Data Structures_Porject_01

학　　과: 컴퓨터공학과

담당교수: 최상호 교수님

학　　번: 2020202037

성　　명: 염정호

1. Introduction

1)목적

본 프로젝트는 BST(Binary Search Tree), 큐(Queue), 연결리스트(Linkedlist) 자료구조를 이용한. 간단한 개인정보 관리 프로그램을 구현해야 한다.

2)세부사항

- 해당 프로그램은 데이터파일(data.txt)로부터 회원이름, 나이, 개인정보수집일자, 가입약관종류를 읽어와 Queue를 구축한다. 이를 member Queue라 부른다.
- 가입 약관에는 A, B, C, D 총 네 가지가 존재하며, 각 6개월, 12개월, 24개월, 36개월의 개인정보 보관 유효기간을 갖는다.

Member Queue

저장된 데이터를 읽어와 구축, 자료구조에 따라 방출되는 순서대로 저장된다.

회원이름은 공백 없이 소문자로 구성되며, 중복되는 회원이름은 존재하지 않는다.

최대 100개의 데이터를 저장 가능하며, 데이터가 꽉 찬 상태에서 push혹은 데이터가 없는 상태에서의 pop진행시 프로그램을 종료한다.

- Queue에서 pop명령어를 수행하게 되면 데이터를 방출하며 방출된 데이터는 Term_bst, Term_List, Name_bst 로 저장된다.

Term_List : 가입약관 종류(A,B,C,D)별로 노드가 구성되며, 노드는 가입약관 종류, 해당 가입약관의 회원 수, 해당 가입약관의 BST포인터 정보가 저장된다. 가입약관의 순서대로 노드가 생성되고, 연결되며 이미 존재하는 가입약관의 경우 새로운 노드를 생성하지 않고 해당하는 기존 노드의 회원수를 증가시킨다.

Term_bst : 가입약관의 종류별로 생성되며 , 각 BST의 노드는 회원이름, 나이, 개인정보수집일자, 개인정보만료일자를 갖는다. 개인정보 만료일자는 개인정보 수집일자에 약관 별 개인정보 보관일자를 더해 계산된다. 각 BST는 개인정보 만료일자를 기준으로 정렬된다.

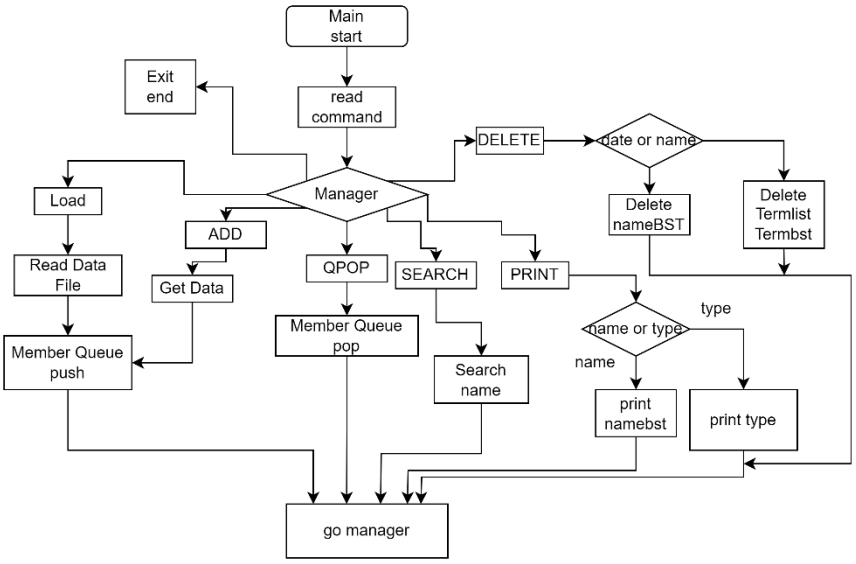
Name_bst : 회원이름, 나이, 개인정보수집일자, 개인정보만료일자, 가입약관종류 정보를 갖는 노드로 구성되며, 회원 이름을 기준으로 정렬된다.

