

Data Structure

Project 1

Professor	심동규, 이기훈
Department	컴퓨터공학과
Name(Student ID)	진세진(2012722025) 이선희(2014722060) 한승주(2015722084)
Date	2016-10-07



1. Introduction

Queue, Binary Search Tree(이하 BST), Circular Linked List를 이용하여 영어 단어장 프로그램을 만든다. 단어들이 들어있는 word.txt라는 텍스트 파일이 존재하고 여기서 0부터100사이의 개수만큼 단어를 Queue를 이용하여 빼내어 외워야 할 단어장인 To-memorize로 이동한다. 다음으로 BST를 이용하여 현재 외우고 있는 단어장인 Memorizing에 넣고 여기서 Test하여 외운 단어를 Circular linked list를 이용하여 Memorized를 구현한다.

To-memorize는 Queue를 이용하여 word.txt라는 텍스트 파일에서 단어를 가져와 Push함수를 이용하여 Queue에 insert하고 insert된 단어들을 Memorizing으로 pop함수를 통해 옮겨진다. 단어가 To-memorize에서 Memorizing으로 옮겨질 때는 명령어 move를 통해서 이동하고 Memorizing에는 최대 100개의 단어만 들어가고 100개이상의 단어가 들어가게 될 경우 설정된 에러코드를 보이도록 한다.

Memorizing은 BST를 이용하여 구현하는데 먼저 BST를 이용하여 알파벳BST를 생성하고 각각의 알파벳 노드 내부에 각 알파벳으로 시작하는 단어 BST가 따로 존재한다. 처음의 알파벳 BST의 경우 프로그램이 시작할 때 바로 구축이 되어야하고 알파벳 노드들 내부에 저장되어 있는 단어들이 A부터 Z까지 모두 합해서 최대 100개의 단어들이 저장될 수 있다. Test명령어를 통해서 Memorizing안의 단어를 Test하고 통과 된 단어는 Move 명령어를 통해 Memorized로 이동한다.

Memorized는 Circular linked list를 이용하여 구현하는데 가장 처음 들어오는 단어를 Head pointer로 지정하고 이후에 들어오는 단어들은 Head pointer의 Next가 아닌 뒤에 연결된다.

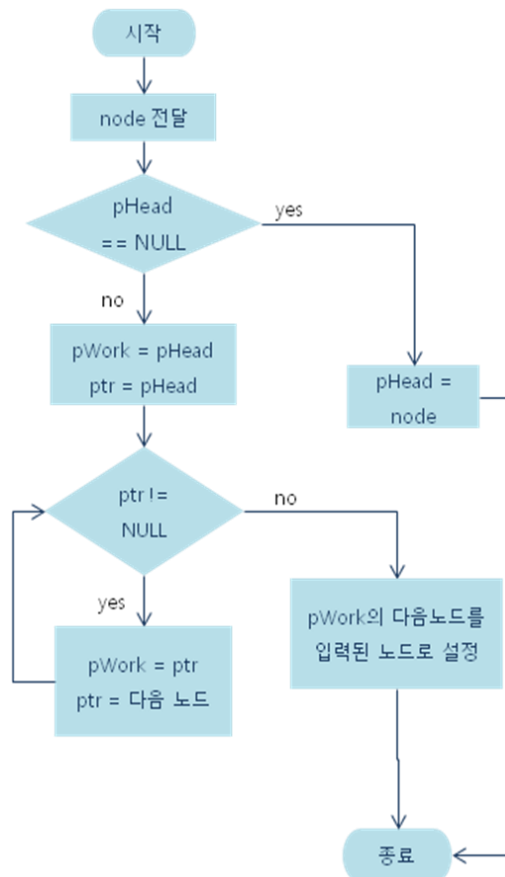
지정된 명령어를 command.txt라는 텍스트 파일에 넣어 차례대로 읽어 처리한다. Test, Update, Search명령어를 실행할 때 대소문자를 구별하지 않고 실행시키고 중간에 에러 발생시에는 에러코드를 출력하도록 한다. 이때 에러코드는 log.txt에 기록한다. 프로그램을 실행시킨 과정을 log.txt에 저장시켜 success되었는지 error가 발생했는지를 기록한다.

명령어에는 Load, Add, Move, Save, Test, Search, Print, Update, Exit가 있다. Load는 세 가지 단어장의 텍스트파일에서 이전에 작성하던 연결리스트를 불러오는 명령어로 각 단어장의 텍스트파일이 존재하지 않을 경우 에러 코드 100을 출력시킨다. Add는 word.txt에서 단어를 가져와 To-memorize에 단어들을 insert하는 것으로 단어 파일이 존재하지않거나 더 이상 단어가 없을 경우 에러 코드 200을 출력한다. Move는 단어들을 이동시키는 명령어로 To-memorize에서 Memorizing으로 Memorizing에서 Memorized로 단어를 이동시킬 때 사용한다. 사용자가 원하는 숫자만큼의 단어가 존재하지 않거나 Memorizing에 단어가 100개 이상이 들어가게 될 경우 에러 코드 300을 출력한다. Save는 각 리스트에 연결된 단어들을 각각의 텍스트 파일에 저장시키는 명령어로 마찬가지로 텍스트 파일이 존재하지 않을 경우 에러코드 400을 출력한다. Test는 단어를 외웠는지 확인하는 명령어로 Memorizing에서 이 명령어를 실행하여 단어가 외운 것이 확인되면 Move명령어를 이용하여 Memorized로 단어를 이동시킨다. 만약 단어가 Memorizing에 없거나 뜻이 틀릴 경우 에러코드 500을 출력한다. Search는 단어의 뜻을 검색하는 명령어로 검색하면 영어 단어와 한글 뜻을 보여주고 마찬가지로 입력한 단어가 없다면 에러 코드를 출력하는데 이 때 에러 코드는 600이다. Print는 각 단어장에 있는 단어들을 출력시켜주는데 여러가지 Tree Traversal을 이용하여 출력시켜준다. 단어장이 존재하지 않으면 에러코드 700을 출력시킨다. Update는 단어의 뜻을 변경시켜주는 명령어이다. 만약 단어가 존재하지 않거나 단어장 정보가 존재하지 않으면 800이라는 에러코드를 출력시킨다. Exit은 말 그대로 프로그램을 종료시키는 명령어로 이 때 모든 메모리를 해제시킨다.

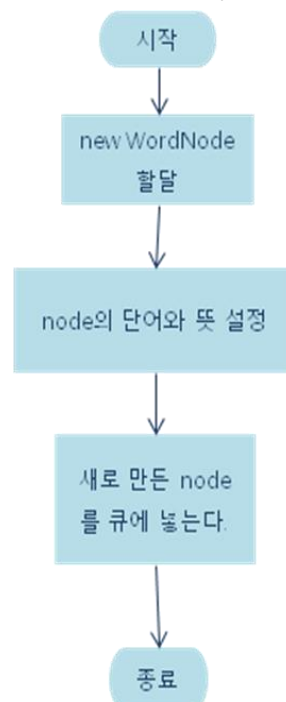
2. Flowchart

1) Queue

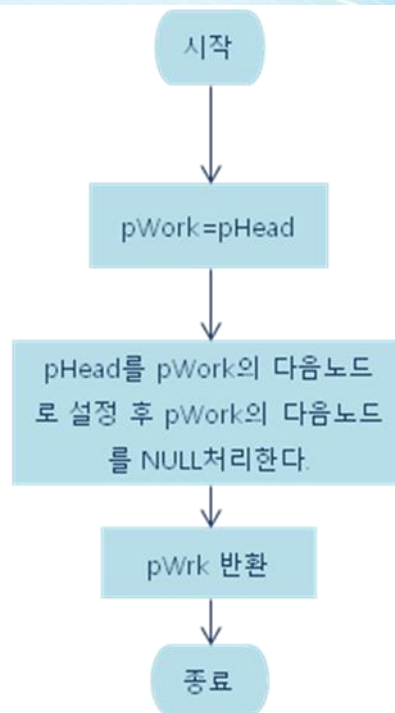
a) void Queue::Push(WordNode*node)



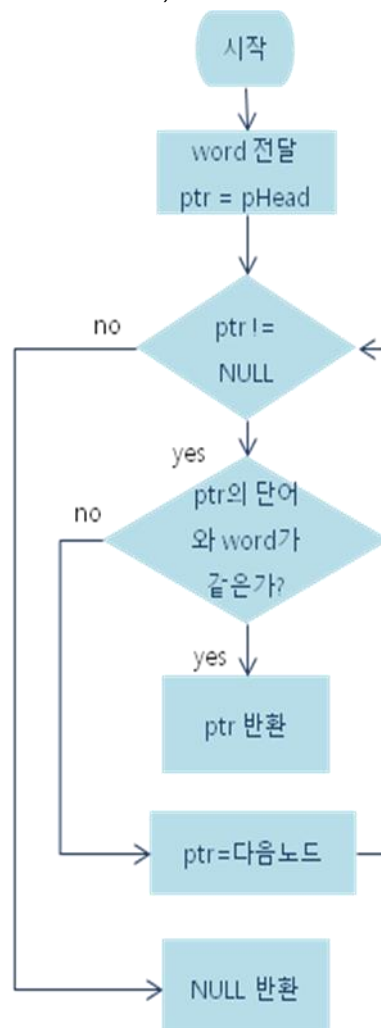
b) void Queue::Push(const char*word,const char*mean)



