Data Structure Report

Project 01

Professor	이기훈 교수님
Department	Computer engineering
Student ID	2015722 087, 070
Name	김민철, 고은나래
Class	설계,실습 (C)
Date	2016. 10. 07

Signature _____

1. Introduction

이번 프로젝트는 이진 탐색 트리(Binary Search Tree, BST)와 환형 연결 리스트(CircularLinked List), Queue를 이용하여 영어 단어장 프로그램을 구현하는 것이다. 여러 영어 단어와 한글 뜻을 입력 받아 외워야 할 단어장(TO_MEMORIZE), 외우고 있는 단어장(MEMORIZING), 이미 외운 단어장(MEMORIZED) 세가지로 나누어 설계한다. 외워야 할 단어장은 Queue 자료구조를 이용하고, 외우고 있는 단어장은 BST 자료구조를 이용하고, 이미 외운 메모장은 CircularLinkedList 자료구조로 설계한다.

프로그램이 동작할 명령(command)는 텍스트 파일을 통한 입출력으로 구현하는데, 프로그램이 동작 가능한 명령에는 LOAD, ADD, MOVE, SAVE, TEST, SEARCH, PRINT, UPDATE가 있다. LOAD는 3가지 단어장을 출력한 텍스트 파일로부터 단어를 불러와서 각 자료구조에 저장하고, ADD는 WORD.TXT 파일에 저장되어있는 단어들을 QUEUE 자료구조에 불러온다. 그리고 MOVE는 큐 자료구조에 저장된 단어들을 입력한 개수만큼 BST로 이동시키고, SAVE는 각 자료구조에 저장되어있는 단어들을 텍스트 파일로 출력한다. TEST는 MEMORIZING 자료구조에 있는 단어의 스펠링과 뜻을 맞추면 MEMORIZED 자료구조로 이동시키는 기능이며, SEARCH는 입력한 단어를 3가지 자료구조 안에서 찾아 뜻과 함께 화면에 나타내주는 기능이고, PRINT는 입력한 자료구조를 입력한 오더 방식으로 화면에 출력한다. 마지막으로 UPDATE기능은 입력한 단어를 찾아 기존의 뜻을 새롭게 입력한 뜻으로 바꾸는 기능이다.

만약 명령어에 따른 기능이 실행되지 못하면 각 동작에 맞는 에러코드를 출력하고, 화면 출력 시 정해진 출력 포맷을 따르며, 모든 출력은 log.txt 파일에 출력한다.

2. Algorithm

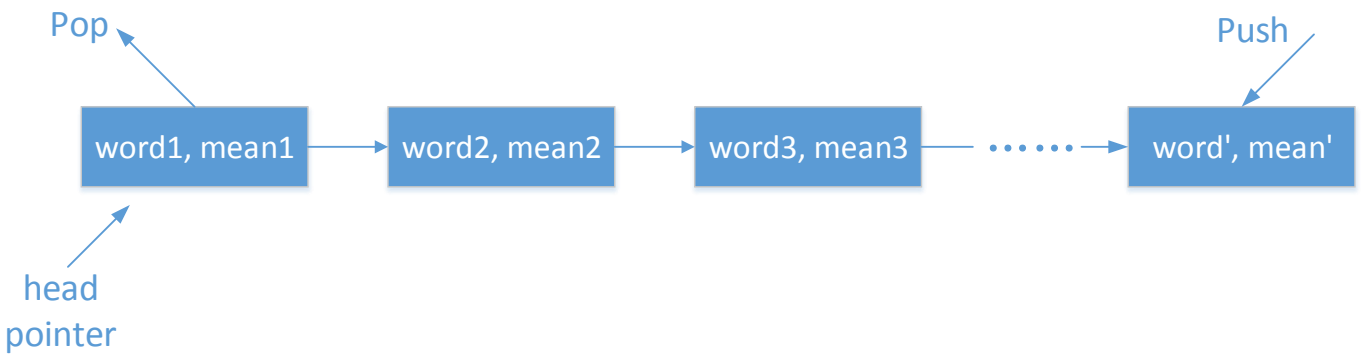
-단어장

구현하려는 단어장 은 총 세 개로, TO_MEMORIZE, MEMORIZING, MEMORIZED로 구성 되어있다. TO_MEMORIZE 단어장은 Queue로, MEMORIZING 단어장은 BST로, MEMORIZED는 Circular Linked list로 구현한다.

