

# 데이터 구조 실습

## Project #1

학 과: 컴퓨터공학과

팀 장: 김동현

학 번: 2012722071

팀 원: 이상효

학 번: 2012722037

### 1) Introduction

=> 이번 프로젝트에서는 영어 단어장 프로그램을 구현한다. 이 프로그램엔 3가지 단어장이 구현되어 있는데, 외워야 할 단어장(TO\_MEMORIZE), 현재 외우고 있는 단어장(MEMORIZING), 다 외운 단어장(MEMORIZED)으로 구성된다. 3가지의 단어장은 다 다른 LinkedList의 형태로 구현되어

있으며, command.txt에 있는 각각의 명령어를 통해 단어장 내에 단어들을 서로 이동하면서, 자유 자재로 단어장을 다룬다.

#### (1) 외워야 할 단어장(TO\_MEMORIZE)

- Queue를 이용하여 구현한다. Queue는 first in first out의 구조로 Push 된 순서대로 Pop 하는 특징을 가지고 있다. word.txt파일에 있는 단어를 Queue를 이용 해 Push하고, 외울 단어를 Pop하여 현재 외우고 있는 단어장(MEMORIZING)으로 넘긴다. 또한, 찾고 싶은 단어를 찾을 수 있고, 단어의 뜻을 바꾸고 싶다면 바꿀 수 있고, Push한 순서대로 출력하고 TO\_MEMORIZE.txt 파일에 저장한다. 마지막으로 각각의 명령어에 따른 예외처리를 구현한다.

#### (2) 현재 외우고 있는 단어장(MEMORIZING)

- BST를 이용해 구현한다. BST는 root를 기준으로 root보다 작으면 왼쪽, 크면 오른쪽으로 뻗어 나가는 트리 구조이다. 먼저 알파벳(A~Z) BST를 구현하고 A~Z의 Node 안에 각각 단어 BST가 구현되어있다. 다시 말해서, BST안에 BST가 구현되어 있는 것이다.

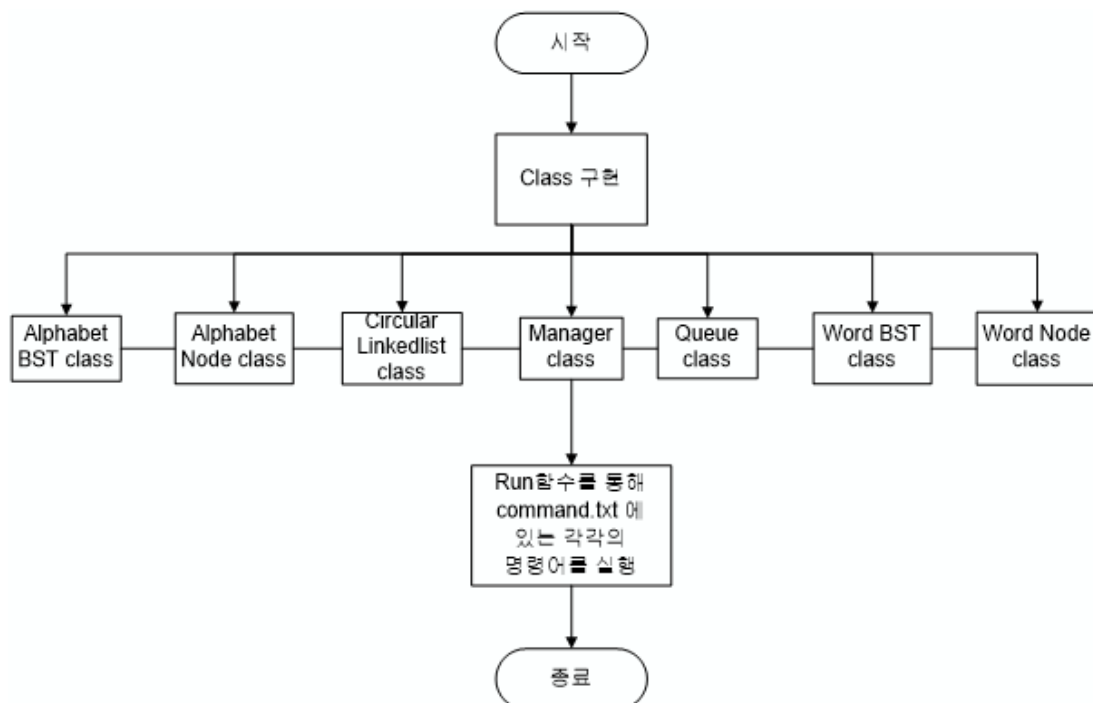
TO\_MEMORIZE에서 넘어온 Node를 먼저 알파벳 BST에서 첫 글자를 비교해서 어느 Node로 들어갈 지 찾은 다음 단어BST에서 크기 비교를 통해 새로 Insert한다. 또한, 다 외웠다면, 해당 단어가 들어있는 Node를 BST에서 삭제하고, MEMORIZED 단어장으로 넘긴다. BST에서 삭제할 땐, 양쪽 자식이 모두 존재하는 경우 왼쪽 Subtree에서 가장 큰 Node를 삭제되는 Node의 위치로 옮긴다. 또한, Print명령어를 통해, 어떤 순회 방식(pre-order, in-order, post-order, level-order)을 선택해서 순회하여 출력할 지 결정하여 출력하고, 단어의 뜻을 바꾸고 싶으면 찾아서 바꾸고, MEMORIZED.txt 파일에 저장한다. 이 때 주의 해야 할 점은 MEMORIZING.txt 파일에 단어를 알파벳순으로 저장하면, LOAD할 때 한쪽으로 치우쳐진 트리 구조가 되어 버리기 때문에 pre-order방식을 이용한다. 마지막으로, 각각의 명령어에 따른 예외처리를 구현한다.

