데이터 구조 설계

1차 프로젝트

제출일: 2023-10-11(수)

학과: 컴퓨터정보공학부

학번: 2020202034

이름: 김태완

1. Introduction

A. 프로젝트 설명: 이 프로젝트는 queue, linked list, binary search tree를 이용해 사람들의 개인정보를 정렬해 보관하는 프로그램을 만드는 것입니다. 사람들의 개인정보는 data.txt에 작성되어 있고, 프로그램에서 실행할 명령어들은 command.txt에 작성되어 있습니다. command.txt에서 명령들을 순차적으로 읽어와 수행하고 결과를 log.txt에 작성하는 것이 기본 구조이지만, 전체적인 동작의 흐름은 MemberQueue에 data.txt에 있는 개인정보를 모두 push하고(ADD 명령어를 통해 추가로 push할 수도 있음), 이를 하나씩 pop하며 개인 정보가 이름 순으로 정렬되는 NameBST, 개인정보 만료일자순으로 정렬되는 TermsBST, 이를 하나씩 가지며 약관 정보에 따라 분류되는 노드를 가진 TermsList를 구성합니다. 그리고 Search, Print, Delete등의 연산을 수행하고, 실패하면 에러 코드를 출력하는 것입니다.

B. 명령어별 설명

i. LOAD: data.txt에 있는 개인정보를 불러와 MemberQueueNode로 만들어 MemberQueue에 삽입합니다. 큐가 차 있거나, 개인정보를 불러오는 데 실패하면 에러 코드를 반환합니다.

ii. ADD: 개인정보를 추가로 MemberQueue에 삽입하는 명령어입니다. 인자가 부족하면 에러 코드를 반환합니다.

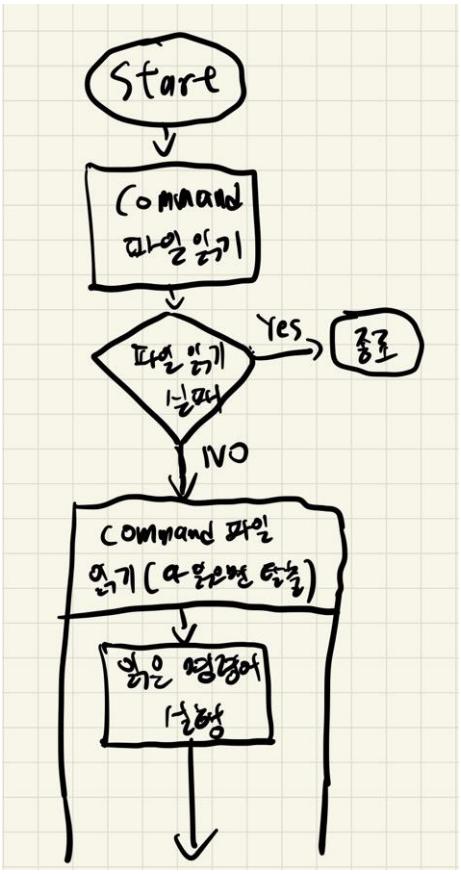
iii. QPOP: 큐에 저장된 개인정보를 차례대로 POP하며 NameBST, TermsList(TermsBST)에 삽입합니다. 큐가 비어있으면 에러 코드를 반환합니다.

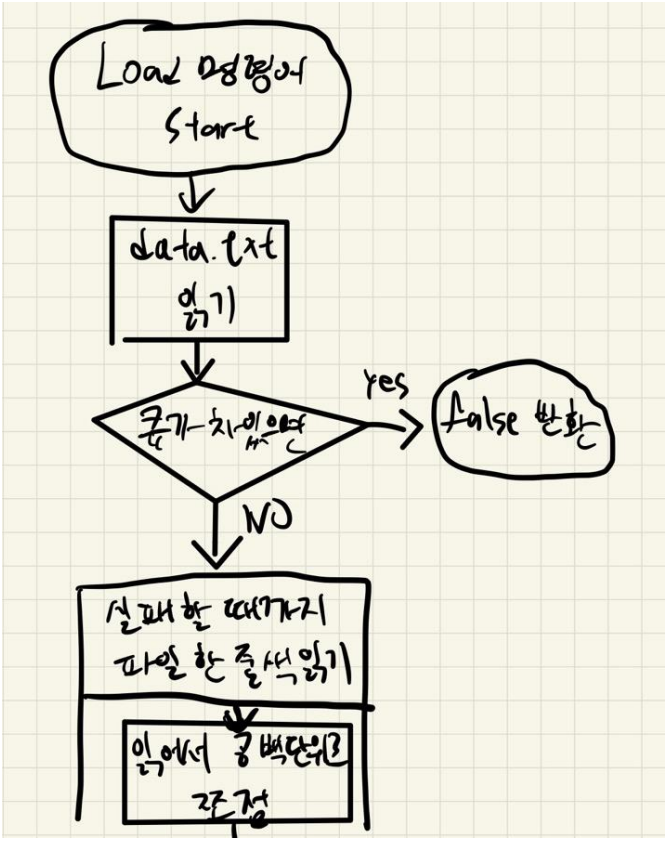
iv. SEARCH & PRINT: SEARCH는 NameBST에서 인자로 받은 name을 가지는 개인정보를 찾아 반환하고, PRINT는 name 기준으로 또는 약관별로 날짜 기준으로 출력합니다. 실패하면 에러 코드 반환