헤더파일	명령어	내용
Manager	Run	command파일을 읽어와 명령어를 수행하고 수행 결과를 log파일에 저장하는 명령어
	PrintSuccess	명령어 수행 성공 문구 출력
	PrintErrorCode	에러코드 문구 출력
	LOAD	data파일로부터 신원정보 데이터를 읽어와 Q에 저장하는 명령어
	ADD	데이터를 Q에 직접적으로 추가하는 명령어
	QPOP	Q에 저장된 데이터를 들어온 순서대로 NameBST와 TermList,TermBST에 저장.
	SEARCH	이름을 입력받아 해당 인원의 정보 출력
	PRINT	Name입력시 NameBST에 저장된 모든 데이터를 출력하고 TYPE입력시 해당 타입의 신원 모두 출력
	DELETE	데이터 삭제, 이름 입력 시 해당 이름을 가진 정보 삭제, 만료일자 입력 시 해당 만료일자 이전의 만료일자를 가진 데이터 삭제
	EXIT	프로그램 종료
MemberQueue	Empty	Q가 비어있는지 확인
	Full	Q에 데이터가 꽉차있는지 확인.
	Push	Q에 데이터 입력 Full상태에서 추가 입력 시 프로그램 종료
	Pop	Q에 데이터 출력 Empty상태에서 추가 입력 시 프로그램 종료
	rear	Rear데이터 출력
MemberQueueNode	SetData	Q노드 데이터 입력
	SetNext	Q노드의 다음 노드 입력
	GetNext	Q노드가 가리키는 노드 출력

	gettype	가입약관 출력
	getname	이름 출력
	getdate	데이터 출력
	getage	나이 출력
NameBST	getRoot	Name BST의 첫 노드 주소 출력
	insert	Name BST에 데이터 추가
	Search& search_p	입력받은 데이터의 노드주소 또는 부모노드 주소를 출력
	print	중위순회 방식으로 데이터 출력
	delete_name	해당 이름 삭제
NameBSTNode	getLeft	왼쪽 노드 주소 출력
	getRight	오른쪽 노드 주소 출력
	setLeft	왼쪽 노드 입력
	setRight	오른쪽 노드 입력
	setdata	데이터 입력
	getname	이름 출력
	getage	나이 데이터 출력
	getstartdate	약관 가입 일자 출력
	getenddate	약관 만기일자 출력
	gettype	가입 형태를 출력
	dataCopy	다른 노드의 데이터를 복사해온다.
TermsBST	getRoot	BST시작주소 출력
	setRoot	시작주소 입력
	insert	TermBST에 데이터 추가
	print	TemrBST에서 해당 약관 데이터 출력
	delete_Term	TermBST에서 데이터 삭제
	level_travel_distory	대상 노드를 포함한 모든 하위노드 해제 및 삭제
	delete_term_name	nameBST에서 삭제된 데이터를 TermBST에서도 삭제
TermsBSTNode	getLeft	왼쪽 노드 주소 출력
	getRight	오른쪽 노드 주소 출력
	setLeft	왼쪽 노드 주소 입력
	setRight	오른쪽 노드 주소 입력
	setdata	데이터 입력
	getname	이름 정보 출력
	getage	나이 정보 출력
	getstartdate	가입일자 정보 출력
	getenddate	만료일자 출력
	dataCopy	노드 데이터 복사.
TermsLIST	getHead	TermLIst의 시작 주소를 불러옴
	insert	TermList에 가입약관별 데이터 추가, 이미 존재하는 약관일 경우 인원수만 증가
	search	해당 타입의 노드를 출력
	delete_list	리스트에서 데이터 삭제 인원이 0명일 경우 해당 노드 해제
	delete_list_name	nameBST에서 지워진 데이터를 List에서도 삭제
TermsListNode	getNext	다음 노드 주소 출력
	setNext	다음 노드 주소 입력
	settype	가입약관 입력
	gettype	가입약관 출력
	getmember	인원수 증가.
	setmember	사라진 인원만큼 member감소
	increase_member	인원수 감소
	setTermBST	BST생성
	gettermbst	해당 bst주소값