#### (3) 다 외운 단어장(MEMORIZED)

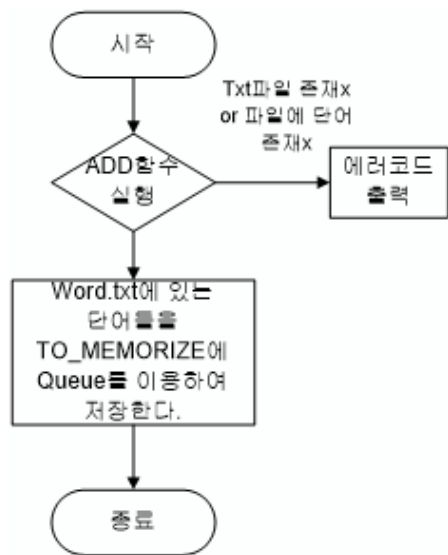
- Circular Linked-list를 이용하여 구현한다. 이전에 배운 single Linked-list와 구조는 동일하며 맨 마지막 Node를 Head와 연결하여 Circular를 이루면 된다. Head를 가리키는 포인터와 Tail을 가리키는 포인터를 이용해 구현하고, MEMORIZING에서 넘어온 Node를

Head와 Tail사이에 연결한다. 또한, 찾고 싶은 단어를 찾을 수 있고, 단어의 뜻을 바꾸고 싶으면 찾아서 바꿀 수 있고, Queue와 마찬가지로, Insert된 순서대로 출력하고, MEMORIZED.txt 파일에 저장한다. 마지막으로 각각 명령어에 따른 예외처리를 구현한다.