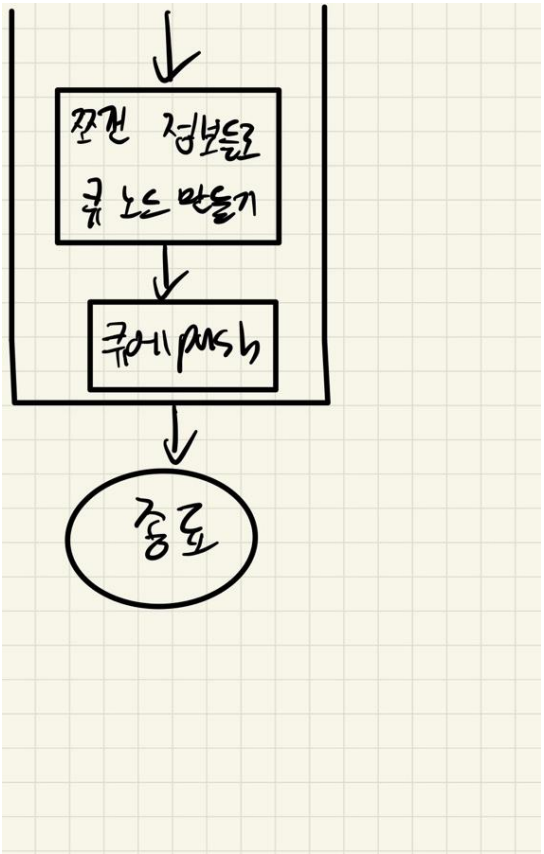
v. DELETE: 해당 정보를 가지는 노드를 모든 자료구조에서 삭제

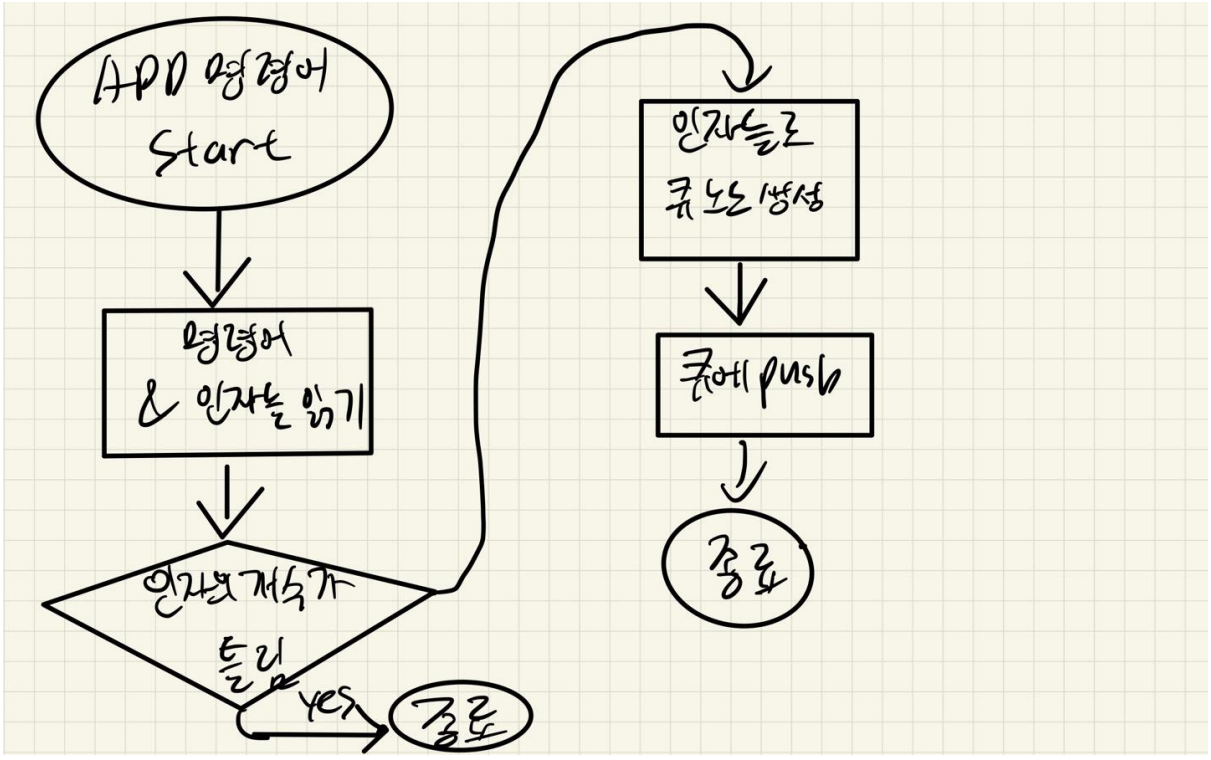
vi. EXIT: 프로그램 종료

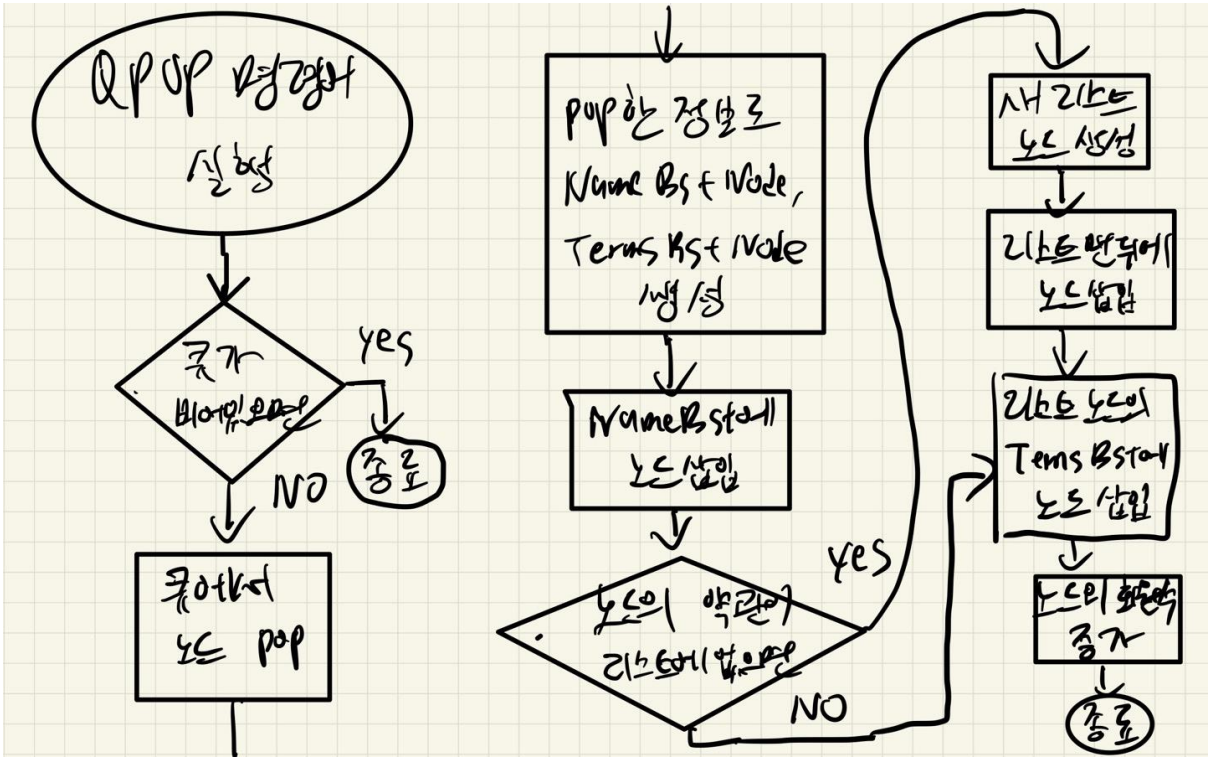
