

데이터구조설계

DS_project1

컴퓨터정보공학부

2020202031

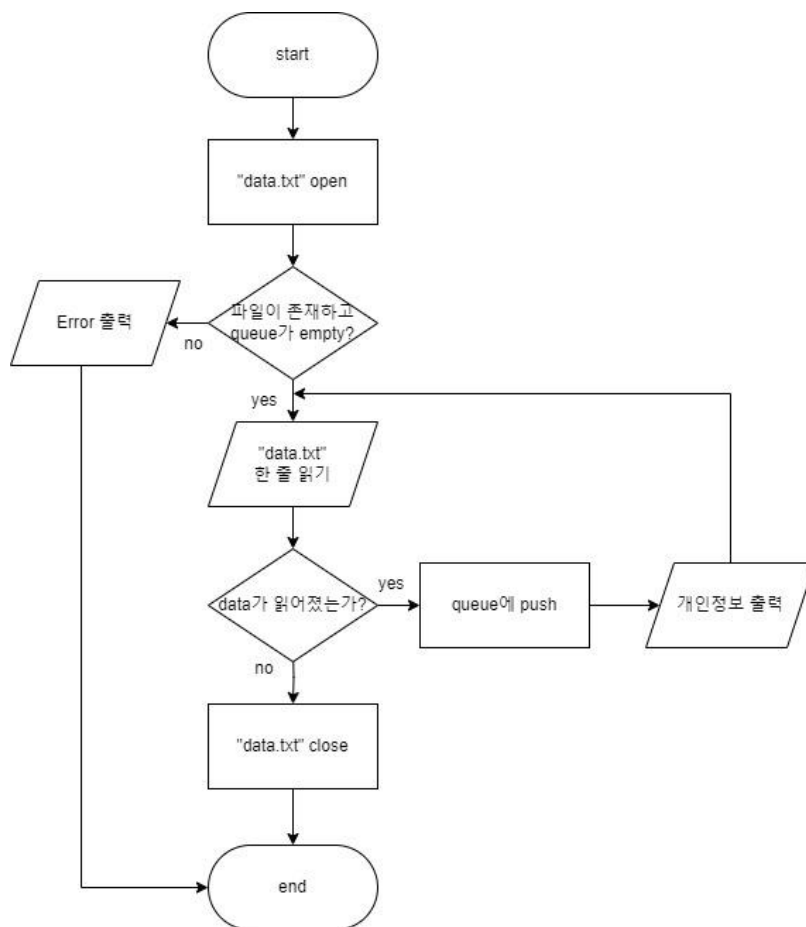
김재현

1. Introduction

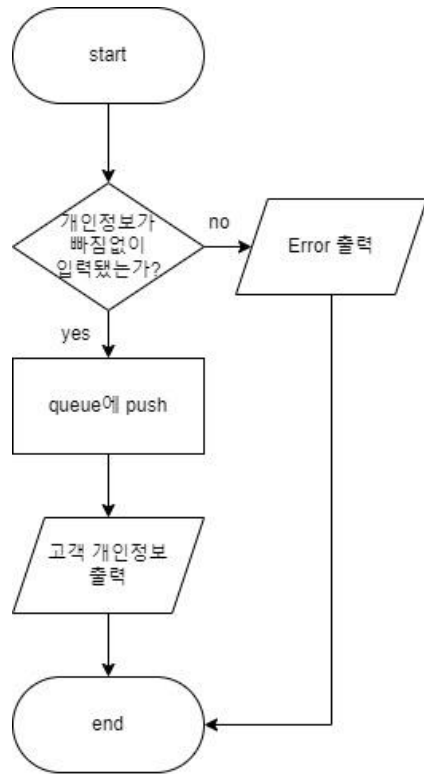
이번 프로젝트에서는 회원이름, 나이, 개인정보수집일자, 가입약관종류를 포함한 개인정보를 이진 탐색 트리, 연결 리스트, 큐 자료구조를 통해 관리하는 프로그램을 구현합니다. 개인정보가 저장돼 있는 텍스트파일로부터 데이터를 전부 읽어들이어서 큐 자료구조에 push해줍니다. 그 후 차례로 데이터들을 pop해서 가입약관종류 별로 termsList를 구성하고 가입약관종류가 같은 데이터끼리 개인정보만료일자 오름차순으로 termsBST를 구성합니다. 동시에 이름 오름차순으로 nameBST를 구성합니다. 이 두 종류의 이진 탐색 트리를 통해 회원정보를 출력할 때 중위순회를 통해 오름차순으로 출력할 수 있고, 데이터를 삭제하더라도 정렬을 유지할 수 있어, 데이터를 탐색하기에도 용이합니다.