<TO_MEMORIZE 단어장 – 'Queue'>

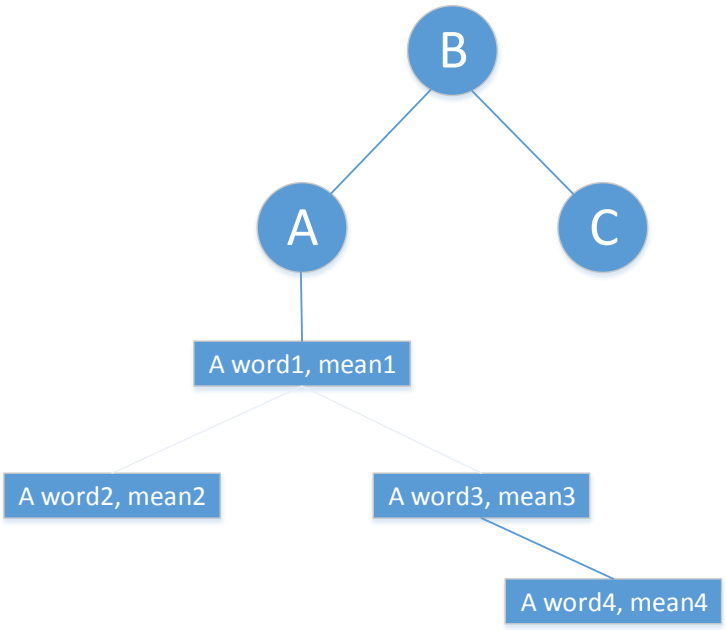
"Word.txt"에서 파일을 읽거나 이미 텍스트 파일에 있는 단어를'LOAD'하여 사용하는 TO_MEMORIZE 단어장이다. 앞으로 외워야 하는 단어장을 말한다. 이는 Queue를 이용하여 구현한다. Queue은 FIFO(First In First Out)으로, 제

일 먼저 들어간 것이 처음으로 나가는 형식이다. Push를 하면 head를 기준으로 마지막에 new node를 받고 연결 시킨다. Pop을 하면 head가 나와 처음에 넣은 것이 나오게 된다.



<MEMORIZING 단어장 – 'Binary Search Tree'>

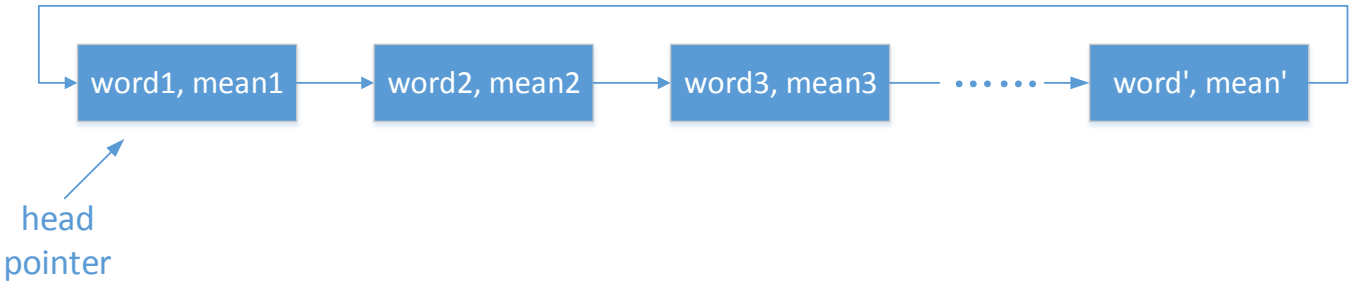
TO_MEMORIZE에서 'MOVE'를 받거나, 이미 텍스트 파일에 있는 단어를 'LOAD'하여 사용하는 현재 외우고 있는 단어장이다. 이는 BST를 이용하여 구현하는데, 노드가 알파벳으로 이루어진 알파벳 BST가 존재하고, 그 아래에 첫 글자에 해당 알파벳을 갖는 word들이 BST로 존재하는 방식이다. 처음 알파벳 BST를 구성할 때는 다음과 같은 순서로 넣어야 한다. (P, H, X, D, L, T, Z, B, F, J, N, R, V, Y, A, C, E, G, I, K, M, O, Q, S, U, W) 만약 A부터 넣는다고 하면, BST로 만들어져 그 뒤부터는 오른쪽 노드로만 Tree가 형성된다. 이를 방지하기 위함이다. Word BST는 일반적인 Insert를 행한다. 아래 그림은 구성한 단어장에서 알파벳 A를 가지는 Word BST와 알파벳 BST와의 연결을 보여주고 있다.



<MEMORIZED 단어장 –'Circular Linked List'>

MEMORIZING에서 'TEST'를 하거나, 텍스트 파일을 'LOAD'하여 사용하는 외운 단어장이다. MEMORIZED 단어장은 Circular Linked list를 이용하여 구현한다. 그냥 linked list와는 하나의 차이가 있는데, 제일 마지막 노드가 head를

가리켜야 한다. 따라서 new node가 들어올 때마다 head를 가리키는 노드는 바뀌어야 한다.



- 명령어(command)에 따른 동작

<LOAD>

TO_MEMORIZE(Queue)	MEMORIZING(BST)	MEMORIZED(Cll)
텍스트 파일을 우선적으로 읽어 내용이 있는지 확인한다. 없다면 ERROR처리 한다. 있다면 파일을 읽어 Queue의 방식대로 Push하여 단어장을 생성한다.	텍스트 파일을 우선적으로 읽어 내용이 있는지 확인한다. 없다면 ERROR처리 한다. 있다면 파일을 읽어 알맞은 알파벳 node를 찾고, 아래 word BST에 insert하는 방식으로 단어장을 생성한다.	텍스트 파일을 우선적으로 읽어 내용이 있는지 확인한다. 없다면 ERROR처리 한다. 있다면 파일을 읽어 Circular Linked List방식대로 insert하는 방식으로 MEMORIZED단어장을 생성한다.