2. Flowchart

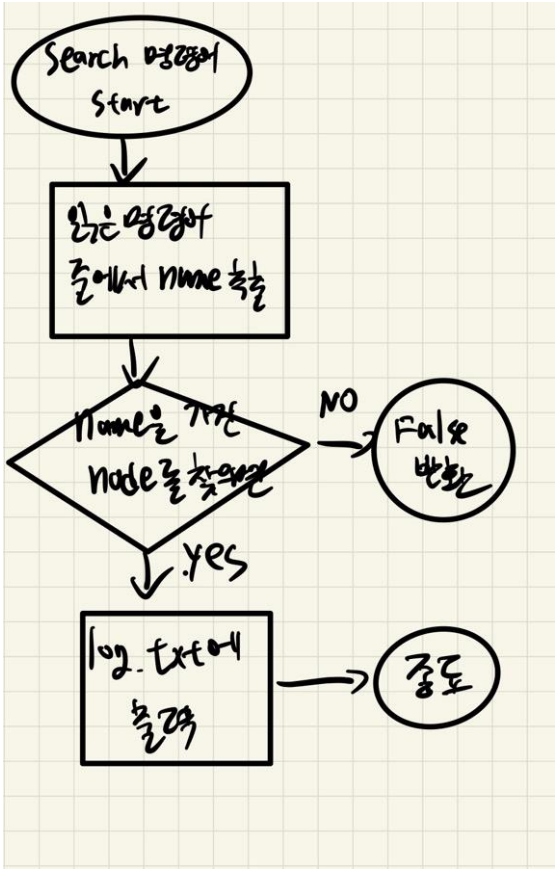

Start
Command 파일 읽기
파일 읽기 실패
Yes
종료
NO
Command 파일 읽기 (a 문장씩 읽음)
읽은 명령어 실행
EXIT 명령어 입력됨
Yes
종료
NO
명령어 실행 결과(성공/실패) log.txt에 작성

