관이 C인 Terms\_BST의 정보를 출력해준 화면이다. 오른쪽 사진은 DELETE를 성공적으로 수행하여 성공 메시지를 출력하고, Name\_BST와 약관이 C인 Terms\_BST의 정보를 다시 출력해준 화면이다. 왼쪽 사진과 오른쪽 사진을 비교하면 BST에서 emily 정보가 사라졌음을 확인할 수 있다.

#### G. EXIT

```

===== EXIT =====
Success
=====

```

EXIT 명령어 수행 결과이다. EXIT 명령어 수행 시, 성공 메시지를 출력하고 프로그램이 종료된다.

## 5. Consideration

txt 파일을 읽어올 때, core dumped 오류가 발생하였다. 이는 파일의 끝에 도달할 때까지 txt 파일의 정보를 한 줄씩 읽어오는 데서 오류가 발생한 것이었다. 기존 코드는 파일의 끝에 도달할 때까지 반복하는 while 문에 txt 파일을 한 줄 읽어오고 코드를 수행하는 방법으로 구현을 하였다. 그러나, 주어진 txt 파일의 제일 끝이 개행이 되어있기 때문에 txt 파일로부터 우선 한 줄을 읽어오고 파일의 끝에 도달할 때까지 반복하는 while문에 코드를 수행하고 한 줄씩 읽어오도록 코드를 수정하였다.

회원의 정보를 어떤 타입을 이용해 받을 것인가에 관한 고민을 하였다. char 배열과 string 두 가지의 경우를 생각해보았는데, 입력 받을 회원 정보들의 format(이름은 20자 이하 등)이 정해져 있기 때문에 char 배열을 이용하였다. char 배열을 이용함으로써, 공백을 기준으로 토큰화 해주는 strtok 함수로 회원 정보 노드를 손쉽게 만들 수 있었다. 또한, 개인정보 수집일자와 약관(A/B/C/D)을 이용해 개인정보 만료 일자를 계산하는 것을 아스키코드를 이용하여 손쉽게 구현할 수 있었다.

ADD의 예외처리(입력 인자가 부족한 경우)를 구현할 때 문제가 발생하였다. 처음에는 입력 받은 인자의 공백 개수를 세서 공백 개수가 3개가 아닌 경우를 입력 인자가 4개가 아닌 경우로 생각하여 에러 코드를 출력하도록 해주었다. 그런데 "tom 50 2020-07-21 "과 같이 약관 정보가 입력되지 않았지만 공백 개수가 3개인 경우 문제가 발생하게 된다. 따라서, strtok 함수를 공백 혹은 개행 문자를 구분자로 하여 하나라도 nullptr인 경우 에러 코드를 출력하도록 해주었다. 예외처리를 이러한 방식으로 구현하였을 때 위의 문제를 해결할 수 있었다.

처음 메모리 누수를 확인했을 때, 약 1000bytes 정도의 메모리 누수가 발생하였다. 작성한 코드를 확인해보니 BST의 노드들을 삭제할 때, 동적할당 해준 노드들을 삭제해주지 않아 메모리 누수가 발생하였다. 따라서, delete를 이용해 사용하지 않는 노드들을 삭제해주었다.