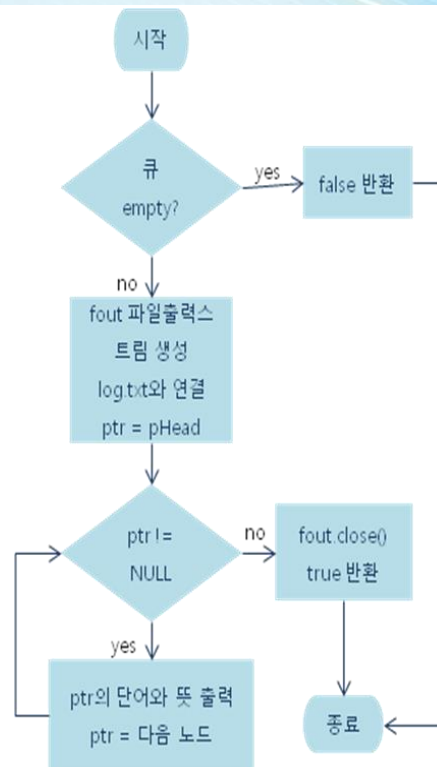
c) WordNode*Queue::Pop()



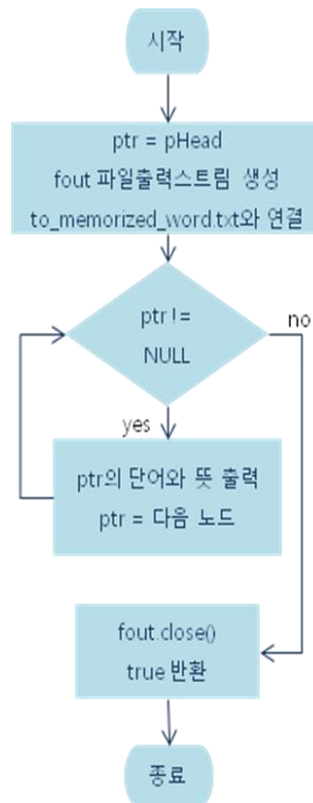
d) WordNode*Queue::Search(const char*word)



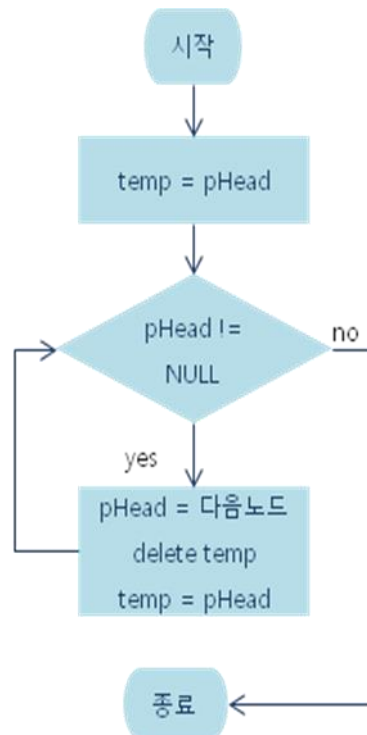
e) bool Queue::Print()



f) bool Queue::Save()

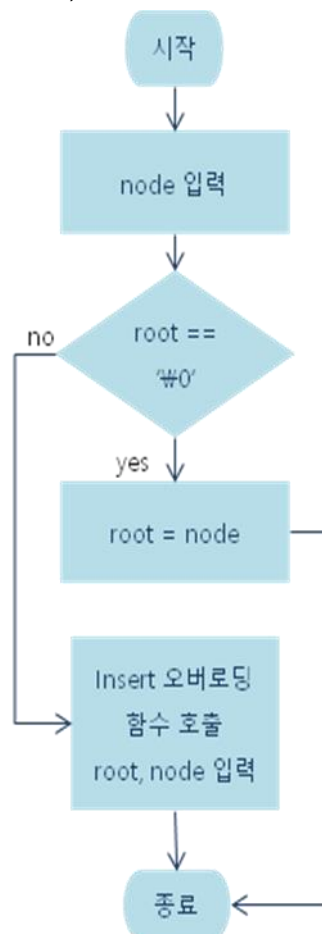


g) Queue::~~Queue()

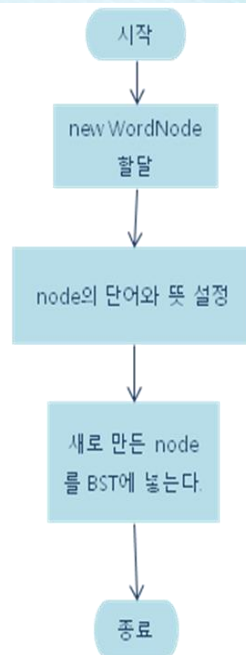


2) WordBST

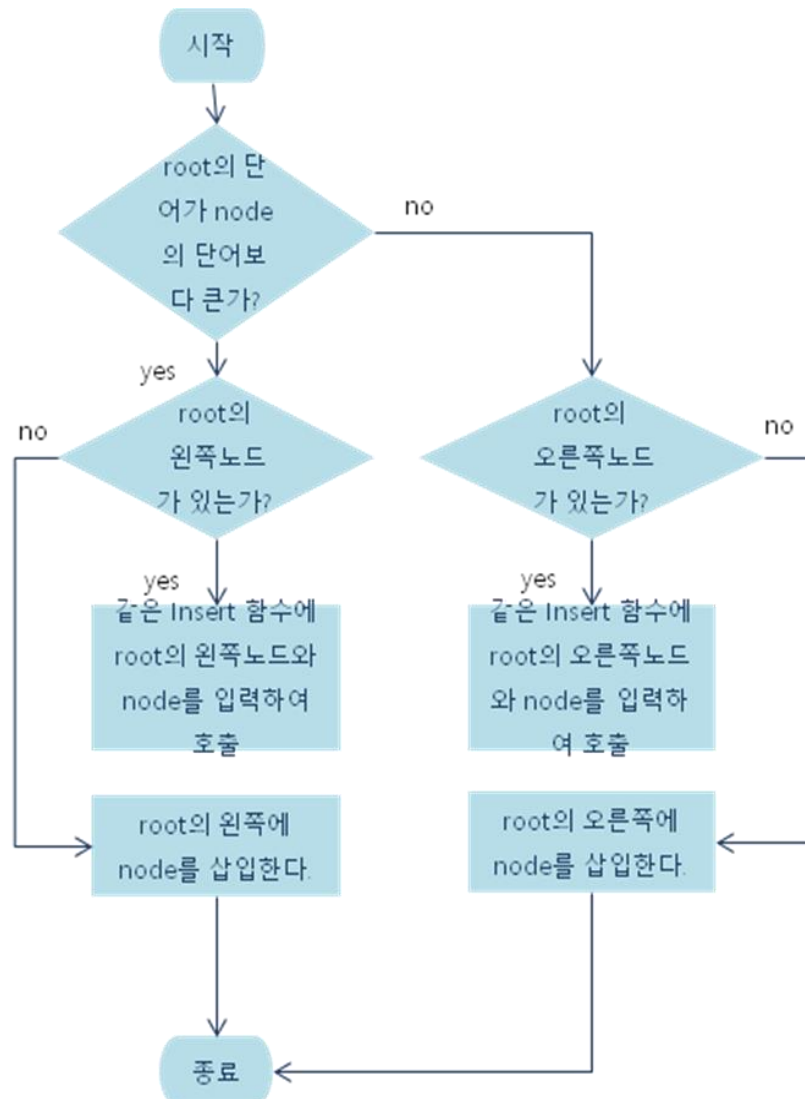
a) void WordBST::Insert(WordNode* node)



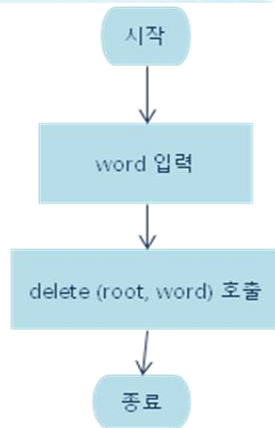
b) void WordBST::Insert(const char* word, const char* mean)