2. Flowchart

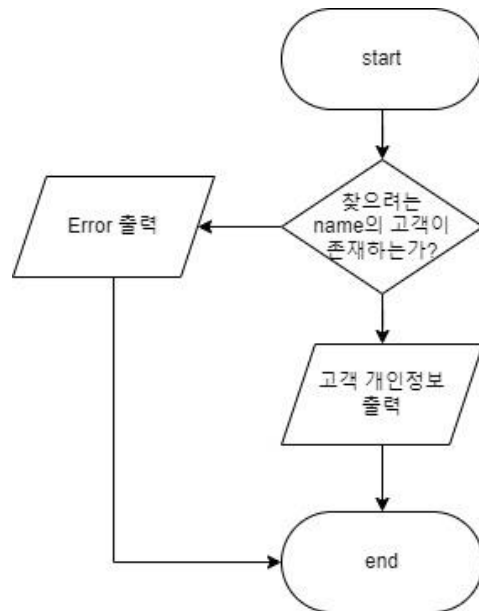
1) LOAD



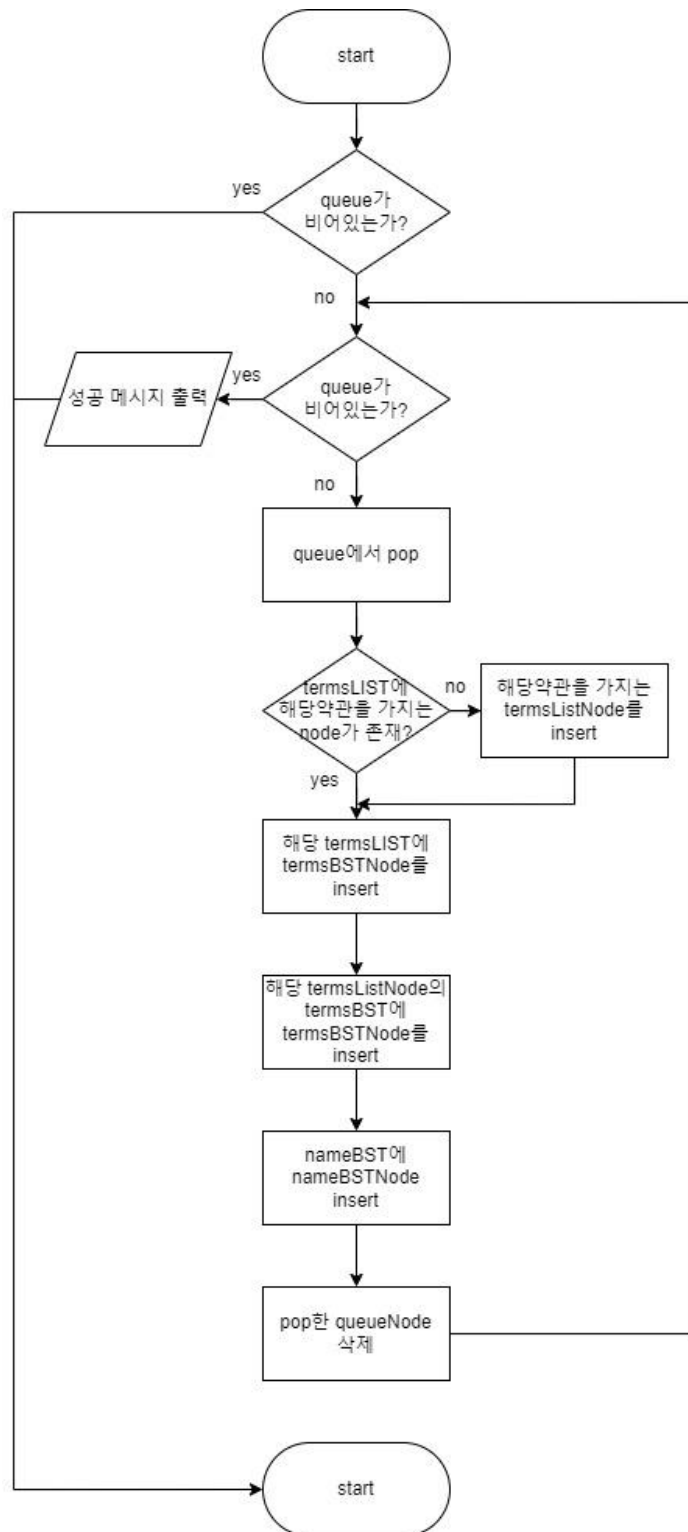
2) ADD



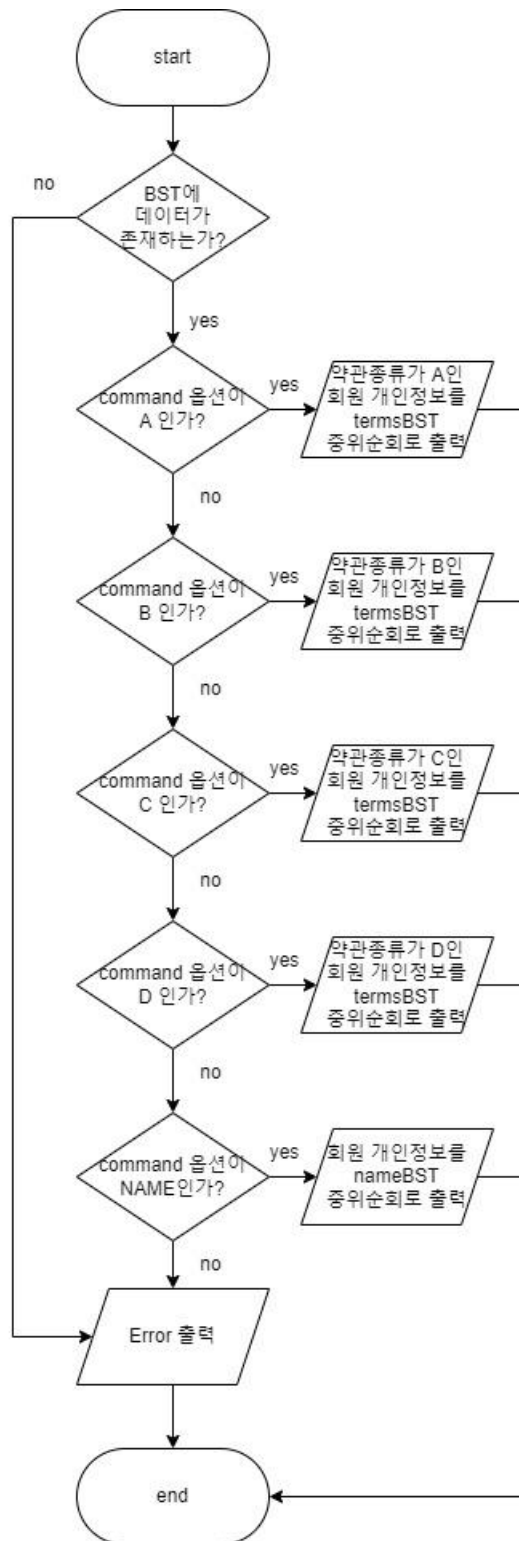