<ADD>

TO_MEMORIZE(Queue)
"Word.txt"를 읽어온다. 텍스트 파일에 단어장 정보가 존재하지 않으면 오류처리 한다. 있다면 Queue에 맞게 단어 정보를 Push하여 TO_MEMORIZE에 모두 저장한다. 단어 개수 제한은 없다.

<MOVE / 숫자>

TO_MEMORIZE(Queue)	MEMORIZING(BST)
MOVE/숫자 명령어를 받으면 TO_MEMORIZE에서는 명령 받은 숫자(1~100)만큼 MOMORIZING(BST)에 넘겨준다. 이 때 Queue의 Pop함수를 이용한다. 하지만 일반적인 Queue와 다르게 Pop할 때 node를 delete하	MEMORIZNG(BST)에서는 큐에서 Pop으로 넘겨 받은 node를 Insert함수를 이용하여 BST에 연결한다. 이 때 word

지 않고, return node로 BST에게 node만을 넘겨주는 방식이다.	BST가 가지는 총 Word node수는 100 개가 되지 않도록 한다.

<SAVE>

TO_MEMORIZE(Queue)	MEMORIZING(BST)	MEMORIZED(Cll)
세 단어장중 하나라도 비어 있으면 우선 ERROR처리 한다. 비어 있는 경우는 Queue와 Circular Linked List에서는 head pointer를 이용하여 판단하고, BST에서는 Word_cnt가 0인지 확인하여 판단한다.		
Queue의 Save()함수를 이용하여 텍스트 파일에 저장한다.	Word BST의 Save()함수를 이용하여 텍스트 파일에 저장한다. 이때 저장 순서는 pre-order를 사용한다.	Circular Linked List의 Save()함수를 이용하여 텍스트 파일에 저장한다.

<TEST / word / mean>

MEMORIZING(BST)	MEMORIZED(Cll)
입력 받은 단어를 Word BST에서 Search 함수를 이용하여 찾고, 이를 빼내어 MEMORIZED로 보낸다. 빼낼 때에는 BST에서 Delete함수와 동일하지만, delete를 하지 않고 노드를 넘겨준다. Delete 함수의 구현은 아래와 같다. -삭제 노드 degree 0일 때: 연결고리가 없으므로 그냥 pop하면 된다. -삭제 노드 degree 1일 때: 찾은 노드가 가지고 있는 자식 노드가 pop하는 노드(찾은 노드)를 대체하도록 연결시켜준 후에 pop한다. -삭제 노드 degree 2일 때: 찾은 노드의 왼쪽 자식 노드 중 가장 큰 노드가 pop하는 노드를 대체하도록 연결시켜준 후에 pop한다.	MEMORIZED(Circular Linked List)에서는 BST에서 Delete로 넘겨 받은 node를 Insert함수를 이용하여 Circular linked list에 연결한다.

<SEARCH / word>

각 Class들의 Search함수를 이용하여 단어를 찾고, 존재하는 경우 출력한다. 존재하지 않으면 Error처리 한다.

<PRINT / wordbook / order>

TO_MEMORIZE(Queue)	MEMORIZING(BST)	MEMORIZED(Cll)
Queue 클래스의 print함수를 이용하여 결과를 출력/저장한다.	MEMORIZING의 PRINT는 종류가 다양하다. 또한 PRINT를 하기 위해서는 Alphabet BST를 거쳐 Word BST로 들어가 출력해야 하는 점의 유의한다. 종류는 아래와 같다. -Inorder: 중위 순회는 NULL이 아닐 때까지 왼쪽으로 우선 이동한다. 끝에 도달했을 때, 그 노드를 방문하고 오른쪽 자식 노드로 간다. 없다면 뒤로 돌아간다. -Preorder: 전위 순회는 중위 순회와 비슷하게 NULL이 되기 전까지 왼쪽으로 간다. 하지만 왼쪽으로 가면서 노드를 방문하게 된다. 따라서 root부터 시작하여 왼쪽으로 NULL이 되기 전까지 방문을 계속한다. NULL이 되면, 오른쪽으로 이동하여 순회를 계속한다. -postorder: 후위 순회 또한 NULL이 되기 전까지 왼쪽으로 간다. 맨 끝까지 가면 그 노드를 방문하고, 그 부모 노드의 또 다른 자식 노드, 오른쪽 노드를 우선 방문한다. 그 후에 부모 노드를 방문하고, 그와 상응하는 오른쪽 노드를 방문한다. 자식을 우선 방문한다고 생각하면 된다. -levelorder : 말 그대로 레벨 순서에 맞게 방문하는 순회를 말한다. 따라서 각 레벨에 해당하는 노드를 차례대로 방문한다. 레벨 오더에서도 많은 방법이 있지만 이번 프로젝트에서는 위부터 level순으로, 왼쪽에서부터 오른쪽 순으로 출력한다.	Circular Linked List 클래스의 print함수를 이용하여 결과를 출력/저장한다.