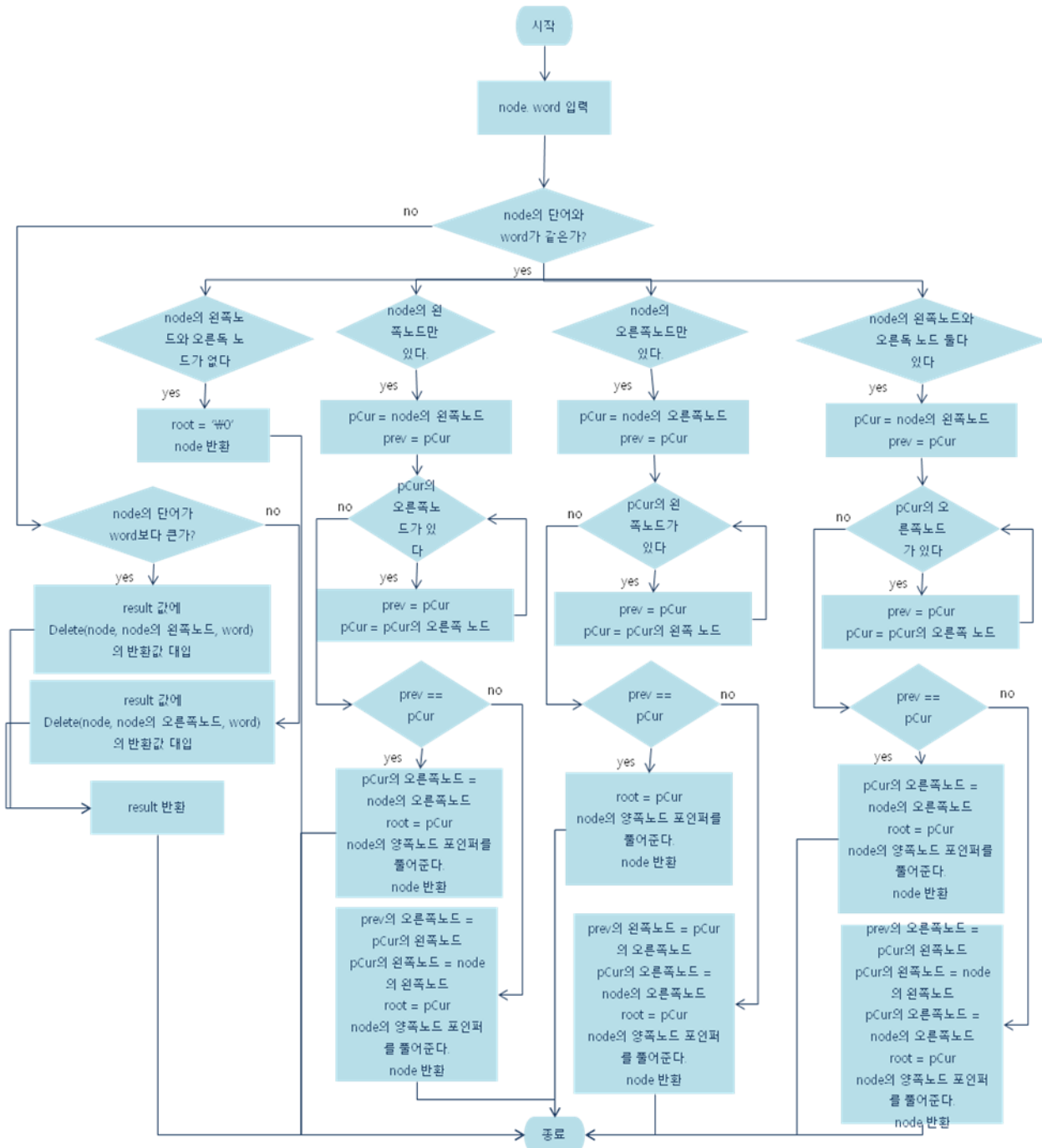
c) void WordBST::Insert(WordNode* root, WordNode* node)



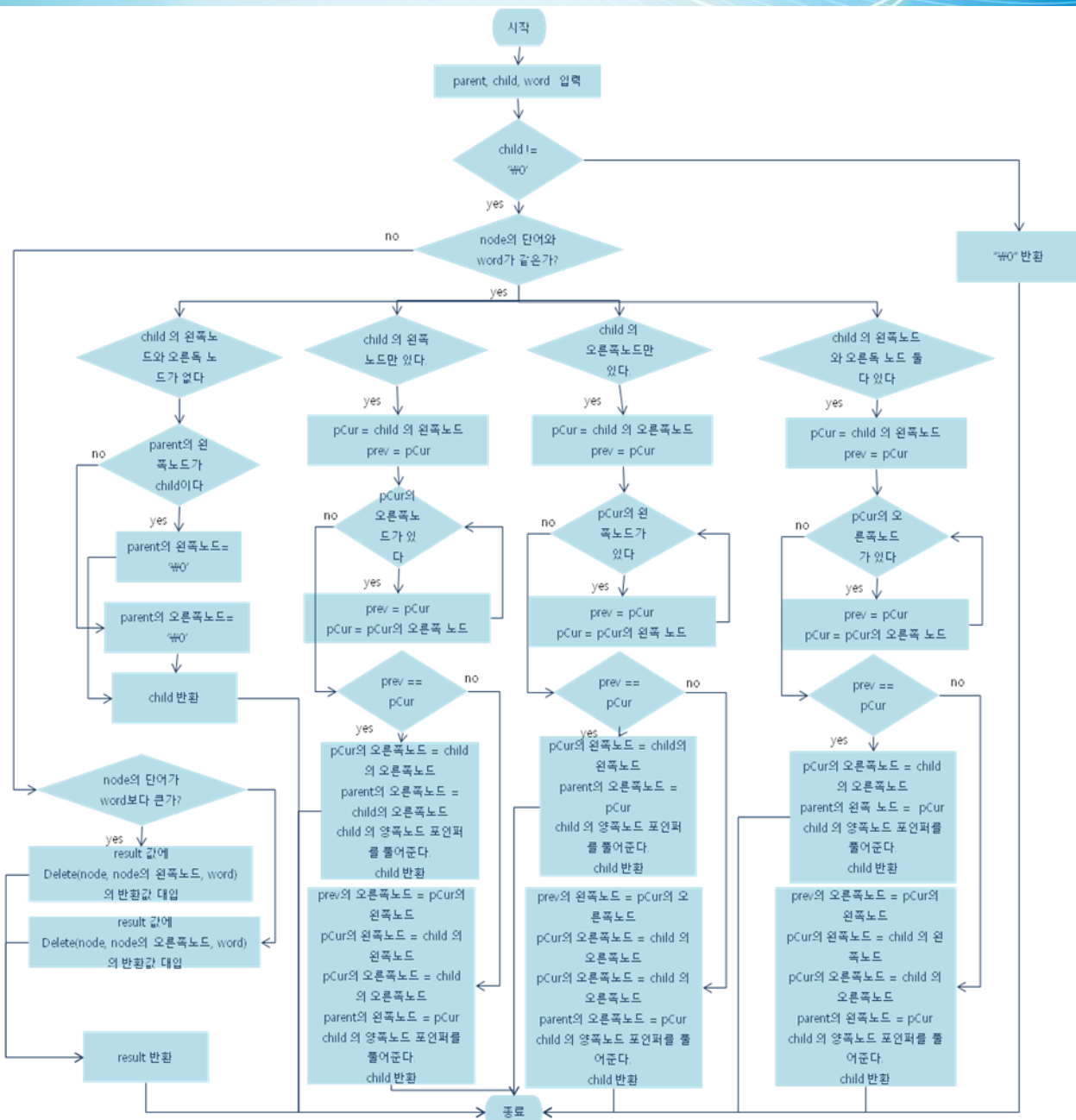
d) WordBST::Delete(const char* word)



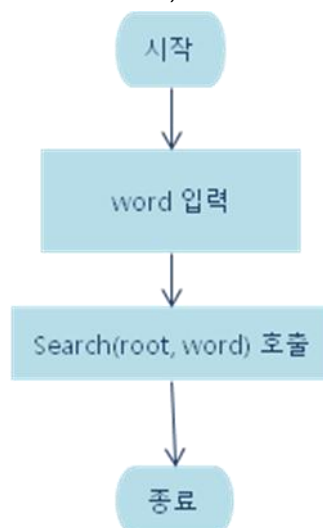
e) WordNode* WordBST::Delete(WordNode* node, const char* word))



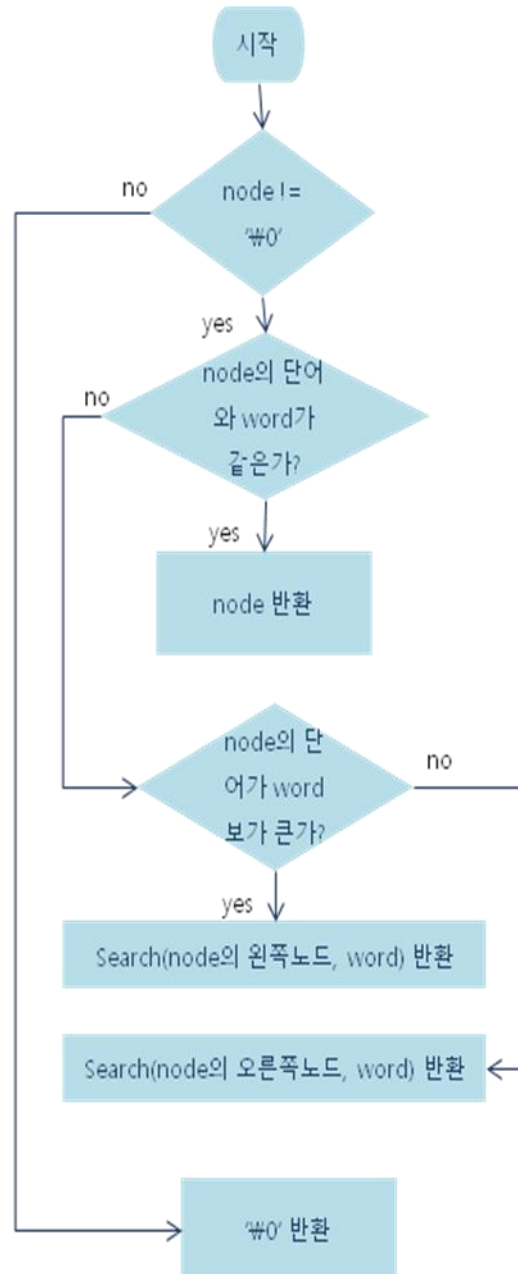
f) WordNode* WordBST::Delete(WordNode* parent, WordNode* child, const char* word)