3) SEARCH



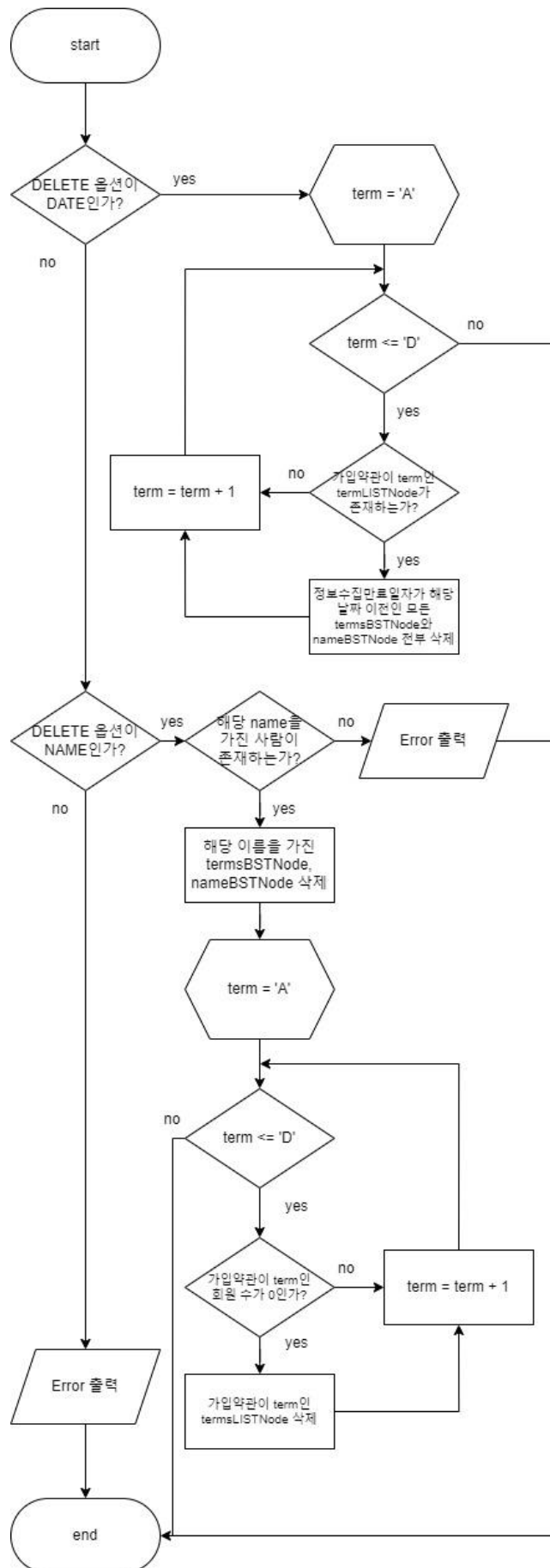
4) QPOP



5) PRINT



6) DELETE



3. Algorithm

1) nameBST의 printAll 메소드

고객의 개인정보를 이름 오름차순으로 출력하기 위해 중위순회를 사용했습니다. 재귀를 통해 모든 노드를 순회함으로써 모든 고객의 개인정보를 출력했고, 재귀 중에서도 중위순회를 통해 고객의 개인정보를 오름차순으로 출력했습니다.