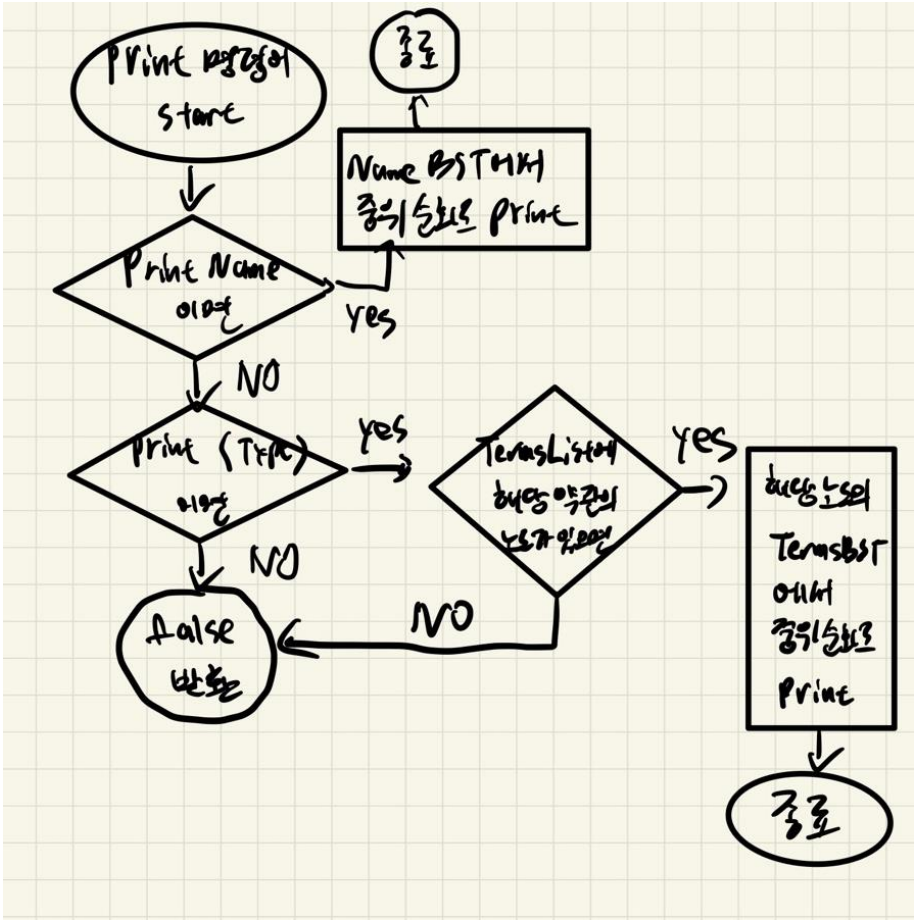

Load 명령어
Start
data.txt
읽기
줄개수 입력됐
yes
False 반환
NO
실패할 때까지
파일 한 줄씩 읽기
읽어서 공백단위로
구분

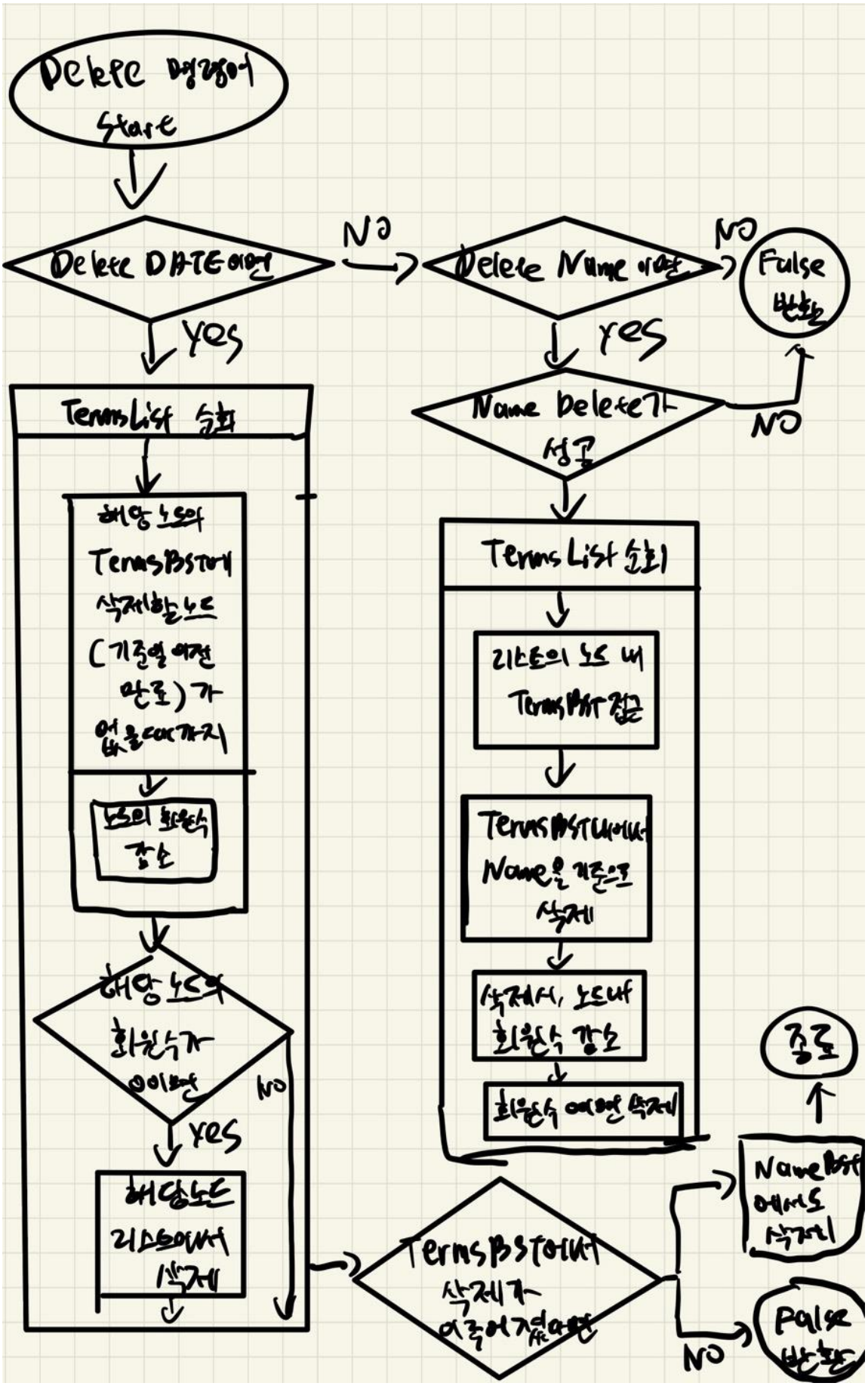

좌표 정보들로
큐 노드 만들기
큐에 push
종료


APD 명령어
Start
명령어
& 인자들 읽기
인자의 개수가
틀림
yes
종료
인자들로
큐 노드 생성
큐에 push
종료


QPOP 명령어
실행
큐가
비어있으면
yes
종료
NO
큐에서
노드 pop
pop한 정보로
NameBST Node,
TermsBST Node
생성
NameBST에
노드삽입
노드의 약칭이
리스트에 있으면
yes
NO
새 리스트
노드 생성
리스트 맨뒤에
노드삽입
리스트 노드의
TermsBST에
노드 삽입
노드의 frequency
증가
종료