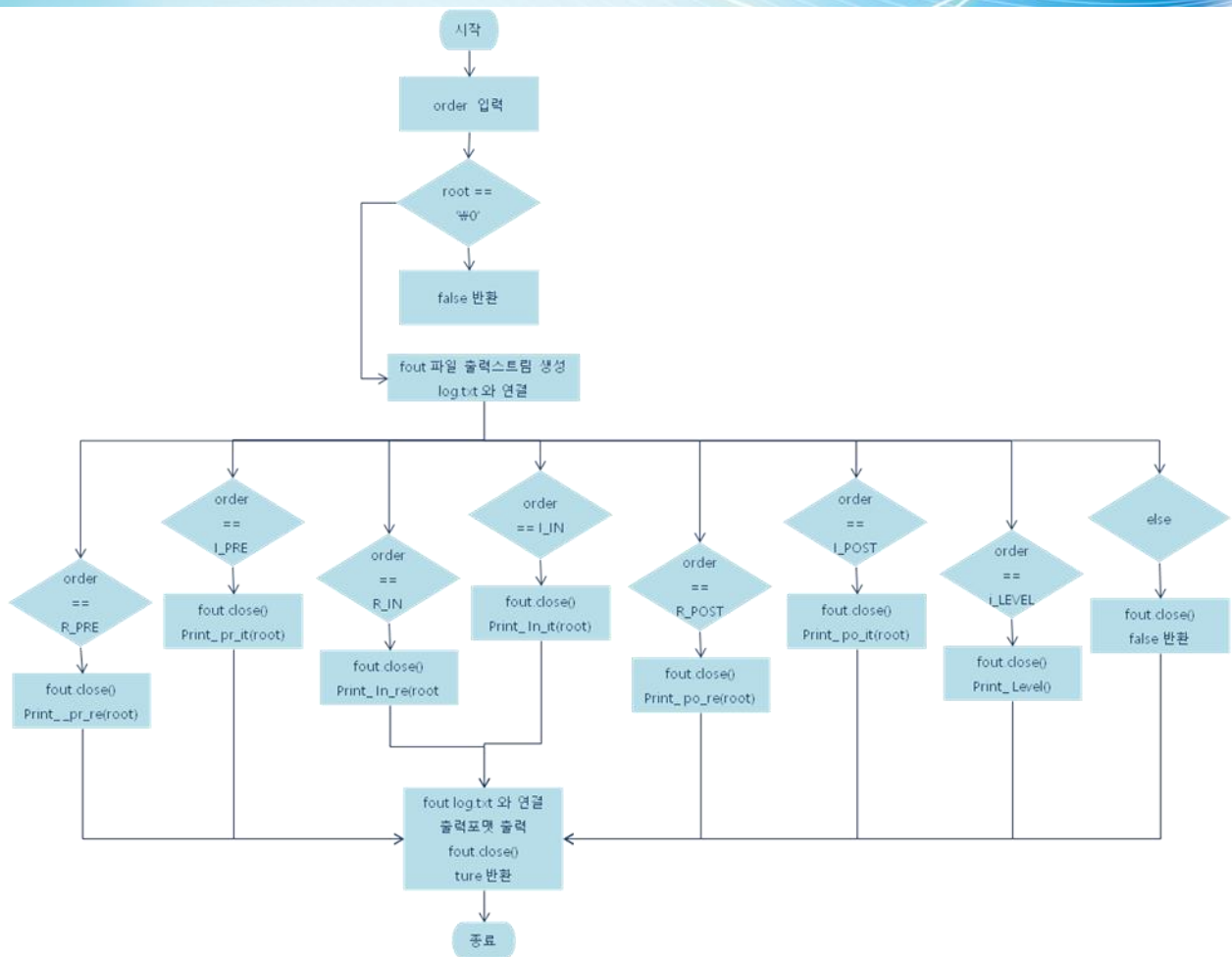
g) WordNode* WordBST::Search(const char* word)



h) WordNode* WordBST::Search(WordNode* node, const char* word)

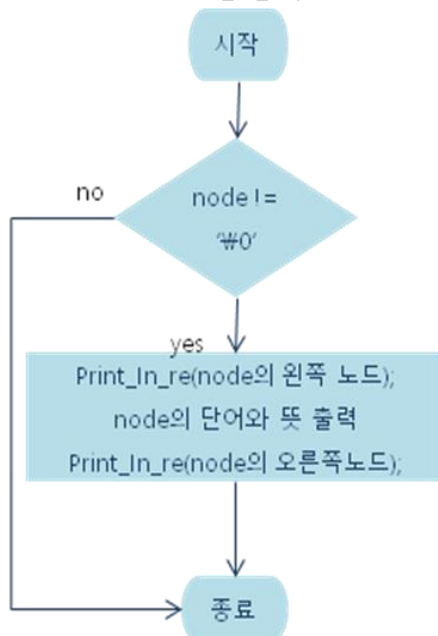


i) bool WordBST::Print(const char* order)

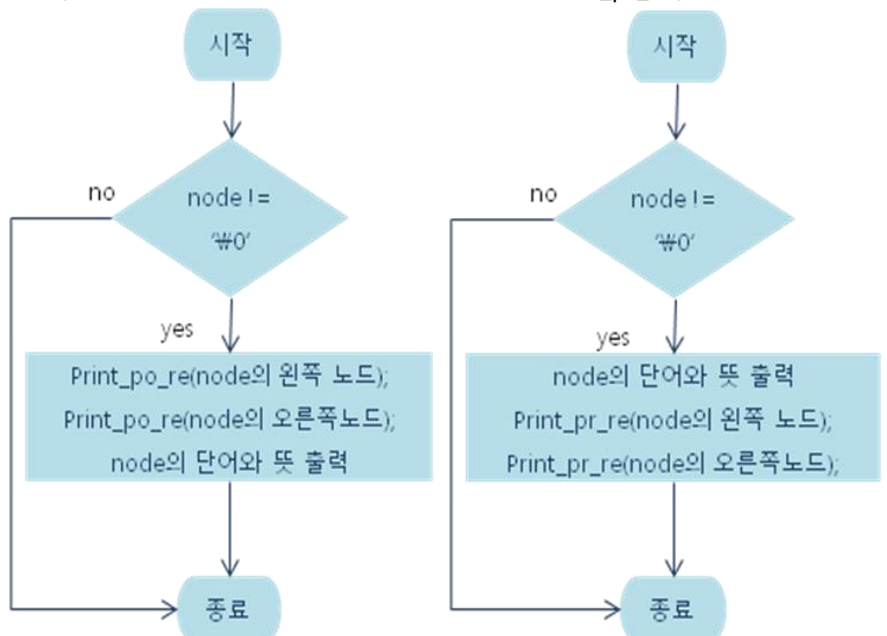


j) 재귀함수를 사용한 PRINT

void WordBST::Print_In_re(WordNode* node)