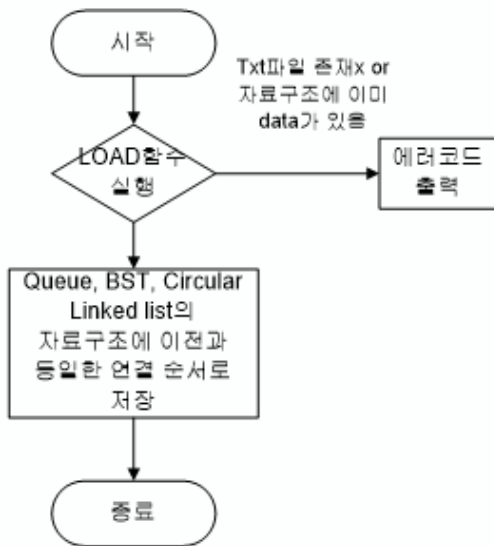
## 2) Flow Chart



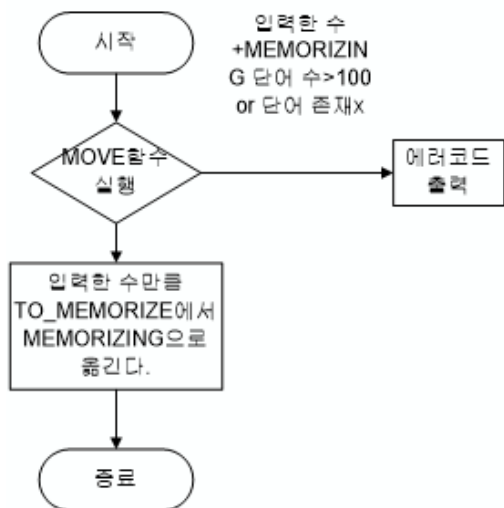
이번 Project에 사용되는 전체적인 클래스들을 나타내고, Main 함수에서 Manager class의 객체를 생성해 run함수를 통해 각각의 명령어들을 실행한다. 또한 아래의 Flow chart들은 각각의 명령어들에 의해 실행되는 함수들의 구현을 나타낸다.