2) nameBST의 insert 메소드

루트부터 시작하여 insert할 적절한 위치를 탐색합니다. insert하려는 고객의 이름이 탐색하는 노드의 고객이름보다 작으면 탐색노드의 오른쪽 노드로 이동하고, insert하려는 고객의 이름이 탐색하는 노드의 고객이름보다 크면 왼쪽 노드로 이동합니다. 이 과정을 탐색노드가 null이 될때까지 반복합니다. 이 때, null이된 탐색노드의 위치가 곧 insert할 적절한 위치가 됩니다. 따라서 부모노드와 insert하려는 노드를 연결시키면 이로서 insert가 완성됩니다.

3) nameBST의 delete 메소드

insert와 유사한 방법으로 일단 delete하고 싶은 node를 찾아냅니다. 그리고 찾아낸 node가 leaf node인지, right child만을 가지는지, left child만을 가지는지, left, right node를 다 가지는지를 판별합니다.

leaf node라면 delete하고자하는 node의 위치를 null로 set한 후, delete 해줍니다.

right child만을 가진다면, delete하고자하는 node가 root라면 root를 오른쪽 자식으로 옮겨주고 delete해줍니다. root가 아니라면, 부모 node가 왼쪽에 있는지, 오른쪽에 있는지를 판별합니다. 부모 node가 오른쪽에 있다면, delete하려는 node의 right child를 부모 node의 left child로 set해줍니다. 만약 부모 node가 왼쪽에 있다면, delete하려는 node의 left child를 부모 node의 right child로 set해줍니다.

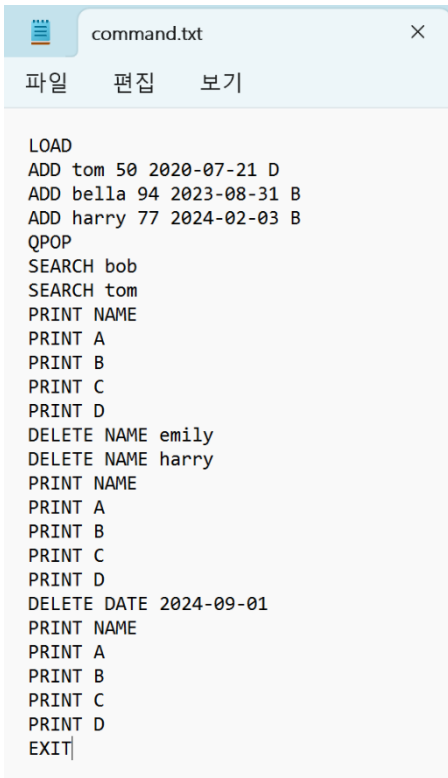
left child만을 가진다면, right child만을 가지는 경우와 반대로 delete과정을 진행해주면 됩니다.

만약 left, right child 모두를 가진다면, 오른쪽 subtree에서 가장 작은 노드를 delete하려는 node 위치에 대체해주고, delete를 진행합니다. 일단 오른쪽 subtree에서 가장 작은 노드를 탐색한 후, delete하려는 노드의 정보를 오른쪽 subtree에서 가장 작은 노드로 업데이트 해줍니다. 그리고 부모 자식 간의 관계를 재정의해준 후 오른쪽 subtree에서 가장 작은 노드의 delete를 진행합니다.

4) termsBST의 delete 메소드

termsBST의 delete 같은 경우에는 개인정보만료일자를 기준으로 작은 값들을 모두 삭제해야 하기에, queue를 사용하거나, 재귀를 사용해야 합니다. queue를 사용할 경우 termsBSTNode를 위한 queue를 새로 정의해 주어야 하기 때문에, 재귀를 사용하도록 했습니다. 후위순회를 사용했고 delete의 반환값을 child로 set해주었습니다. 그리고 입력된 날짜보다 개인정보만료일자가 작은 노드들을 삭제하고, 이 정보를 반환해줬습니다. 그렇기에 root가 null이 되는 경우는 없고, 어떻게든 root가 재정의되게 됩니다. 또한 termsBSTNode를 삭제하면서 nameBSTNode도 함께 삭제해줌으로써 DELETE명령어를 완벽히 수행했습니다.