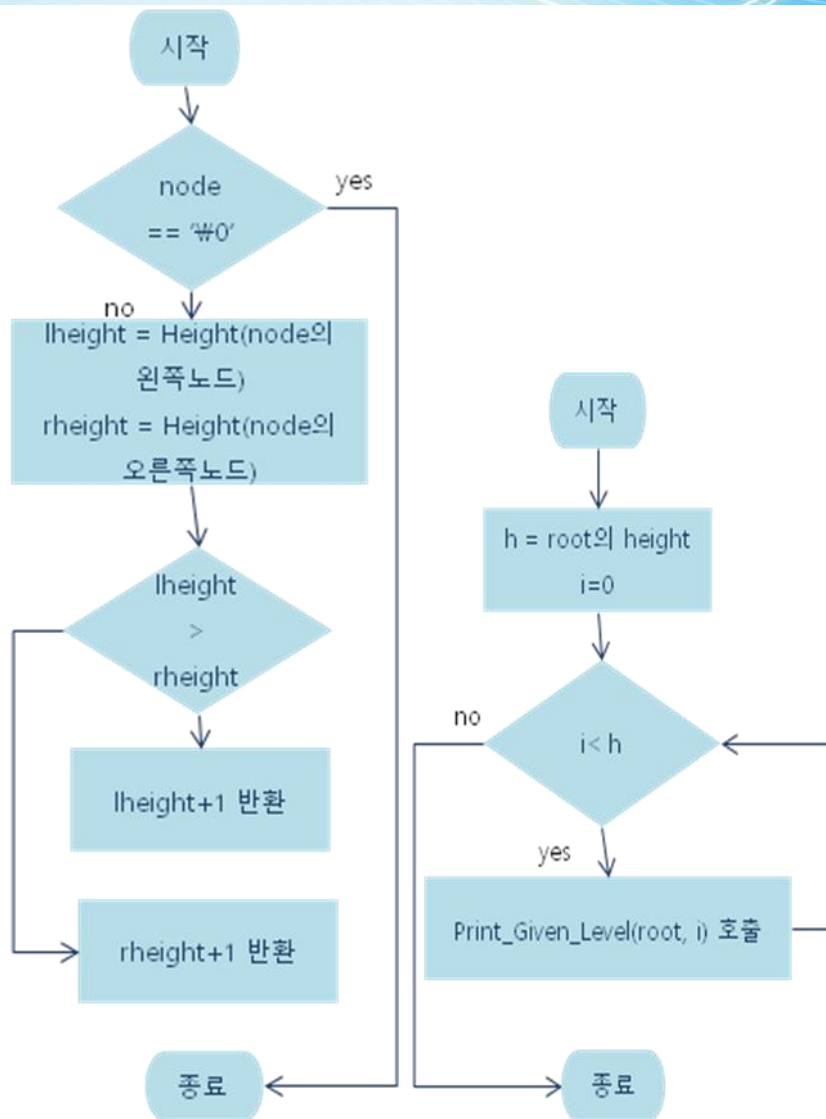


void WordBST::Print_pr_re(WordNode* node)



void WordBST::Print_po_re(WordNode* node)

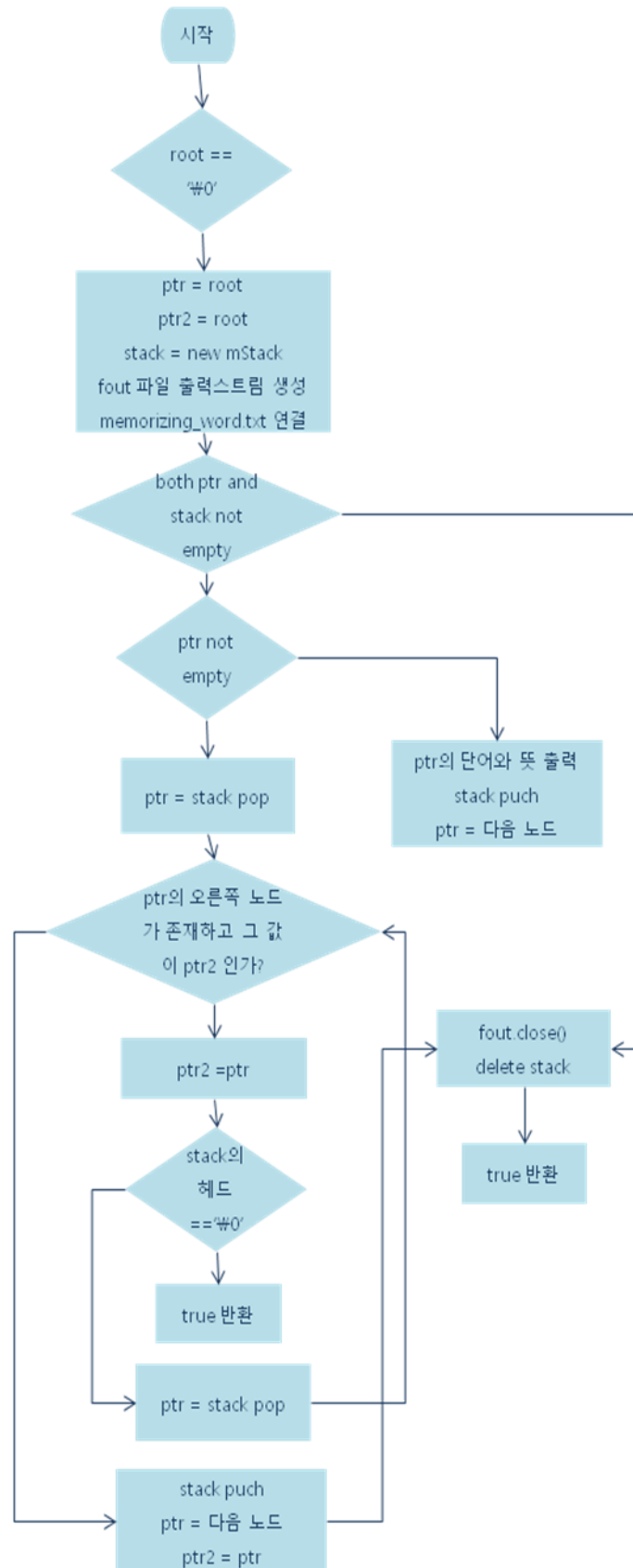
int WordBST::Height(WordNode* node)



void WordBST::Print_Level()

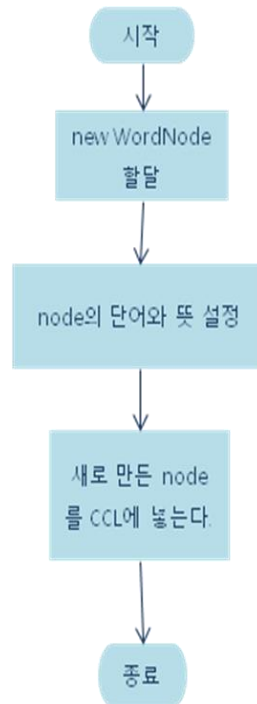
k) 반복문 print

l)bool WordBST::Save()

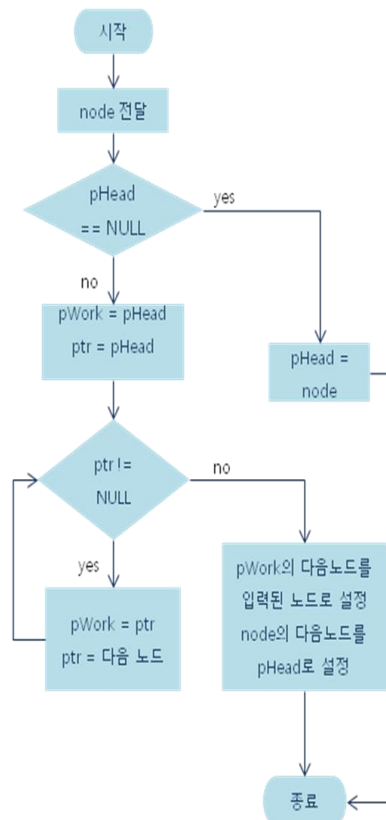


3) CircularLinkedList

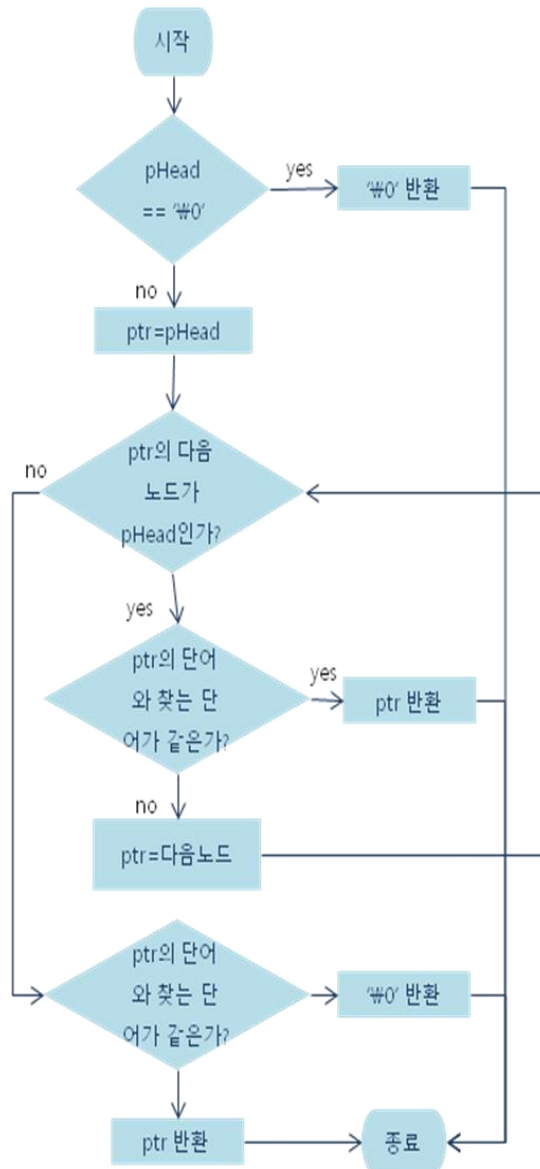
a) void CircularLinkedList::Insert(const char* word, const char* mean)