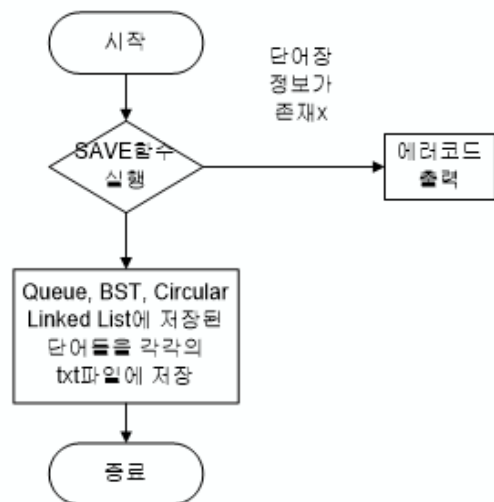
ADD함수



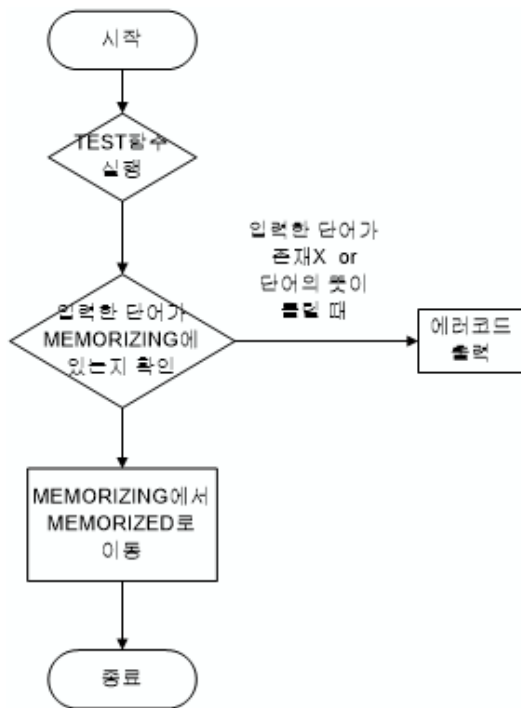
LOAD함수



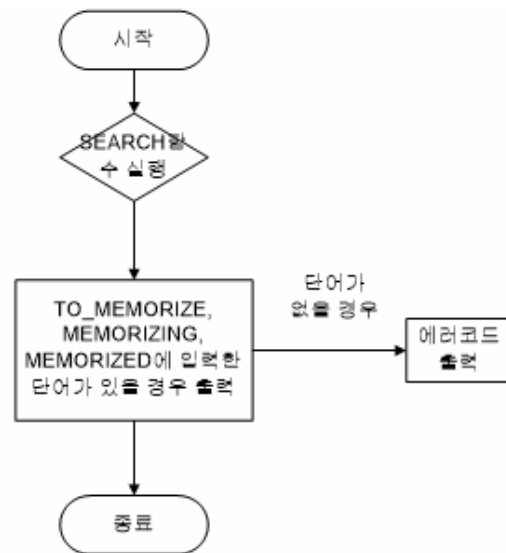
MOVE함수



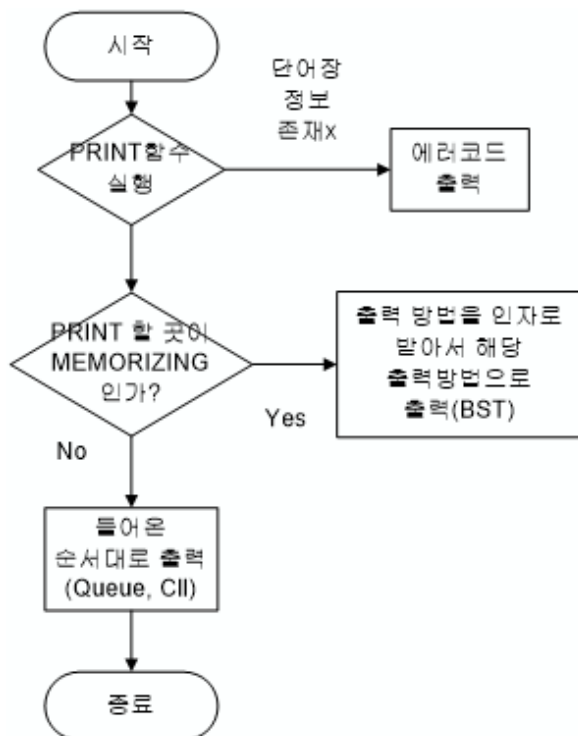
SAVE함수



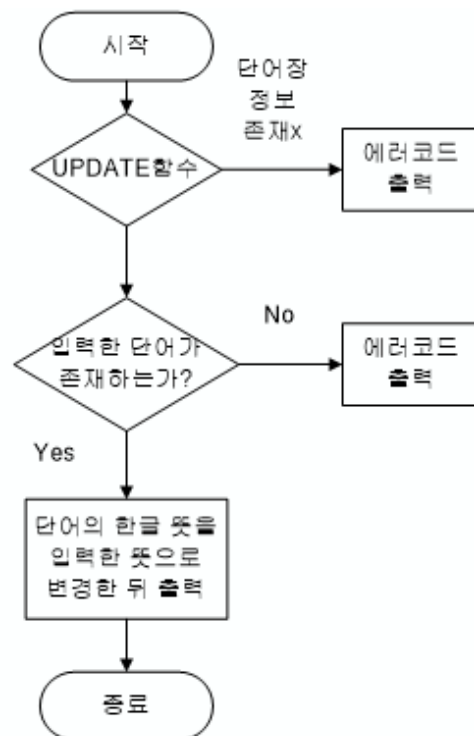
TEST 함수



SEARCH 함수



PRINT 함수



UPDATE 함수

### 3) Algorithm

이번 프로젝트에서는 3가지로 나뉜 영어 단어장을 구현한다. 외워야 될 단어장(TO\_MEMORIZE) 외우고 있는 단어장(MEMORIZING) 다 외운 단어장(MEMORIZED)으로 구분된다. 먼저 TO\_MEMORIZE는 Word.txt에 있는 모든 단어를 Queue를 이용해 구현한다. 다음으로 MEMORIZING은 이중 BST로 구현 되는데, 먼저 Alphabet BST를 구현하고, 각각의 알파벳 노드 안에 그 알파벳으로 시작하는 단어들이 Word BST를 이루고 있다. BST를 구현 하려면 BST의 규칙을 알아야 하는데, 먼저 부모의 노드보다 단어의 사전 순서가 빠르면 왼쪽, 늦으면 오른쪽으로 연결된다. 또한 삭제를 할 때 양쪽 자식 노드가 모두 존재할 경우 왼쪽 서브 트리에서 가장 큰 노드를 제거되는 노드의 자리에 이동시킨다. 만약 왼쪽 노드가 존재하지 않으면, 오른쪽에서 가장 작은 노드를 이동시킨다. 마지막으로 MEMORIZED는 Circular Linked-list로 구현한다. single Linked-List에서 tail을 head와 다시 연결해 준 구조이다. 이렇게 3가지로 나뉜 영어 단어장이 각각 다른 Linked-list의 구조로 연결되어 있다. 이제 이 단어들을 각각의 명령어들을 통해 다룬다. 명령어들에는 LOAD, ADD, MOVE, SAVE, TEST, SEARCH, PRINT, UPDATE, EXIT가 있다. 각각 이들이 bool함수로 정의 되어 참 거짓을 판별해서 참이면 함수를 실행하여 결과를 도출하고, 거짓이면 에러코드를 출력한다.