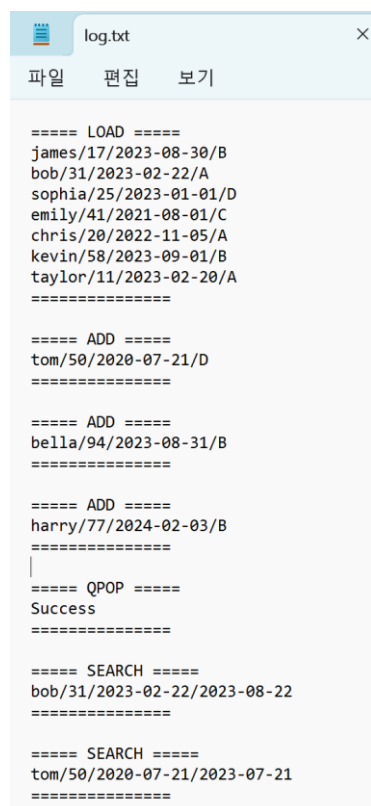
4. Result Screen



```
command.txt
파일 편집 보기

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
DELETE NAME harry
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
DELETE DATE 2024-09-01
PRINT NAME
PRINT A
PRINT B
PRINT C
PRINT D
EXIT
```


위의 명령어 순서대로 검증을 진행했다.



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

LOAD를 실행하니 data.txt 파일에서 개인정보를 읽어와 memberqueuenode를 생성하고 memberqueue에 push해 주는 동시에 고객의 개인정보를 출력해주는 것을 확인할 수 있다.

ADD를 통해 개인정보를 입력하면 LOAD와 동일하게 개인정보를 통해 memberqueuenode를 생성하고 memberqueue에 push해 주는 동시에 고객의 개인정보를 출력해주는 것을 확인할 수 있다.

결국 LOAD와 ADD는 고객의 개인정보를 한번에 memberqueue에 넣어주느냐 한 개만을 넣어주느냐의 차이라고 볼 수 있다.

QPOP 명령어는 memberqueue에 들어있는 memberqueuenode 전부를 pop해주고 그 정

보들로 nameBSTNode, TermsLISTNode, TermsBSTNode를 생성하고 데이터들을 정리해줍니다. 그 과정이 오류 없이 잘 진행됐다면 Success를 출력합니다.

SEARCH 명령어는 찾고자 하는 고객의 이름 name을 입력 받고, 그 이름을 가지고 있는 고객을 탐색합니다. 이때 nameBST가 오름차순으로 정렬돼있으므로 root부터 차례로 name의 크기를 비교해가며 고객을 탐색하고, name을 가지는 고객이 존재하면 그 고객의 개인정보를 출력하고, 존재하지 않는다면 오류메시지를 띄웁니다.



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

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

==== PRINT ====
Terms_BST C
emily/41/2021-08-01/2021-08-01
=====

==== PRINT ====
Terms_BST D
tom/50/2020-07-21/2020-07-21
sophia/25/2023-01-01/2023-01-01
=====
```

PRINT 명령어는 인자로 NAME을 받느냐 DATE를 받느냐 두 가지 경우로 나뉩니다.

PRINT NAME이면 nameBST에 이름 오름차순으로 저장된 모든 고객의 개인정보를 이름 오름차순으로 출력합니다. 모든 고객의 개인정보를 출력해야 하므로 재귀를 사용합니다. 그리고 이름 오름차순으로 정렬된 BST를 이름 오름차순으로 출력해야하기 때문에 재귀 중에서도 중위순회를 사용하여 정렬된 그대로 고객의 개인정보를 출력합니다.

PRINT DATE이면 DATE 뒤에 붙은 termType으로 어떤 termType을 가지는 고객의 개인정보를 출력할 것인지를 정합니다. termType을 가지는 모든 고객의 개인정보를 출력하는데, 이 때 개인정보수집만료일자를 기준으로 오름차순 출력합니다. termBST는 개인정보수집만료일자를 기준으로 오름차순으로 정렬돼있기 때문에 이 역시 PRINT NAME과 동일하게 재귀 중에서도 중위순회를 사용하여 탐색하면서 개인정보수집만료일자 기준 오름차순으로 고객의 개인정보를 출력합니다.