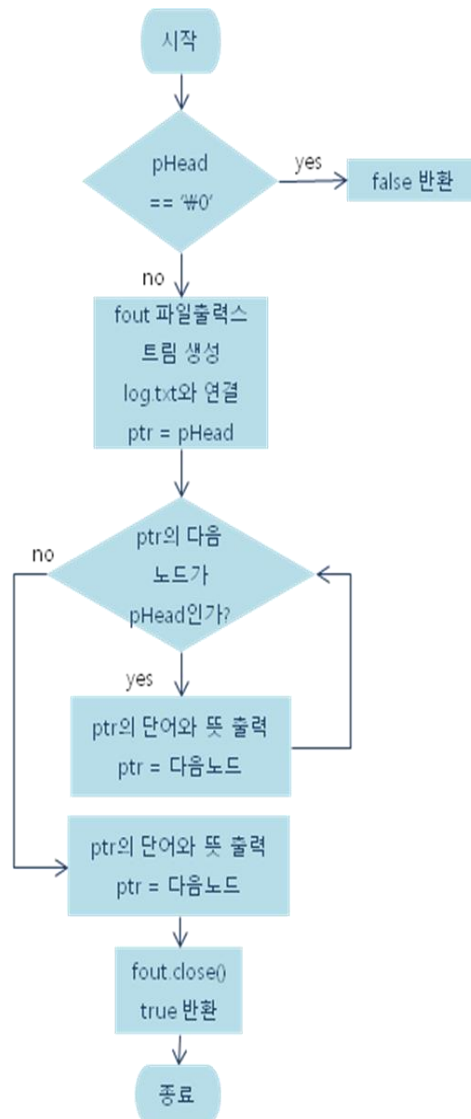
b) void CircularLinkedList::Insert(WordNode* node)



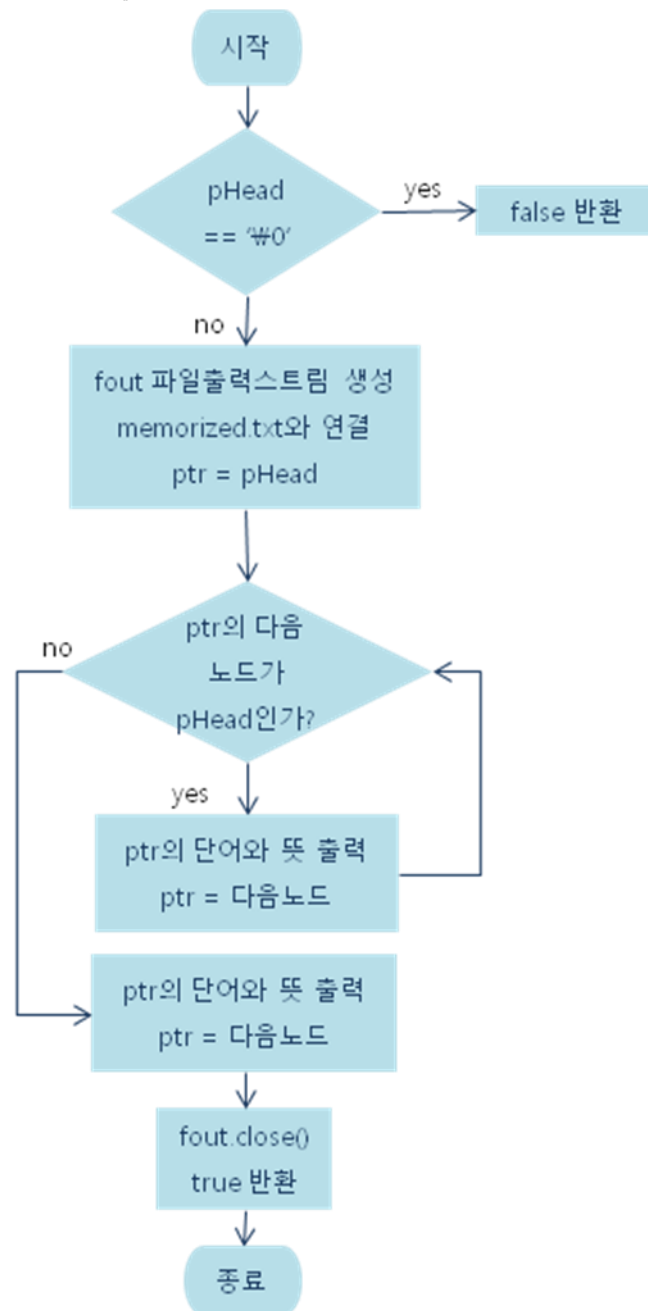
c) WordNode* CircularLinkedList::Search(const char* word)



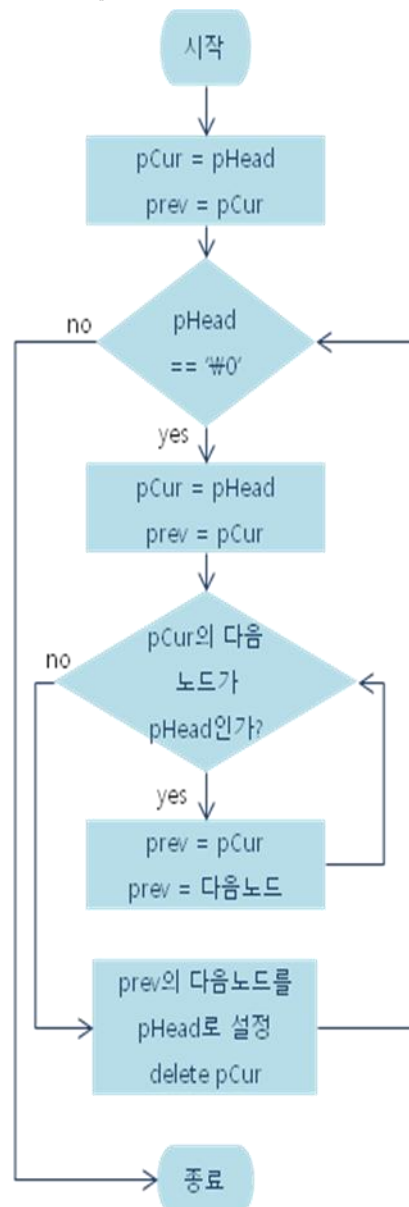
d) bool CircularLinkedList::Print()



e) bool CircularLinkedList::Save()



f) CircularLinkedList::~CircularLinkedList()