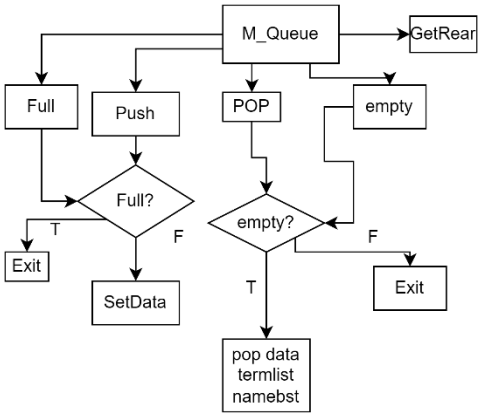
2. Flowchart

Main&Manager flowchart



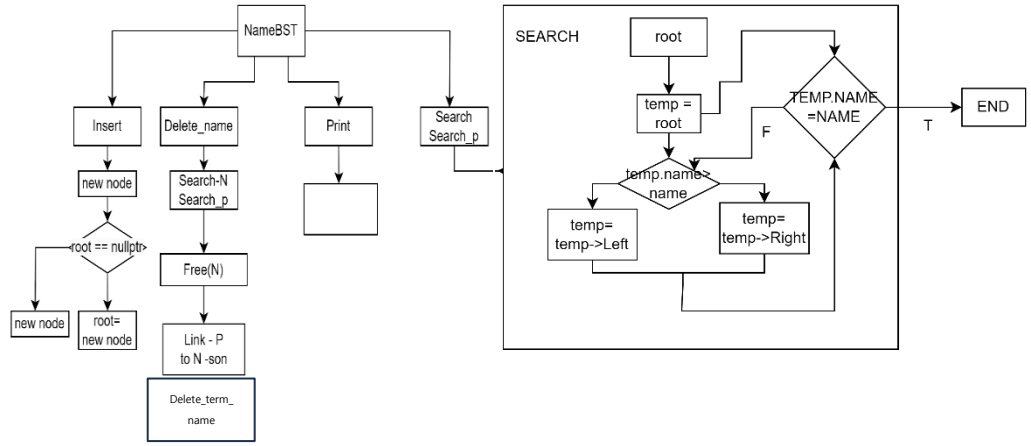
Command를 입력 받고 manager에서 명령어에 따라 실행, Exit명령어 입력 시 종료하게 된다. Exit이외의 명령어는 수행 후 다음 명령어를 입력 받는다.