```
log.txt
파일 편집 보기
===== DELETE =====
Success
=====

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

===== PRINT =====
Terms_BST B
james/17/2023-08-30/2023-08-30
bella/94/2023-08-31/2023-08-31
kevin/58/2023-09-01/2023-09-01
=====

===== ERROR =====
500
=====

===== PRINT =====
Terms_BST D
tom/50/2020-07-21/2020-07-21
sophia/25/2023-01-01/2023-01-01
=====
```

DELETE명령어 역시 NAME, DATE 두 가지 경우로 나뉜다. 먼저 NAME과 함께 지우고자 하는 고객의 이름 name이 붙는다면 name을 가지는 고객의 개인정보를 nameBST, termsBST 모두에서 삭제합니다. 성공적으로 삭제를 완료했다면 Success를 출력합니다.

DELETE를 하기 전과 후 PRINT 명령을 실행했을 때 DELETE가 진행된 후 삭제된 고객은 더 이상 출력되지 않는 것을 확인함으로써 DELETE가 잘 수행됐음을 알 수 있습니다.

```

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

===== ERROR =====
500
=====

===== ERROR =====
500
=====

===== ERROR =====
500
=====

===== ERROR =====
500
=====

===== ERROR =====
500
=====

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

```

DELETE명령어 뒤에 DATE와 함께 개인정보수집만료일자 date가 온다면 개인정보수집만료일자가 date 이전인 고객들의 개인정보를 nameBST, termsBST 모두에서 삭제합니다.

삭제가 제대로 수행됐다면 Success를 출력합니다.

모든 고객의 개인정보가 삭제되도록 가장 큰 날짜를 인자로 주었더니 모든 고객의 개인정보가 삭제되어, PRINT를 수행하려고 해도 남아있는 고객정보가 없기에 ERROR메시지를 띄우는 모습입니다.

5. Consideration

고객의 개인정보에는 개인정보수집일자, 개인정보만료일자 등 0000-00-00 형식의 날짜 정보가 포함된다. 이러한 날짜 정보를 년, 월, 일 각각 따로 변수에 저장해준다면, 개인정보를 입력 받은 후 날짜 정보를 년, 월, 일로 구분 지어 주는 것 또한 꽤나 복잡한 작업이 될 수도 있었다. 하지만 이번 실습에서는 날짜 정보를 통해 진행하는 연산이 비교연산 뿐이다. 0000-00-00 형식의 문자열은 그 자체로 충분히 날짜 크기를 비교할 수 있기에 0000-00-00 형식 그대로 저장한 것이 알고리즘 구현에 있어 코드를 조금이나마 최적화 시키는데 도움이 됐던 것 같다.

이번 실습을 진행하기 전, 프로젝트 제안서를 보고, 꽤나 막막함을 느꼈습니다. 그래서 일단 뭐라도 진행시켜야겠다는 조급함에 일단 BST class 정의를 시작하고 도중에 Manager를 작성시작 하는 등, 사전에 계획을 세우지 않고 마구잡이식으로 코드를 짜려

고 하다 보니, 코드가 간결하지 못하고 점점 복잡해지고, 그러다 보니, 코드를 작성하는데 시간 또한 배로 걸리는 것 같은 느낌이 들었습니다. 매 과제마다 Flowchart와 수도코드 등을 작성하도록 하는 이유를 알게 되었습니다. 이번 실습을 통해 다음부터는 사전에 어느정도 구조를 잡아 놓고 코딩을 시작할 수 있을 것 같습니다.