<UPDATE / word / mean>

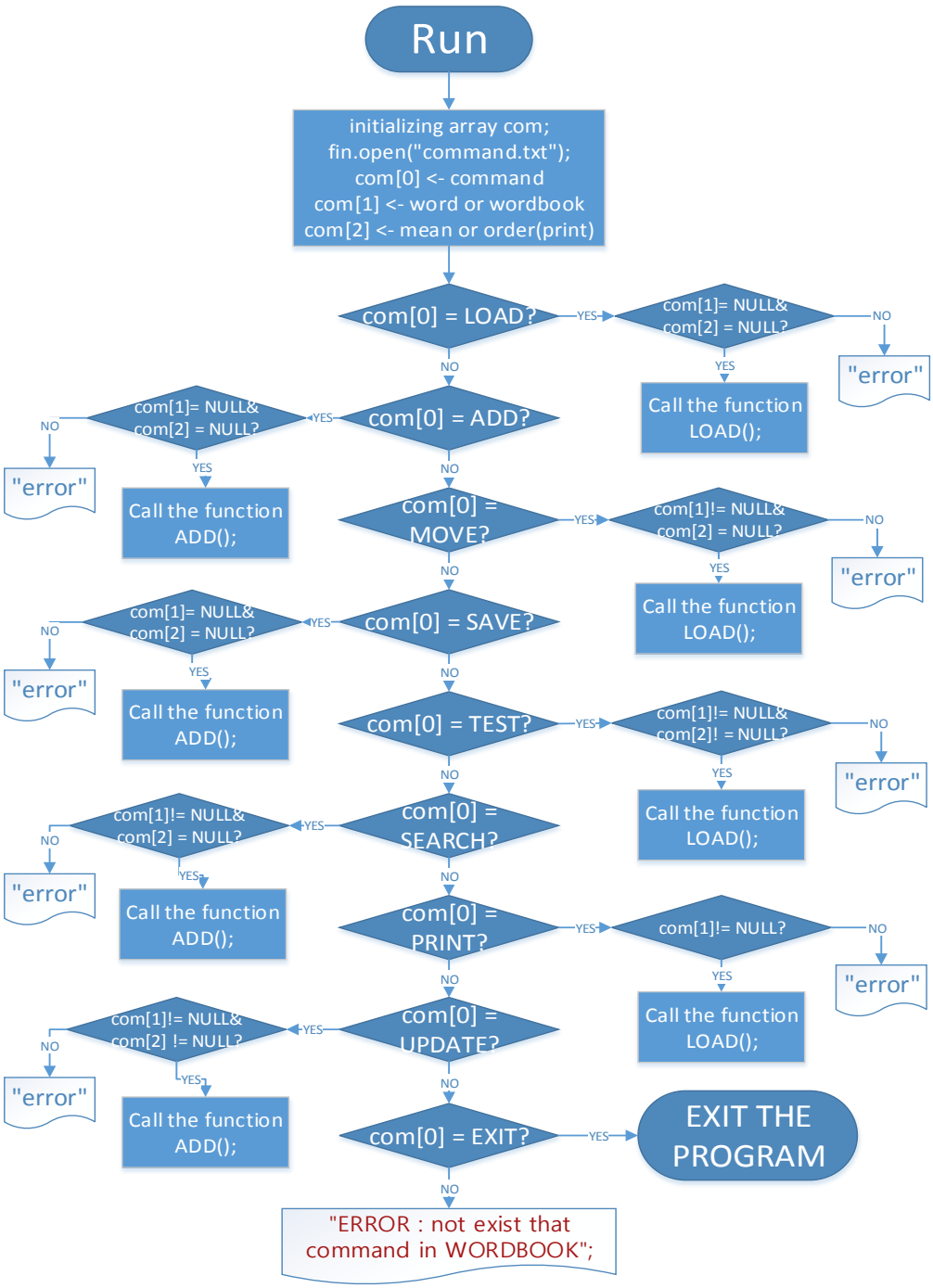
각 class들의 Search함수를 이용하여 단어를 찾고, 존재한다면 Set함수를 사용하여 단어의 mean를 변경 시켜준 후 출력한다. 만약 단어나 단어장 정보가 없을 경우 Error처리 한다.

<EXIT>

-프로그램을 종료한다.