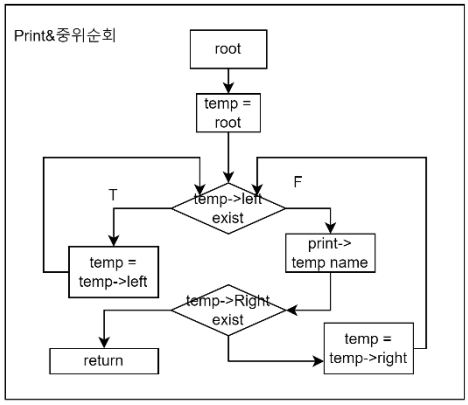
M_Queue flow charts



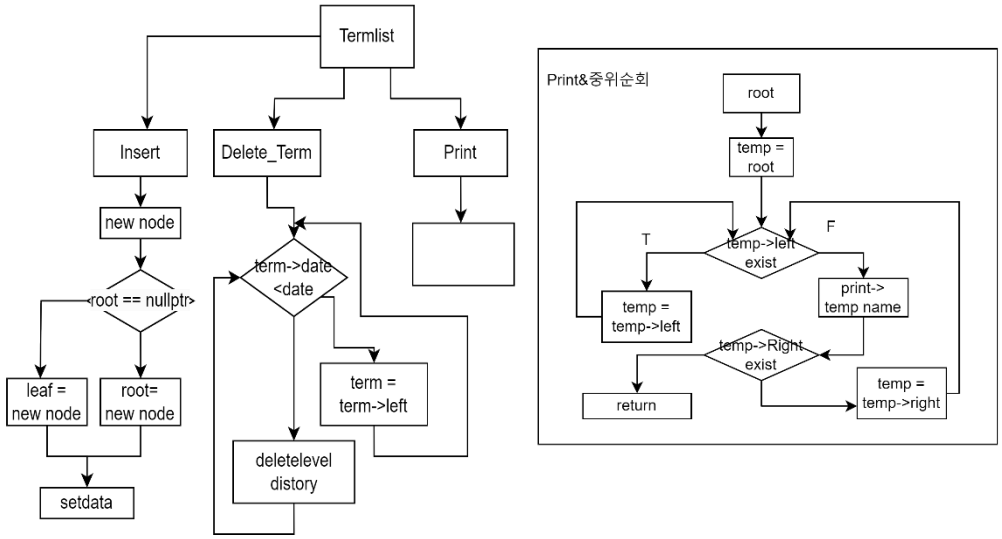
Pop date 실행시 termlist, namebst로 이동, 데이터 저장

NameBst flow chart



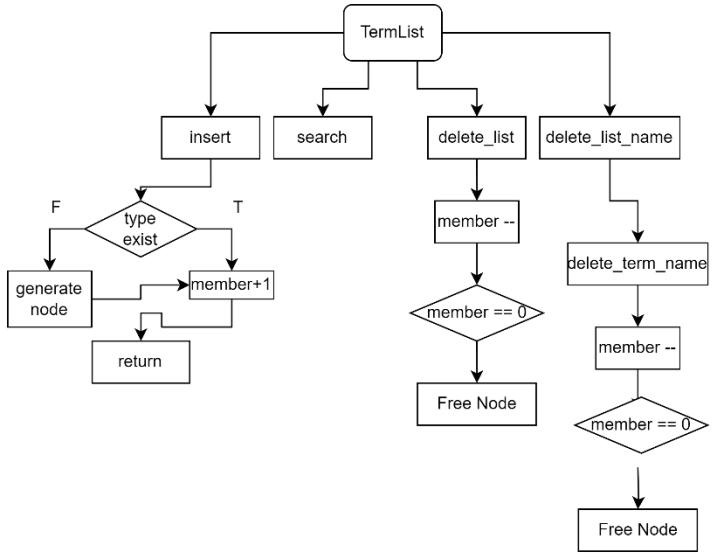


TermsBST



Distory함수 실행 시 term을 포함한 모든 왼쪽 노드가 파괴되고 term은 자신의 오른쪽 노드로 이동
날짜가 더 큰 경우 왼쪽 노드로 이동해 다시 판별 term에 null값이 들어갈 때 까지

TermsList



deletelist삭제된 데이터 수 만큼 list맴버에서 삭제, 맴버가 존재하지 않을경우 메모리 해제

3. Algorithm

Manager.cpp{

파일 입력,출력 함수 실행->파일 존재하지 않을 시 에러출력;

반복{

Command를 한 줄 씩 읽어옴;

명령어를 명령어와 인자로 구분해서 사용;

}

if명령어만 존재시 해당 명령어로 이동;

명령어 실행;

else 인자가 필요한 명령어를 받은 경우 해당 명령어로 이동;

명령어 실행;

}

MemberQueue{

생성자: 변수 초기화;

Empty: 큐가 비어 있는지 확인한다. ;

Full: 큐가 다 차 있는지 확인한다. ;

Push{

큐에 값저장, 이미 full인 경우 프로그램 종료;

들어온 값이 첫 값일 경우 헤드 주소로 설정;

데이터 세팅;

처음 들어온 데이터가 아닐경우 rear위치로 이동해 노드 연결 후 값 세팅;

}

Pop{

출력할 값이 없을 경우 프로그램 종료;

아닌 경우;

T-List, Name_List에 값 저장;

front주소 이동;

이전 메모리 해제;

}

Rear{

rear주소 반환

}

}

NameBST{

생성자 :루트 값 초기화

Insert : root가 존재하지 않을 경우->새로 생성된 노드가 루트 노드가 된다.