(1) LOAD: 기존의 단어장을 불러오는 명령어로 각각 to\_memorize.txt, memorizing.txt, memorized.txt 파일에 이전과 동일한 순서를 가지도록 연결하여 저장한다. 파일이 존재하지 않으면 에러코드를 출력한다.

(2) ADD: Word.txt에 있는 단어들을 전부 TO\_MEMORIZE에 Queue의 형태로 저장한다. 파일이 없거나, 파일에 단어가 존재하지 않다면 에러코드를 출력한다.

(3) MOVE: ex)MOVE 100 이면 100개의 단어들을 TO\_MEMORIZE에서 MEMORIZING으로 옮긴다. 입력 받은 수는 1과 100 사이이며, 입력 받은 수와 MEMORIZING에 있는 단어 수의 합이 100을 넘지 않도록 한다. 또한 TO\_MEMORIZE에 있는 단어 수 보다 입력 받은 수가 많아도 안되므로 이러한 경우엔 에러코드를 출력한다.

(4) SAVE: 현재 단어장의 정보를 각각 to\_memorize.txt, memorizing.txt, memorized.txt에 저장한다. LOAD 명령어와 유사한 명령어이다. 역시 파일이 존재하지 않다면 에러코드를 출력한다.

(5) TEST: MEMORIZING에 있는 단어를 TEST를 통해 입력한 단어가 이곳에 있는지 확인한 후 단어의 뜻까지 맞다면, MEMORIZED로 단어를 옮긴다. 이때, BST의 삭제 규칙에 의해 MEMORIZING에서 삭제하고, Circular Linked-list에서 head와 tail 사이에 이 단어를 연결한다. 입력한 단어가 MEMORIZED에 존재하지 않거나, 단어의 뜻이 틀리면 에러코드를 출력한다.

(6) SEARCH: 단어의 뜻을 찾아서 출력한다. TO\_MEMORIZE, MEMORIZING, MEMORIZED에 각각 이 단어가 있는지 확인하고, 있다면 그 뜻을 출력한다. 만약 3군데 다 없다면 에러코드를 출력한다.