3. Flow chart

<manager flow chart>



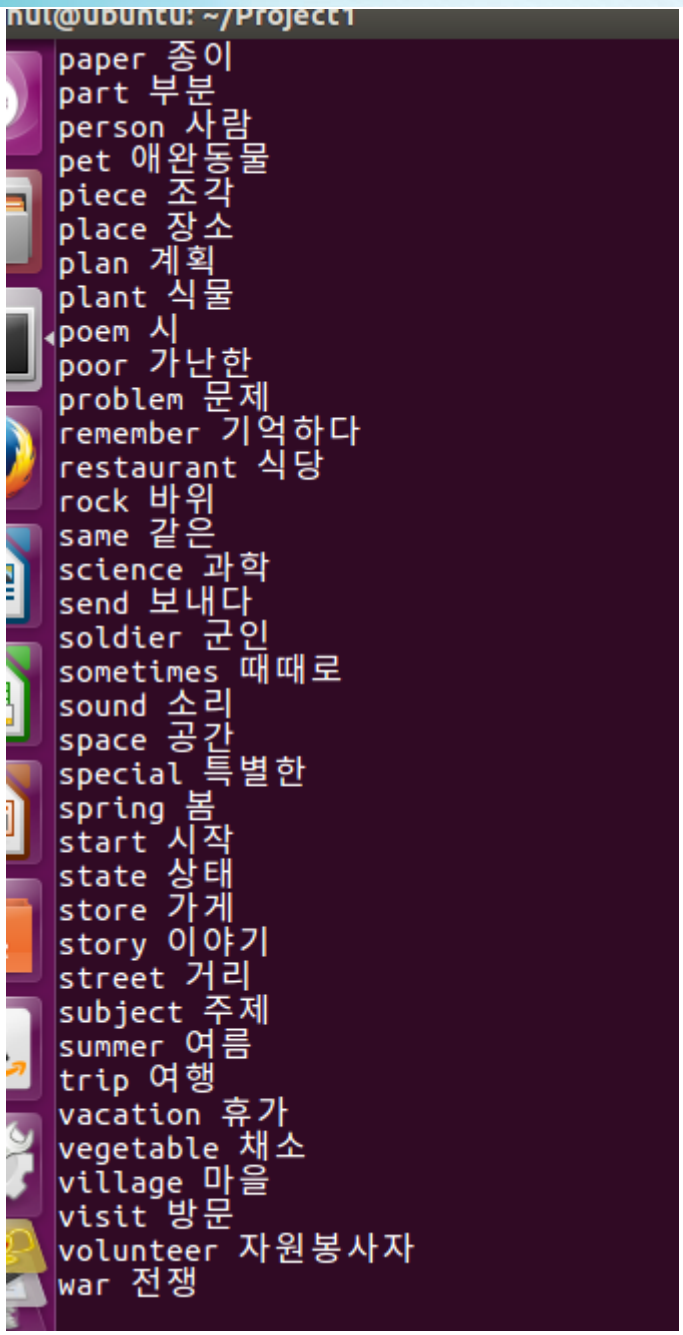
4. Result

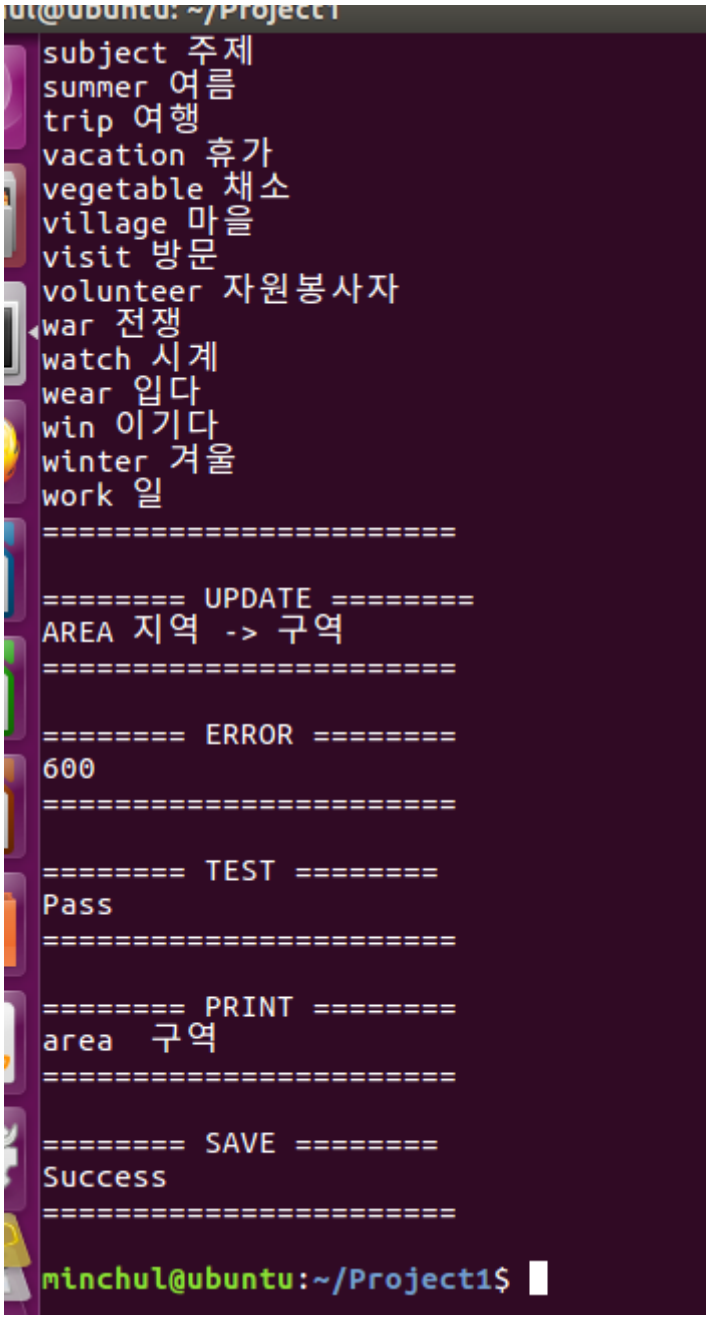
TEST1

```
ADD
MOVE 100
PRINT MEMORIZED
PRINT MEMORIZING R_IN
UPDATE AREA 구역
SEARCH AREA
TEST AREA 구역
PRINT MEMORIZED
SAVE
EXIT
```

```
minchul@ubuntu:~/Project1$ ./wordbook
======== ADD ========
Success
======================

======== MOVE ========
Success
======================

======== ERROR ========
700
======================

======== PRINT ========
activity 활동
advertisement 광고
always 언제나
area 지역
art 미술
autumn 가을
back 뒤
bird 새
bottle 병
clean 깨끗한
clothes 옷
cook 요리사
course 과정
course 강좌
culture 문화
different 다른
each 각각
early 일찍
earth 지구
easy 쉬운
end 끝
enjoy 즐기다
example 예
```

```
example 예
face 얼굴
famous 유명한
fast 빨리
fire 불
fly 날다
forest 숲
fun 재미
gene 유전자
group 집단
habit 습관
hand 손
history 역사
hot 뜨거운
important 중요한
information 정보
island 섬
job 직업
just 단지
leaf 잎
learn 배우다
letter 편지
life 인생
light 빛
line 선
listen 듣다
lot 많이
make 만들다
mind 마음
movie 영화
name 이름
nature 자연
neighbor 이웃
newspaper 신문
office 사무실
often 자주
paper 종이
```