처음 들어온 노드가 아닌 경우:

리프노드에 데이터 세팅 후 탐색을 위해 데이터 들어갈 위치 확인.

값에 따라 branch의 왼쪽 혹은 오른쪽 노드에 값 세팅

Search&Search_p{

루트 노드부터 탐색

노드 마다 이름을 비교해서 노드 위치 탐색

사전순위가 더 빠른 이름일 경우 왼쪽,

느린 순서일 경우 오른쪽 노드로 이동

Data가 존재하지 않을 시 nullptr출력

}

Print(){

중위순회 방식을 이용해 BST탐색

사전순위에 따라왼쪽 또는 오른쪽 노드로 재귀;

왼쪽 노드 값이 없을 경우 자기자신 출려;

오른쪽 노드에 값 확인 후 재귀

}

Delete_name{

입력받은 이름을 찾아서 제거

해당노드의 자식노드의 개수에 따라 다르게 실행

0개일 경우 : 메모리 해제

1개일 경우 : branch노드 연결 후 해제

2개일 경우 : 왼쪽 노드의 가장 큰값 혹은 오른쪽 노드의 가장 작은 값 중 하나를 선택해 해제한다

메모리 해제;

}

}

NameBST{

Insert{

root가 존재하지 않을 경우->새로 생성된 노드가 루트 노드가 된다.

처음 들어온 노드가 아닌 경우:
리프노드에 데이터 세팅 후 탐색을 위해 데이터 들어갈 위치 확인.
값에 따라 branch의 왼쪽 혹은 오른쪽 노드에 값 세팅
}

Print(){

중위순회 방식을 이용해 BST탐색
사전순위에 따라왼쪽 또는 오른쪽 노드로 재귀;
왼쪽 노드 값이 없을 경우 자기자신 출려;
오른쪽 노드에 값 확인 후 재귀
}

level_travel_distory{

입력받은 노드를 포함 한 모든 하위 노드 해제;
NameBST로 이름을 넘겨 해제후
해당 노드 해제 count를 1 증가시킴
Count를 리턴해 몇 개의 노드가 사라졌는지 알 수 있다.
}

Delete_Term{

Term에서 노드제거,
해당 노드가 이전 날짜인지 이후 주소인지 확인한다;
이전 날짜라면 비교 대상 노드를 포함한 왼쪽 노드를 모두 해제하기 위해
Level_travel_distroy실행, 오른쪽 노드가 루트 노드가 된다.
비교 노드가 이후 날짜라면 비교 노드의 왼쪽으로 이동해서 재귀함수 실행.
제거된 노드의 개수만큼 termlist의 member개수를 줄여준다.
member개수가 0인 노드는 해제
}

Delete_term_name{

Name_BST에서 삭제한 노드를 Term에서 삭제해준다.
해당 노드를 삭제할 때에는 NameBST에서 삭제한 방법을 이용해
Term BST에서도 삭제한다.
}

}

TermsList{

Insert :

처음 들어온 값이라면 새로운 노드를 헤드노드로 설정
반출된 값을 분석해 노드에 저장.
TermsBST head 생성
이미 해당 타입 노드가 존재할 경우
맴버수를 증가시키고 TermsBST주소에 연결시킨다.