Search 명령어 start
입력 명령어 줄에서 name 추출
Name을 가진 node를 찾았는가
NO
False 반환
yes
log.txt에 출력
종료


Print 명령어 start
종료
Name BST에서 중위순회로 print
Print Name 이었는가
yes
NO
Print (Type) 이었는가
yes
TermsList에 해당 약관의 분류가 있었는가
yes
해당 노드의 TermsBST에서 중위순회로 print
종료
NO
NO
false 반환


Delete 명령어 Start
Delete DATE이면
NO
Delete Name이면
NO
False 반환
YES
YES
TermsList 순회
Name Delete가 성공
NO
해당노드의 TermsBST에서 삭제할노드 (기준에 맞는) 가 없을때까지
TermsList 순회
노드의 회원수 감소
리스트의 노드 내 TermsBST 접근
해당노드의 회원수가 0이면
NO
YES
TermsBST에서 Name을 기준으로 삭제
삭제시, 노드내 회원수 감소
종료
회원수 0이면 삭제
해당노드 리스트에서 삭제
TermsBST에서 삭제가 이루어졌으면
NameBST에서도 삭제
NO
False 반환

3. Algorithm

A. QPOP Command

i. Queue에서 node 하나 pop

ii. Pop한 node의 정보로 NameBSTNode, TermsBSTNode 생성

iii. NameBST에 생성한 NameBSTNode 삽입

iv. TermsBSTNode의 가입약관 정보와 일치하는 TermsListNode가 있는지를 TermsList를 순회하며 검사

v. 있으면 해당 노드의 회원 수를 올리고, 해당 노드의 TermsBST에 TermsBSTNode 삽입

vi. 없으면 해당 노드의 약관 정보를 가지는 TermsListNode 하나를 생성 후 TermsList에 추가 후, 생성한 TermsListNode의 TermsBST에 노드를 삽입합니다.

B. SEARCH, PRINT Command

i. BST의 구조: 루트 노드를 기준으로 왼쪽 노드는 특정 값(예: 키 값)이 작은 데이터를 가지고, 오른쪽 노드는 큰 데이터를 가집니다. 그리고 각각의 child node에 대해서도 이 BST의 구조가 유지됩니다(재귀적).

ii. BST Search: NameBST는 이름을 기준으로 정렬되어 있는데, 왼쪽 노드는 이름이 알파벳 순으로 앞쪽이고, 오른쪽 노드는 이름이 같거나 알파벳 순으로 뒤쪽인 구조입니다. 따라서 탐색의 과정은 다음과 같습니다.

1. 루트 노드부터 탐색을 시작합니다.
2. 탐색 중인 노드의 Name 정보를 search 대상인 name과 비교해 작으면 왼쪽 노드로 이동하고, 크면 오른쪽 노드로 이동합니다. 만약 같으면 반복문을 빠져나옵니다.
3. 탐색 중인 노드가 null pointer가 될 때까지 반복하고, 탐색에 성공하였다면 성공한 node의 포인터 정보를, 아니면 null pointer를 반환합니다.

iii. BST Print(In-Order)

1. BST에서 노드의 정보를 정렬된 순으로 출력하려면 중위 순회를 해야합니다. 중위 순회란, 트리에서 루트 노드를 기준으로 왼쪽 노드, 루트 노드, 오른쪽 노드 순으로 방문하는 것을 말합니다. 트리는 각각의 노드를 루트 노드로 가지는 subtree를 가진다고 가정할 수 있기 때문에, 이를 재귀함수로 구현할 수 있습니다. 이때 루트 노드부터 중위 순회를 하면, 트리의 가

장 왼쪽 노드부터 중간 부분을 거쳐 오른쪽 노드로 옮겨 가며 방문하게 됩니다.

2. 이때 방문하여 수행할 수 있는 연산은 다양한데, 여기서는 단순히 node의 내용을 출력하는 print를 수행하는 것입니다.

C. DELETE Command

i. NameBST에서 Name 정보를 기준으로 삭제

1. NameBST는 Name을 기준으로 정렬이 되어있기 때문에 일반적인 BST에서 노드를 delete하는 방법을 사용하면 됩니다.

2. 우선, 위에서 설명한 Search 알고리즘으로 지워야 할 노드(이하 p)를 찾습니다. 이때, 삭제를 위해 추가로 해당 노드의 부모 노드를 가리키는 포인터(이하 pp)를 하나 선언해 search 과정에서 루트부터 찾아 내려갈 때 계속해서 탐색 중인 노드의 부모 노드를 가리키게 합니다.

3. 노드를 찾지 못하면 끝내고, 찾는다면 경우를 4가지로 나눌 수 있습니다.