paper 종이
part 부분
person 사람
pet 애완동물
piece 조각
place 장소
plan 계획
plant 식물
poem 시
poor 가난한
problem 문제
remember 기억하다
restaurant 식당
rock 바위
same 같은
science 과학
send 보내다
soldier 군인
sometimes 때때로
sound 소리
space 공간
special 특별한
spring 봄
start 시작
state 상태
store 가게
story 이야기
street 거리
subject 주제
summer 여름
trip 여행
vacation 휴가
vegetable 채소
village 마을
visit 방문
volunteer 자원봉사자
war 전쟁

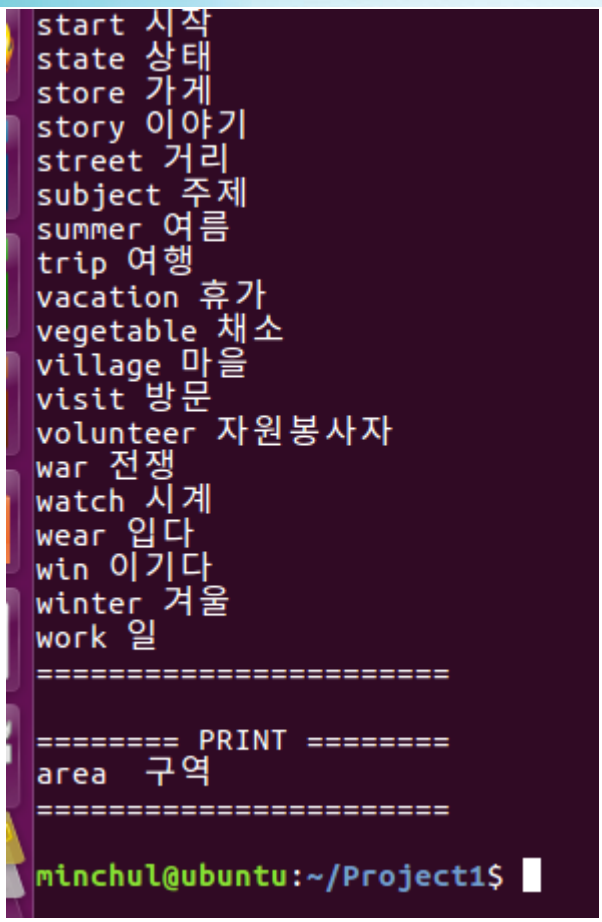

subject 주제
summer 여름
trip 여행
vacation 휴가
vegetable 채소
village 마을
visit 방문
volunteer 자원봉사자
war 전쟁
watch 시계
wear 입다
win 이기다
winter 겨울
work 일
========================
======== UPDATE ========
AREA 지역 -> 구역
========================
======== ERROR ========
600
========================
======== TEST ========
Pass
========================
======== PRINT ========
area 구역
========================
======== SAVE ========
Success
========================
minchul@ubuntu:~/Project1$

TEST2

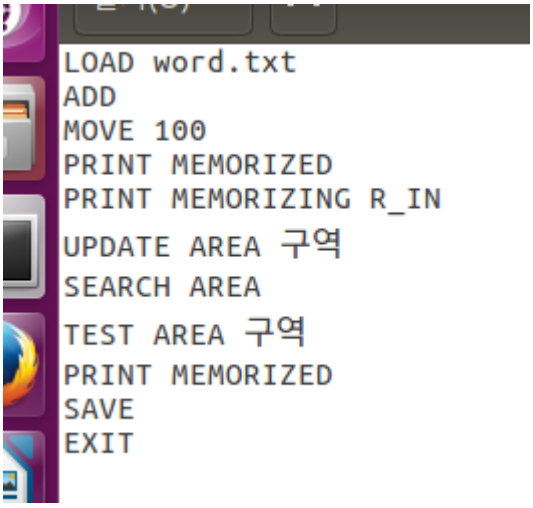
```
LOAD
PRINT TO_MEMORIZE
PRINT MEMORIZING R_IN
PRINT MEMORIZED
EXIT
```

```
minchul@ubuntu:~/Project1$ ./wordbook
======== LOAD ========
Success
======================

======== PRINT ========
machine 기계
fact 사실
rule 규칙
die 죽다
busy 바쁜
sick 아픈
health 건강
holiday 공휴일
goal 목표
order 순서
result 결과
half 절반의
decide 결정하다
choose 선택하다
difficult 어려운
foreign 외국의
brain 뇌
voice 목소리
opinion 의견
age 나이
price 가격
dish 접시
subway 지하철
bear 곰
human 인간
buy 사다
sell 팔다
follow 따르다
miss 놓치다
```

```
pole 막대기
rubber 고무
root 뿌리
structure 구조
vote 투표
patient 환자
quick 빠른
captain 선장
bucket 양동이
cage 새장
kite 연
miracle 기적
======================

======== PRINT ========
activity 활동
advertisement 광고
always 언제나
art 미술
autumn 가을
back 뒤
bird 새
bottle 병
clean 깨끗한
clothes 옷
cook 요리사
course 과정
course 강좌
culture 문화
different 다른
each 각각
early 일찍
earth 지구
easy 쉬운
end 끝
enjoy 즐기다
```