Search: 해당 타입의 노드가 존재하는지 확인
타입이 존재 할 경우 노드 위치 반환
존재하지 않을 경우 nullptr;

delete_list: 리스트의 멤버를 감소시키는 함수
termsBST에서 해제된 메모리만큼 member감소;
member수가 0일 경우 해당 노드 해제 이전 노드와 해당 노드의
다음 노드 연결

delete_list_name : NameBST에서 제거한 노드를 Terms에서도 제거,
타입을 이용해 위치를 찾고 맴버 변수 감소;
Enddate를 기준으로 정렬 되어있기 때문에 enddate를 이용해 해당 노드를 탐색하고
데이터 제거, NameBST에서 삭제한 방법을 이용해 Term BST에서도 삭제한다.

}

4. Result Screen

```
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 Files

```
LOAD
ADD tom 50d2020-07-21 D
ADD tom 50 2020-07-21 D
ADD bella 94 2023-08-31 B
ADD harry 77 2024-02-03 B
```

1. 데이터 로드 성공
2. 첫 ADD입력 오류
3. ADD입력 * 4

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

1. 데이터 이동
2. bob데이터 출력
3. tom데이터 출력
4. Name_BST 출력

```
===== 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 A
PRINT B
PRINT C
PRINT D
```

각 가입약관 별로 데이터 출력

```
===== PRINT =====
Terms_BST
chris/20/2022-11-05/2022-17-05/
taylor/11/2023-02-20/2023-08-20/
bob/31/2023-02-22/2023-08-22/
================
===== PRINT =====
Terms_BST
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 =====
Terms_BST
emily/41/2021-08-01/2023-08-01/
================
===== PRINT =====
Terms_BST
tom/50/2020-07-21/2023-07-21/
sophia/25/2023-01-01/2026-01-01/
================
```

NameBST 출력

```
===== 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/
================
```

```
DELETE NAME emily
PRINT C
DELETE DATE 2024-09-01
PRINT NAME
```

1. emily 데이터 제거
2. C 타입약관 출력

   ->존재 X
3. 2024-09-01

   이전 만료일자 전부 제거
4. 데이터 출력

```
===== DELETE =====
Success
===============

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

===== 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/
================
```

```
PRINT A
PRINT B
PRINT D
EXIT
```

1. 약관별 데이터 출력

   A 존재 X

2. 프로그램 종료

```
_____________
===== ERROR =====
500
===============

===== PRINT =====
Terms_BST
kevin/58/2023-09-01/2024-09-01/
harry/77/2024-02-03/2025-02-03/
================
===== PRINT =====
Terms_BST
sophia/25/2023-01-01/2026-01-01/
================
===== EXIT =====
Success
===============
```

5. Consideration

1. 오랜만에 하는 linkedlist이다 보니 Node구성과 cpp파일 구성을 알기 위해 지난 객체지향프로그래밍 과제를 많이 찾아봐야 했다.
2. data파일이 로드가 안되는 경우가 존재했는데. 데이터 파일 위치를 sln파일 위치에 넣어서 파일이 로드되지 못했다.
3. Pop진행시 반환형을 노드 포인터 형태로 진행하였었는데 값을 리턴 할 필요가 없어 "return" 으로 선언하니 프로그램 오류가 발생하였다.
4. 프로젝트를 계속 진행하면서 LNK2019라는 에러를 많이 접하게 되었는데 처음엔 에러 위치를 알아도 어느 부분이 잘못되었는지 몰랐다, class형 변수를 생성 할 때 존재하지 않는 class형으로 정의하거나 오타가 있을 경우 발생하는 에러였고, 자세히 읽어보면 어느 부분에서 에러가 발생하였는지 알 수 있었다.
5. NameBST데이터 입력 형태를 강의 자료와 같은 형태로 입력되겠더니 하면서 오류를 찾았는데 강의 자료이 예시가 잘못된 것이었다. (실제로 그려서 알게되었다.)
6. 헤더파일을 선언할 때 Name과 Term에서 서로를 헤더파일로 하니 오류가 발생해 서로를 헤더파일로 사용할 수 없다는 것을 알게되었다.