A. 삭제할 노드가 leaf node(자식이 없는 노드)인 경우: p가 pp의 왼쪽 자식이면 pp의 왼쪽 포인터를, 오른쪽 자식이면 오른쪽 포인터를 널로 초기화시키고 p를 delete합니다.

B. 삭제할 노드가 왼쪽 자식만 가진 경우: p가 pp의 왼쪽 자식이면 pp가 왼쪽 자식으로 p의 왼쪽 자식을 가지게 하고, 오른쪽 자식이면 오른쪽 자식으로 p의 왼쪽 자식을 가지게 합니다. 그리고 p를 delete합니다.

C. 삭제할 노드가 오른쪽 자식만 가진 경우: p가 pp의 왼쪽 자식이면 pp가 왼쪽 자식으로 p의 오른쪽 자식을 가지게 하고, 오른쪽 자식이면 오른쪽 자식으로 p의 오른쪽 자식을 가지게 합니다. 그리고 p를 delete합니다.

D. 삭제할 노드가 양쪽 자식을 모두 가진 경우: 이 경우에는 탐색을 한 번 더 해야 합니다. Ppv, pv, cur 세 포인터 변수를 두어 ppv가 p를, pv가 p의 오른쪽 자식, cur이 p의 오른쪽 자식의 왼쪽 자식을 가리키게 초기화한 후, ppv가 pv를, pv가 cur을 따라가게 하고 cur이 null이 될 때까지 왼쪽 자식으로 이동하도록 동작을 반복합니다. 이렇게 되면 pv가 p의 오른쪽 subtree에서 가장 작은, 즉 가장 왼쪽 노드를 가리키게 됩니다. 만약 탐색이 끝난 후에 ppv == p라면 cur이 시작부터 null이었다는 이야기고, 이는 p의 오른쪽 subtree에서 pv(subtree의

root node)가 가장 작은 이름을 가진다는 의미(pv는 왼쪽 자식을 가지지 않음)입니다. 따라서 pv의 정보를 p로 옮기고 ppv가 오른쪽 자식으로 pv의 오른쪽 자식을 가지게 하고 pv를 delete하면 됩니다. 그 외의 경우에는 ppv의 왼쪽 자식이 pv이기 때문에 pv의 정보를 p로 옮기고 ppv가 왼쪽 자식으로 pv의 오른쪽 자식을 가지게 하고 pv를 delete하면 됩니다.

ii. TermsBST에서 개인정보 만료일자를 기준으로 삭제

1. 이 경우에서는 한 개인정보 만료일자를 기준으로 작은 노드를 삭제하기 때문에 여러 노드를 삭제하는 경우를 생각해야 합니다. 그러나 저는 한 번에 하나씩 노드를 삭제하고, 이를 DELETE 명령어 함수를 구현할 때 삭제할 노드가 없을 때까지 반복시켜 여러 노드를 한 번에 지우도록 구현하였습니다.

2. 우선 인자로 받은 만료일자 이전의 노드를 지우는 방법은 NameBST에서 name을 기준으로 지울 때와 거의 유사합니다. 다만, 삭제할 노드를 찾을 때 계속해서 왼쪽 노드로만 간다는 점이 차이점입니다.

3. TermsBST에서 delete 함수가 한 번 호출되면 기준이 되는 만료일자보다 적은 날짜의 노드를 하나 지우고 끝나는데, 이를 앞서 언급한 것처럼 DELETE 명령어 함수 내에서 삭제할 노드가 없을 때까지 반복해 모두 삭제합니다.