start 시작
state 상태
store 가게
story 이야기
street 거리
subject 주제
summer 여름
trip 여행
vacation 휴가
vegetable 채소
village 마을
visit 방문
volunteer 자원봉사자
war 전쟁
watch 시계
wear 입다
win 이기다
winter 겨울
work 일
========== PRINT ========
area 구역
==========
minchul@ubuntu:~/Project1$

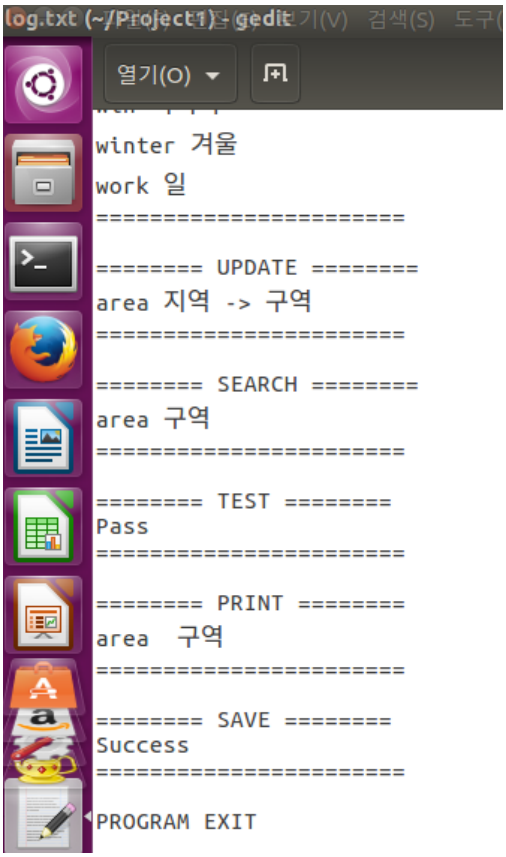
TEST3

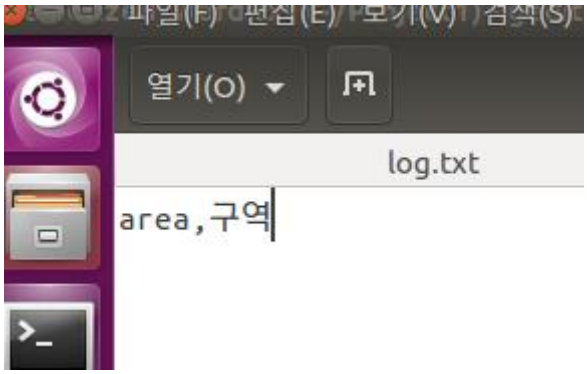

LOAD word.txt
ADD
MOVE 100
PRINT MEMORIZED
PRINT MEMORIZING R_IN
UPDATE AREA 구역
SEARCH AREA
TEST AREA 구역
PRINT MEMORIZED
SAVE
EXIT

```
minchul@ubuntu:~/Project1$ ./wordbook
======== ERROR ========
100
=======================

======== ADD ========
Success
=======================

======== MOVE ========
Success
=======================

======== ERROR ========
700
=======================

======== PRINT ========
activity 활동
advertisement 광고
always 언제나
area 지역
art 미술
autumn 가을
back 뒤
bird 새
bottle 병
clean 깨끗한
clothes 옷
cook 요리사
course 과정
```

```
visit 방문
volunteer 자원봉사자
war 전쟁
watch 시계
wear 입다
win 이기다
winter 겨울
work 일
=======================

======== UPDATE ========
area 지역 -> 구역
=======================

======== SEARCH ========
area 구역
=======================

======== TEST ========
Pass
=======================

======== PRINT ========
area  구역
=======================

======== SAVE ========
Success
=======================

minchul@ubuntu:~/Project1$
```