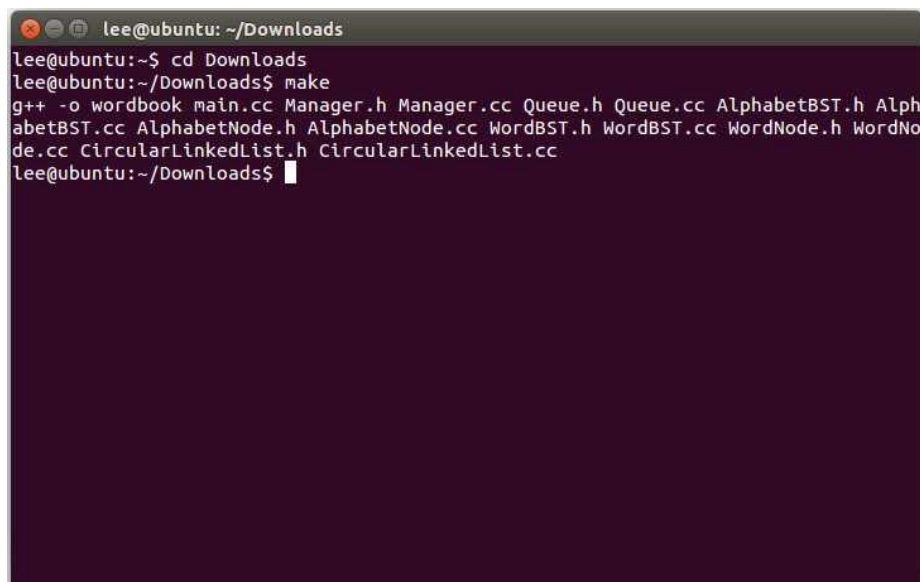
(7) PRINT: 단어장에 있는 단어들을 전부 출력한다. TO\_MEMORIZE와 MEMORIZED 는 그냥 들어온 순서대로 출력하면 된다. 한편, MEMORIZING에 있는 BST구조를 출력할 때는 출력하는 방법이 여러 가지가 있다. In-order, post-order, pre-order, level-order가 있는데, 이를 각각 재귀함수와 반복문을 사용하여 구현해서 출력을 한다. 만약 단어장의 정보가 존재하지 않는다면 에러코드를 출력한다.

(8) UPDATE: 입력 받은 단어가 TO\_MEMORIZE, MEMORIZING, MEMORIZED에 각각 있는지 확인한 후, 있으면 이를 한글 뜻으로 입력한 뜻으로 변경한 뒤, 출력한다. 만약 입력 받은 단어가 3군데에 전부 없다면 에러코드를 출력한다.

이와 같은 명령어들을 통해 3개로 나뉘어진 단어장 들을 관리한다. 이러한 모든 관리는 Manager class에서 이루어지며 모든 명령을 받고 이 프로그램의 총 지휘를 하는 역할을 한다.

## 4) Result Screen

<file을 make 했습니다.>



```
lee@ubuntu: ~/Downloads
lee@ubuntu:~$ cd Downloads
lee@ubuntu:~/Downloads$ make
g++ -o wordbook main.cc Manager.h Manager.cc Queue.h Queue.cc AlphabetBST.h AlphabetBST.cc AlphabetNode.h AlphabetNode.cc WordBST.h WordBST.cc WordNode.h WordNode.cc CircularLinkedList.h CircularLinkedList.cc
lee@ubuntu:~/Downloads$
```

<./wordbook 명령어를 이용해서 실행합니다.>

<보시는 바와 같이 ADD, LOAD, SAVE, MOVE의 경우가 완벽하게 실행되고 있습니다.>

```
lee@ubuntu: ~/Downloads
lee@ubuntu:~$ cd Downloads
lee@ubuntu:~/Downloads$ make
g++ -o wordbook main.cc Manager.h Manager.cc Queue.h Queue.cc AlphabetBST.h Alph
abetBST.cc AlphabetNode.h AlphabetNode.cc WordBST.h WordBST.cc WordNode.h WordNo
de.cc CircularLinkedList.h CircularLinkedList.cc
lee@ubuntu:~/Downloads$ ./wordbook
=====ADD=====
Success
=====LOAD=====
Success
=====SAVE=====
Success
=====MOVE=====
Success
=====
pet 애완동물
leaf 잎
area 지역
art 미술
subject 주제
machine 기계
```

<이어서 PRINT함수 또한 잘 실행이 되고 있습니다.>

```
light 빛
watch 시계
=====PRINT=====
pet 애완동물
leaf 잎
area 지역
art 미술
subject 주제
machine 기계
rule 규칙
busy 바쁜
health 건강
goal 목표
result 결과
decide 결정하다
difficult 어려운
brain 뇌
opinion 의견
price 가격
subway 지하철
human 인간
sell 팔다
miss 놓치다
delicious 맛있는
```

<SERACH의 경우 이상한 값이 나오는 이유를 찾지 못하고 시간이 부족하여 제출하였습니다.>

<UPDATE, TEST 의 경우는 실험을 진행하지 못하여 사진첨부를 못했습니다.>

```
mind 마음
visit 방문
light 빛
watch 시계
=====SEARCH=====
♦싼
ability 능력
action 행
=====
lee@ubuntu:~/Downloads$ z
```