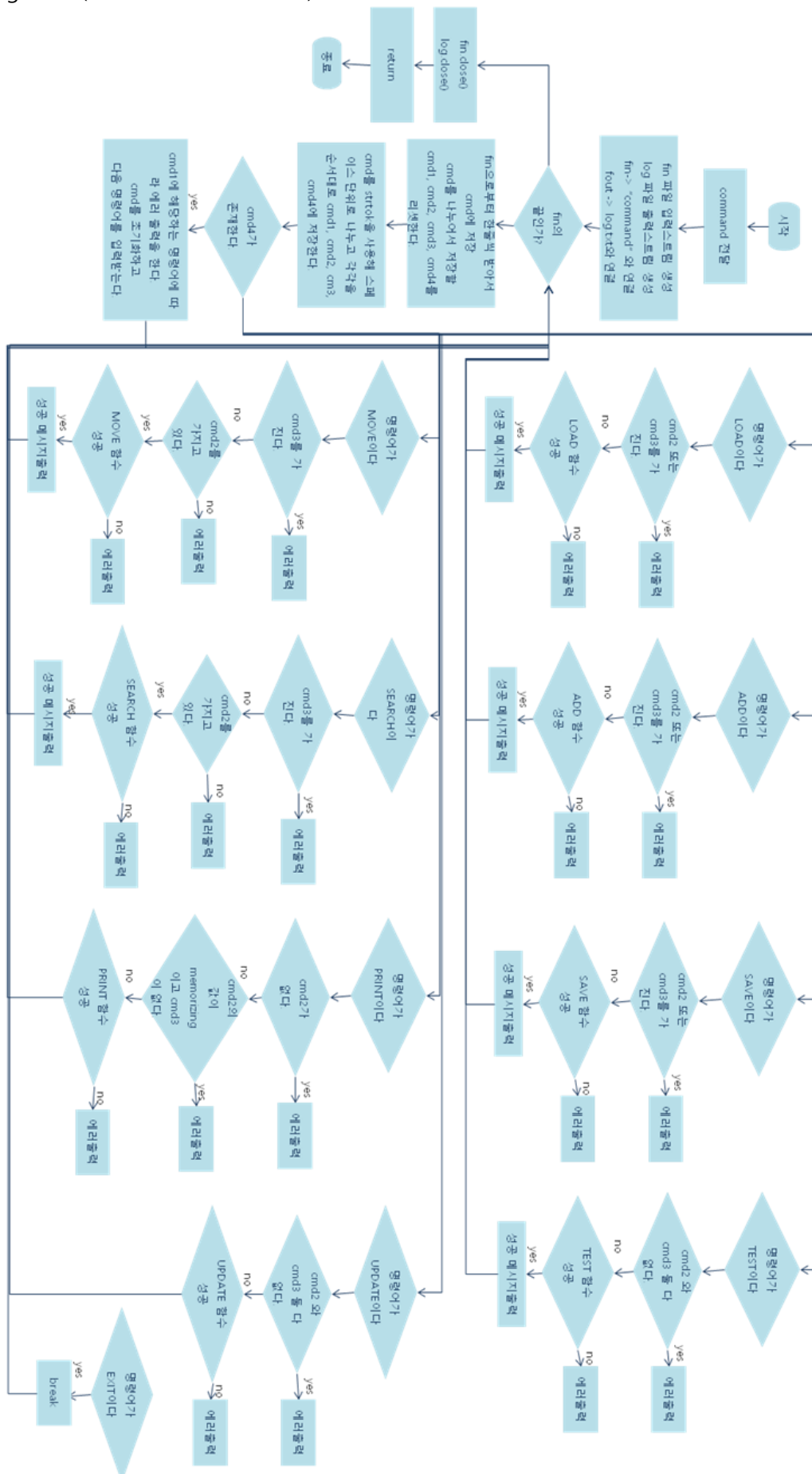


4) main 함수, Manager

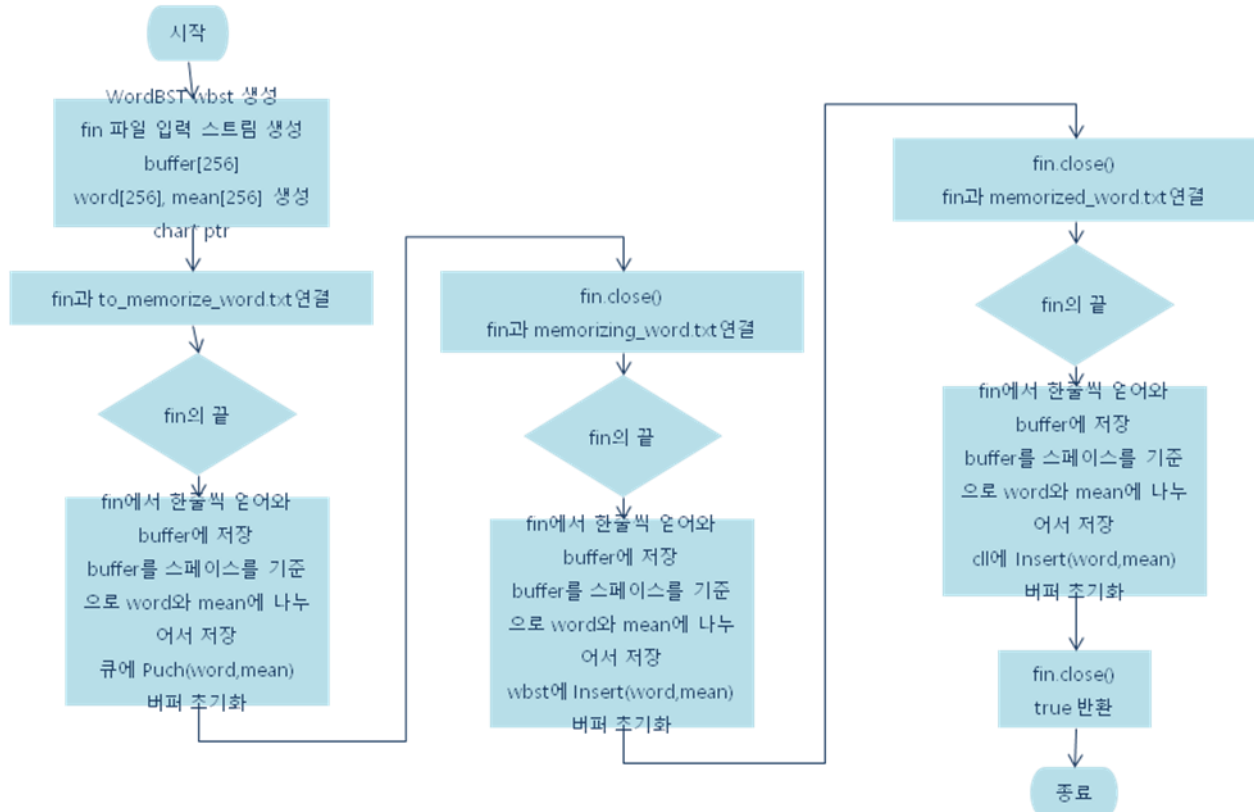
a)



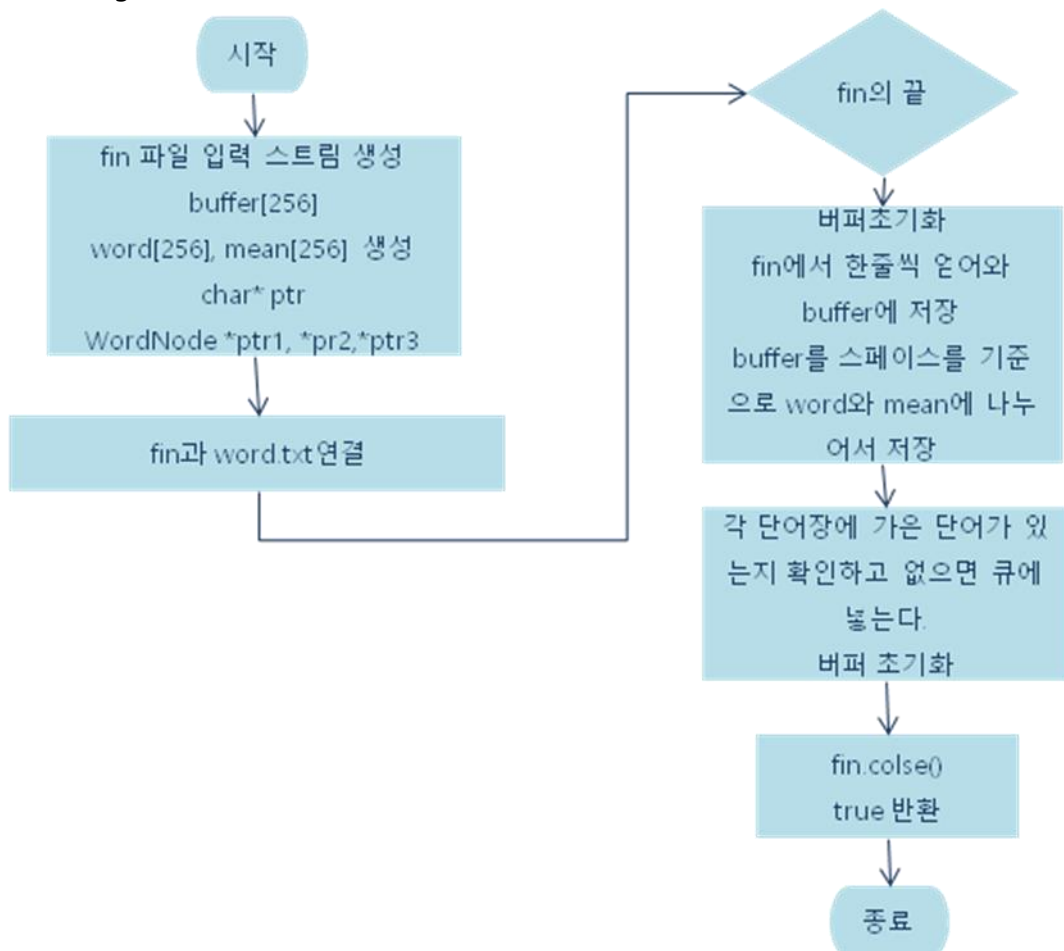
b) void Manager::run(const char * command)



c) bool Manager::LOAD()



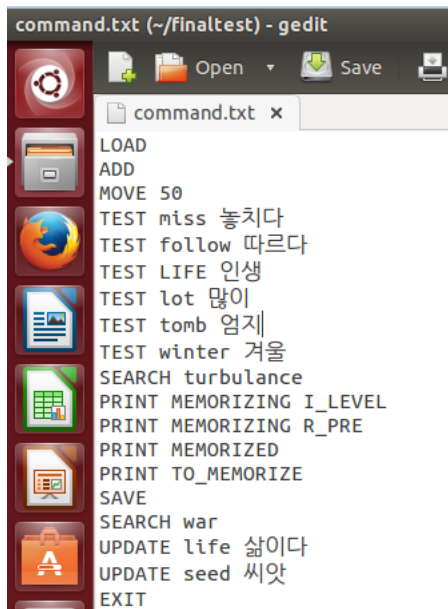
d) bool Manager::ADD()



3. Algorithm

Command.txt.파일을 최초로 인식하여 해당 글 안에 있는 명령어들을 인식해서 올바른 명령어인지 판단을 내린 후 해당 명령어에 맞도록 수행을 해줘야 한다. 가장 중요한 알고리즘으로는 BST 부분이였다. 단순한 BST가 아닌 이중 BST로 1차에서 해당 노드를 검색해서 꺼낸 다음 2차 BST에 삽입을 하는 방식이다. 또한 BST를 삽입하면서 해당 노드보다 작은 값이면 왼쪽으로, 큰값이면 오른쪽을 삽입을 진행하며 이러한 삽입과정을 재귀함수를 통해서 수행하였다. 출력은 2가지 방법으로 재귀함수를 사용한 방법과 반복문과 스택 저장소를 사용하여 출력하는 방법을 사용했다. 전위후위중위 2개씩 그리고 레벨순위로 총7개의 출력형식을 가지게 된다. 삭제를 수행할때에는 삭제대상인 노드를 먼저 찾은뒤에 해당 부모노드와 자식들중 왼쪽 자식들중에서 가장 큰 값, 혹은 오른쪽 자식들 중에서 가장 작은 값을 꺼내서 연결하도록 하고 해당 노드는 좌우 연결을 비워준 다음 반환해주는 것으로 마무리 지었다.