iii. TermsBST에서 Name 정보를 기준으로 삭제

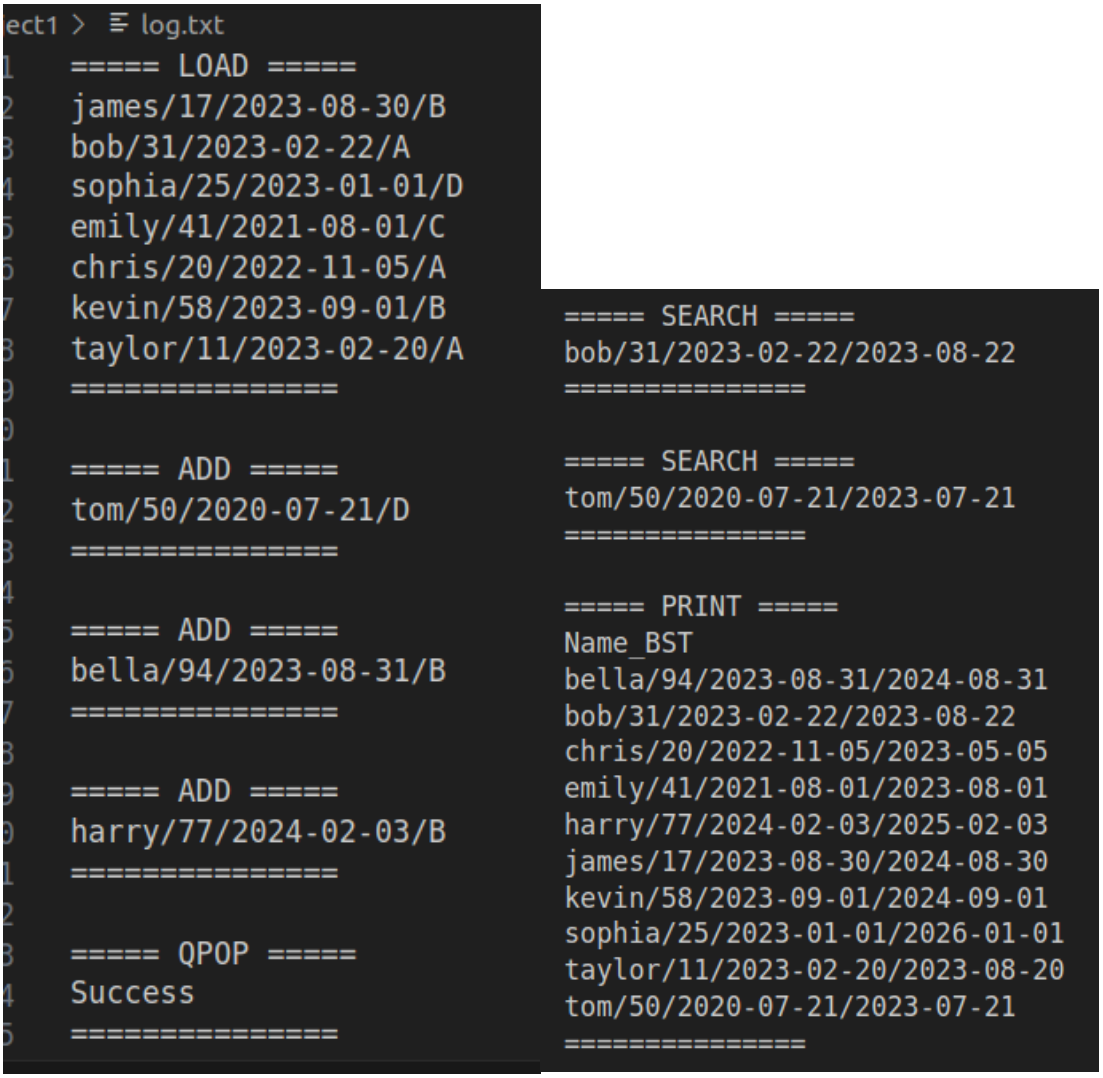
1. TermsBST는 Name을 기준으로 정렬되어 있지 않기 때문에 후위 순회를 통해 노드를 방문하여 기준 name과 같은 name을 가지면 삭제하는 방식으로 구현하였습니다.

2. TermsListNode에서 TermsBST 내의 노드 수(회원 수)를 가지고 있기 때문에, 삭제 시에 이를 업데이트 시키기 위해 Delete 함수의 인자로 int &cnt를 넣어 삭제할 때마다 삭제한 횟수를 증가시켜주도록 했습니다.

iv. NameBST에서 개인정보 만료일자를 기준으로 삭제

1. 위의 방식과 똑같이 트리를 후위순회하며 노드를 방문하고, 노드의 개인정보 만료일자가 기준 이전이면 삭제하는 방식으로 구현하였습니다.

4. Result Screen



```
ect1 > ≡ log.txt
1    ===== LOAD =====
2    james/17/2023-08-30/B
3    bob/31/2023-02-22/A
4    sophia/25/2023-01-01/D
5    emily/41/2021-08-01/C
6    chris/20/2022-11-05/A
7    kevin/58/2023-09-01/B
8    taylor/11/2023-02-20/A
9    ===============
0
1    ===== ADD =====
2    tom/50/2020-07-21/D
3    ===============
4
5    ===== ADD =====
6    bella/94/2023-08-31/B
7    ===============
8
9    ===== ADD =====
0    harry/77/2024-02-03/B
1    ===============
2
3    ===== QPOP =====
4    Success
5    ===============
```

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

LOAD 명령어는 파일에서 data를 읽어 노드로 만들어 큐에 push하는 명령어인데, 결과 화면을 보면 성공적으로 push되었음을 알 수 있습니다.