## 5) Consideration

이름	프로젝트에서 맡은 역할	본인이 스스로 생각하는 자신의 점수(10)
김동현	Queue, Circular Linked-list, Manager, MEMORIZING의 Delete, I_PRE, I_POST 구현	5
고찰	<p>이전에 배운 고급프로그래밍 시간에 나온 과제들에 비해 구현해야 할 것이 너무 많아졌고, 처음에 과제를 딱 받았을 때, Class도 너무 많고 복잡해서 무슨 소린지 잘 이해하지 못하였다. 또한 예외처리를 해야 하는 것도 많고, 디버깅을 하는데 너무 많은 시간이 걸렸다. 한편, 팀 프로젝트를 학교에서 처음 진행해 보았는데, 팀이 있어서 나눠서 해도 된다는 생각에 과제 시작을 너무 늦게 한 감도 있었고, 팀원과 소스를 합쳐서 프로젝트를 완성할 때 팀원이 짰 코드까지 이해를 해야 하는데 남의 소스를 이해하는데 좀 어려움이 있었다. 다음에 또 팀 프로젝트를 할 기회가 있다면, 이번을 계기로 좀 보완해서 잘 해보고 싶다.</p>	

이상호 (팀원):

이름	프로젝트에서 맡은 역할	본인이 스스로 생각하는 자신의 점수(10)
이상호	AlphabetBST, WordBST, MEMORIZING의 I_IN, R_IN, R_PRE, R_POST, Levelorder	7
고찰	<p>- 내가 맡은 부분</p> <p>저는 이번 프로젝트에서 AlphabetTree를 만들고 그 안에 WordTree 다시 만드는 부분을 맡았습니다. 물론 Search, Print, Insert, 등 부분적인 함수도 전부 구현하였습니다.</p> <p>- 추가로 구현한 함수</p> <p>Expressword: Save함수 구현 시 fout을 이용해서 txt file에 값을 표출하는데 내가 구성한 알고리즘의 경우는 Inorder로 AlphabetTree를 순회하여 알파벳 순서대로 word값을 받아오는 방법을 선택하였다. 해당 노드의 word를 모두 받아온 후 Save함</p>	

	<p>수로 넘겨주는 함수가 필요하였다. 2차원 배열로 Save에 word들을 넘기면 Save는 txt파일에 한 라인씩 word값을 표출한다.</p> <p>Define: node의 wordmean 변수에 word와 mean을 한번에 넘겨주는 함수이다.</p> <p>-구현하며 힘들었던 점</p> <ol style="list-style-type: none"><li>1. 노드 안에 새로운 Tree를 생성하는 점이 가장 힘들었다. word를 받아와 AlphabetTree를 순회하여 word의 가장 앞 알파벳과 비교한 후 같은 알파벳을 지니고 있는 노드의 bst변수를 이용해서 다시 WordTree를 만들 때 많은 에러와 시행착오를 겪었다.</li><li>2. Visual Studio 2015 ver 에서 linux로 코드를 옮겼을 때 무수히 많은 오류가 나며 프로그램이 실행되지 않음에 상당히 놀랐다. 명령어를 익힌 후 'make'와 프로그램을 실행함으로 서 많은 오류를 해결할 수 있었다. 디버깅을 하는 법을 몰라서 오류가 나는 해당 함수를 찾아서 printf를 해가며 결과 값을 확인하는 작업도 신선했다.</li><li>3. 학교에서 컴퓨터언어 관련 수업을 들으면서 팀 프로젝트는 처음 해보았다. 팀원과 각자 할 일을 정해서 프로그램을 만드는 작업이 처음에는 쉬울 줄 알았지만 그렇게 녹록한 일이 아니라는 것을 프로젝트 진행하면서 깨달았다. 현재 팀원은 2명이었지만 더욱 인원이 많아지면 공통화 작업을 통해서 서로의 코드를 잘 이해 할 수 있도록 하는 부분부터 고쳐나가야겠다.</li></ol>
--	---