LOG.TXT, MEMORIZED.TXT





5. Consideration

이름	프로젝트에서 맡은 역할	본인 스스로 생각하는 자신의 점수
김민철	Word Node, Word BST, Circular Linked List, Manager (Load, Save, Add, Search, Update, Test, Print - I_POST, Circular Linked List) -같이 수정한 부분이 많아서 나누기가 힘듦	10
고찰	나는 이번 프로젝트에서 처음으로 프로그래밍을 조별과제로 진행하게 되었는데, 처음에는 '사람마다 코드를 작성하는 방식이 다른데, 어떻게 조별과제로 진행할까'하고 생각하며 걱정도 되었고, 나눠서 코드를 작성하고 난 다음에 하나의 프로젝트로 합치는 경우에 발생하는 문제들은 어떻게 해결해야할지 걱정이 되었다. 솔직히 엄청나게 어려운 문제는 아니여서 머리를 맞대고 어떻게 해결할까 하는 문제는 별로 없어서, 만나더라도 설계를 하고 회의를 한다기 보다는, 각자 맡은 영역 코드를 다 작성했는지 확인하면서 상대방의 코드를 읽어보고 물어보고 이해하는 식으로 진행하였다. 함수의 이름, 변수의 이름 등등 모든것들이 정해져있어서 사실 하나로 합치는데 큰 어려움도 없었고, Delete함수와 None Recursive Postorder함수를 작성할때 가장 어려웠고, Windows 운영체제 기반의 visual studio에서 작성한 코드를 리눅스 환경에서 실행하는 것이 최고로 어려웠다. 처음 접한 리눅스인지라, 컴파일하는 방법이나 한글이 깨지는 문제, 파일을 실행하는 문제 등 여러가지 문제들이 많았다. visual studio 에서는 작동되는 코드들이 리눅스에서는 되지 않는 경우도 많았고, 디버깅하는 방법도 어려워서 문제가 하나 발생하면 어디를 수정해야하는지 찾는 것조차 어려웠다. 처음에는 리눅스에 한글어를 설치하는 것조차 어려워했던 내가 어느새 리눅스에 조금 적응하여 자연스럽게 문제를 찾고 코드를 수정하고, 컴파일을 다시하고, 프로그램을 확인하는 것을 보면서 굉장히 뿌듯했고, 좋은 조원을 만나 과제를 진행하는데 재밌기도했다. 처음에는 조별로 진행하지말고 개인별로 이 과제를 진행했으면 했지만 조별과제도 조별과제만의 장점이 있다는 것을 배웠다.	

이름	프로젝트에서 맡은 역할	본인 스스로 생각하는 자신의 점수
고은나래	Alphabet Node, Alphabet BST, Queue, Stack, Manger (Print – Queue, R_IN, I_IN, R_PRE, I_PRE, R_POST, I_LEVEL)	10
고찰	이번 프로젝트 에서는 1학기 때 배웠던 큐와 스택, Linked list와 DS에서 배운 BST를 활용하여 단어장을 만들었다. 기존에 알았던 큐와 Linked List는 쉬웠지만, Binary Search Tree중에서 PRINT를 중점으로 코딩하였는데, 재귀함수를 사용하지 않고 짜는 Post Order가 어려웠다. -프로젝트가 visual로 정상적으로 돌아간다고 해도 리눅스 환경에서 정상적으로 작동하는지 보는 것 또한 중요했다. 실습 시간에 리눅스를 설치는 했지만, 여기서 c++코드를 어떻게 돌려야 하는지 처음에는 막막했다. 리눅스를 잘하는 사람에게 물어보면서 어떻게 컴파일 해야하는지, 에러가 어디서 뜨는지 배웠고, 오류를 찾아 제대로 고쳐 구현하였다. -visual에서는 예전과 다르게 신기한 오류를 많이 봤다. 첫 오류는 함수를 사용하는데 메모리 접근에 문제가 생겨 다른 함수로 바꿔주었다. 또 보안 문제가 생기는 함수가 있어 그 함수를 사용하지 않고 내가 원하는 코드를 구현해야 했다. linux에서는 비교적 간단한 오류였는데, 처음 우리가 word파일을 받았을 때 visual이 읽을 수 없는 문자로 띄어쓰기가 되어있었다. 이를 visual이 받을 수 있도록 바꿔준 후에 코드를 완성하고 리눅스에 올리면서 나는 오류를 고쳐야 했다. 이는 띄어쓰기를 '₩r'로 바꾸고, word파일을 원본으로 바꿔 해결했다. 또 하나의 오류는 함수 문제였는데, visual에서만 제대로 동작하는 함수였다. 그래서 linux에서 작동하는 함수로 고쳐 오류를 잡았다. -코드를 짜는 과제나 프로젝트를 조별로 한 것은 처음이었다. 내 코드만 잘 구현하는 것에 끝나지 않고 내가 아닌 다른 사람 코드를 보고 주석 없이 이해해야 했다. 처음에는 간단한 코드인데도 각자 방식이 달라 이해하기 어려웠다. 각자 짠 코드를 합쳐 manager를 만들거나 오류를 잡아가면서 코드를 이해했다. 코드를 합칠 때에는 서로 인자나 함수 call이 달라 난항을 겪었다. 코드를 합치고 바로 완성된 것이 아니였기 때문에 여러 번 수정을 거치면서 바뀌는 부분이 어딘지 거의 외울 정도였다.내 코드와 다른 사람의 코드를 합쳐 하나의 프로	

	젝트를 구현한다는 것이 어렵기는 했지만, 나와는 다른 방식으로 짠 코드를 보면서 더 넓게 코딩할 수 있게 되었다.