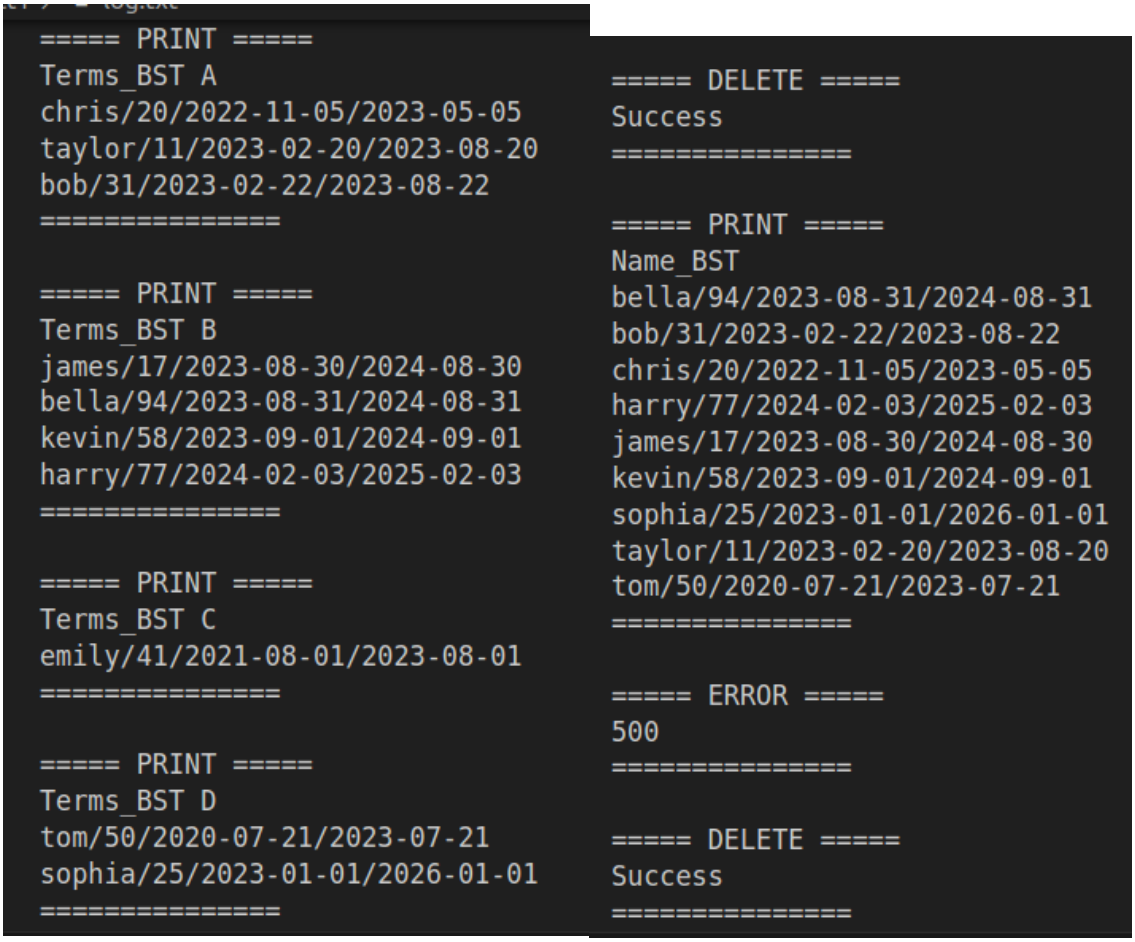
ADD 명령어는 인자로 받은 데이터로 노드를 만들어 큐에 push하는 명령어인데, 결과 화면만 보면 성공적으로 큐에 push되었는지는 모르지만, 뒤에 PRINT 명령어를 실행한 결과를 보면 tom, bella, harry가 존재하므로 잘 push되었음을 할 수 있습니다.

QPOP 명령어는 큐에 저장된 개인정보들을 NameBST, TermsList(TermsBST)로 보내주는 명령어인데, 성공했음을 알 수 있습니다.

SEARCH 명령어는 인자로 받은 이름의 개인정보를 찾아 출력해주는데, commant.txt를 보면 SEARCH bob, SEARCH tom을 입력했으므로 bob과 tom의 개인정보가 잘 출력되었음을 알 수 있습니다.

PRINT 명령어는 NAME을 인자로 받으면 이름 순으로, A~D의 약관 종류를 입력받으면 각 약

관에 해당하는 개인정보를 만료일자 순으로 출력해 주는 명령어인데, 우선 PRINT NAME은 이름 순으로 잘 출력되었음을 알 수 있습니다.



그리고 PRINT A~D를 했을 때는 개인정보만료인자 순으로 잘 출력되었음을 알 수 있습니다.

DELETE 명령어는 인자로 이름을 받으면 해당 이름을 가진 노드를, 날짜를 받으면 개인정보 만료일이 그 날짜 전인 노드를 NameBST와 TermsList내의 TermsBST에서 지우는 명령어인데, 일단 첫 번째 DELETE는 DELETE NAME emily였는데, 바로 다음 결과에 Name을 기준으로 PRINT했을 때 emily가 없음을 확인할 수 있고, 그 다음 명령어는 PRINT C였는데 약관이 C인 사람이 emily밖에 없었으므로 약관이 C인 TermsListNode가 존재하지 않아 에러 코드가 출력되었음을 볼 수 있습니다. 그리고 두 번째 DELETE는 DELETE 2024-09-01 이었는데, 우선 정상적으로 지워졌다는 결과를 볼 수 있습니다.

```
===== 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 =====
500
===============

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

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

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

첫 번째 PRINT NAME의 결과를 확인하면 개인정보 만료일자가 2024-09-01 이전인 노드들이 NameBST에서 모두 사라진 것을 알 수 있고, 그 다음 에러 코드가 나온 PRINT는 PRINT A였는데, 기존의 A 약관을 가졌던 사람들인 chris, taylor, bob의 개인정보 만료일자가 모두 2024-09-01 이전이기 때문에 A 약관을 가지는 사람이 사라져 A 약관인 사람의 정보를 가지는 TermsListNode가 삭제되어 에러 코드가 나온 것입니다. B 약관을 가지는 사람과 D 약관을 가지는 사람들 중에는 삭제되지 않은 개인정보가 있어 남은 사람들의 정보가 출력되는 것을 볼 수 있습니다.

EXIT 명령어는 프로그램을 종료하는 명령어로, 이 명령어가 실행된 후에 프로그램이 종료된 것을 볼 수 있습니다.

5. Consideration

A. 프로젝트를 진행하면서 수업시간에 배웠던 BST에 대해 더 자세히 공부한 것 같습니다. 특히 BST에서 자식을 2개 가지는 노드의 삭제를 강의 때는 완전히 이해하지 못했는데, 코드를 작성하고 돌려보며 오류를 수정하는 과정에서 흐름을 완전하게 이해했습니다. 또, 서로 다른 자료구조를 연결하는 과정에서 각 자료구조를 class로 구현할 때 method의 기능을 명확하고 간결하게 구성해야 후에 사용할 때 더 편하다는 것을 잘 알 수 있었습니다.