4. Result Screen



```
command.txt (~/.Finaltest) - gedit
LOAD
ADD
MOVE 50
TEST miss 놓치다
TEST follow 따르다
TEST LIFE 인생
TEST lot 많이
TEST tomb 엄지
TEST winter 겨울
SEARCH turbulence
PRINT MEMORIZING I_LEVEL
PRINT MEMORIZING R_PRE
PRINT MEMORIZED
PRINT TO_MEMORIZE
SAVE
SEARCH war
UPDATE life 삶이다
UPDATE seed 씨앗
EXIT
```

위와 같이 command파일을 작성하고 컴파일하면

```
shung2@ubuntu:~/finaltest$ ./run
===== LOAD =====
Success
=====
===== ADD =====
Success
=====
===== MOVE =====
Success
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== ERROR =====
500
=====
===== TEST =====
Pass
=====
===== ERROR =====
600
=====
```

Load, add, move가 성공적으로 실행된 것이 보여졌다. 위의 miss와 follow는 memorizing파일에 단어가 없으므로 에러코드가 출력되었고 life, lot, winter는 이미 memorized에 있으므로 에러코드가 출력되었다. Tomb는 뜻이 잘못 되어 에러코드가 출력되었다. memorizing파일에 있던 winter는 제대로 test되어 성공되었음을 볼 수 있다. 단어장 어디에도 turbulence라는 단어가 없으므로 search의 에러코드에 해당하는 600을 출력했다.

```
=====PRINT=====
hand      손
different  다른
letter     편지
listen     듣다
light      빛
bird       새
fire       불
face       얼굴
fun        재미
job        직업
activity   활동
autumn     가을
clothes    옷
clean      깨끗한
course     강좌
each       각각
example    예
enjoy      즐기다
important  중요한
=====
```

memorizing단어장의 Level order로 출력한 결과 값이다.

```

=====PRINT=====
hand    손
different    다른
bird    새
activity    활동
autumn   가을
clothes  옷
clean    깨끗한
course   강좌
fire     불
face     얼굴
fun      재미
each     각각
example  예
enjoy    즐기다
letter   편지
listen   듣다
light    빛
job      직업
important    중요한
=====

```

memorizing단어장의 pre order로 출력한 결과 값이다.

```

=====PRINT=====
life     인생
winter   겨울
lot      많이
earth    지구
=====

```

memorized단어장에 들어있는 단어를 출력한 것이다.

machine	기계		cover	덮개		secret	비밀		=====PRINT=====	
fact	사실		catch	잡다		luck	행운		famous	유명한
rule	규칙		wash	씻다		success	성공		special	특별한
die	죽다		dark	더러운	환경	coin	동전		just	단지
busy	바쁜		dirty	더러운	환경	goods	상품		nature	자연
sick	아픈		tired	피곤한	환경	address	주소		restaurant	식당
health	건강		upset	속상한	환경	circle	원		group	집단
holiday	공휴일		environment	환경	환경	mix	섞다		habit	습관
goal	목표		pollution	오염	환경	library	도서관		culture	문화
order	순서		lake	호수	환경	museum	박물관		information	정보
result	결과		desert	사막	환경	ocean	바다		advertisement	광고
half	절반		insect	곤충	환경	sentence	문장		science	과학
decide	결정하다		accident	사고	환경	proverb	속담		gene	유전자
choose	선택하다		collage	대학	환경	invention	발명		war	전쟁
difficult	어려운		advice	조언	환경	trash	쓰레기		store	가게
foreign	외국의		hobby	취미	환경	moment	순간		sound	소리
brain	뇌		mark	표시	환경	product	제품		fly	날다
voice	목소리		post	우편	환경	ring	반지		easy	쉬운
opinion	견해		receive	받다	환경	date	날짜		poor	가난한
age	나이		add	더하다	환경	peace	평화		fast	빨리
price	가격		physical	신체의	환경	storm	폭풍		back	뒤
dish	접시		modern	현대의	환경	disease	병		always	언제나
subway	지하철		pretty	예쁜	환경	ghost	유령		history	역사
bear	곰		clerk	사무원	환경	bone	뼈		state	상태
human	인간		foot	발	환경	wisdom	지혜		soldier	군인
buy	사다		company	회사	환경	candle	초		village	마을
sell	팔다		factory	공장	환경	zoo	동물원		office	사무실
follow	따르다		garage	차고	환경	scene	장면		island	섬
miss	놓치다		palace	궁전	환경	memory	기억		piece	조각
close	닫다		hole	구멍	환경	recipy	조리법		rock	바위
delicious	맛있는		bill	청구서	환경	wheel	바퀴		line	선
sad	슬픈		seed	씨	환경	kid	아이		cook	요리사
however	그러나		medicine	약	환경	smoke	연기		end	끝
diary	일기		meaning	의미	환경	cute	귀여운		remember	기억하다
cartoon	만화		race	경주	환경	already	이미		wear	입다
event	사건		collect	모으다	환경	hometown	고향		send	보내다
toy	장난감		heavy	무거운	환경	yard	마당		hot	뜨거운
weight	무게		useful	유용한	환경	bridge	다리		early	일찍
smart	영리한		maybe	아마	환경	fairy	요정		often	자주
president	대통령		sense	느낌	환경	flag	깃발		sometimes	때때로
arm	팔		image	이미지	환경	pain	고통		pet	애완동물
meal	식사		map	지도	환경	guide	안내자		vegetable	채소
skill	기술		type	유형	환경	promise	약속		leaf	잎
contest	경쟁		project	계획	환경	bake	굽다		forest	숲
prize	상품		traffic	교통	환경	background	배경		area	지역
chance	기회		safety	안전	환경	distance	거리		neighbor	이웃
shape	모양		spaceship	우주선	환경	behavior	행동		art	미술
wall	벽		rainbow	무지개	환경	danger	위험		poem	시
smell	냄새		seat	좌석	환경	object	물건		subject	주제
judge	판단하다		magic	마술	환경	tear	눈물		bottle	병

honor	명예			
expensive	비싼			
ability	능력			
action	행동			
pride	자존심			
slave	노예			
rope	줄			
tip	노릇			
sunrise	일출			
garbage	쓰레기			
cash	현금			
police	경찰			
dictionary	사전			
adult	어른			
brick	벽돌			
temple	사원			
wing	날개			
purpose	목적			
god	신			
guest	손님			
diver	잠수부			
cell	세포			
crowd	군중			
dig	파			
wood	나무			
cave	동굴			
tomb	무덤			
tool	도구			
satellite	위성			
wealth	재산			
attention	주의			
fold	접다			
lawyer	변호사			
blood	피			
gun	총			
spirit	정신			
route	길			
pole	막대기			
rubber	고무			
root	뿌리			
structure	구조			
vote	투표			
patient	환자			
quick	빠른			
captain	선장			
bucket	양동이			
cage	새장			
kite	연			
miracle	기적			
problem	문제			

story	이야기
name	이름
place	장소
work	일
make	만들다
person	사람
movie	영화
part	부분
plan	계획
plant	식물
learn	배우다
same	같은
trip	여행
vacation	휴가
summer	여름
spring	봄
space	공간
street	거리
paper	종이
newspaper	신문
mind	마음
volunteer	자원봉사자
visit	방문
start	시작
watch	시계
win	이기다

To_memorize에 있는 단어를 출력한 것이다.

```

===== SAVE =====
Success
=====
===== SEARCH =====
war    전쟁
=====
===== UPDATE =====
life   인생 -> 삶이다
=====
===== UPDATE =====
seed   씨 -> 씨앗
=====

```

성공적으로 save를 실행했음을 볼 수 있고 단어장에 있는 단어 war를 찾은 것을 볼 수 있다. 또한, update를 제대로 수행한 것을 볼 수 있다.

위를 컴파일한 상태에서

```

LOAD
ADD
MOVE 50

PRINT MEMORIZING I_IN
PRINT MEMORIZING I_PRE
PRINT MEMORIZING R_POST
PRINT MEMORIZED
PRINT TO_MEMORIZE
SAVE
SEARCH war|
UPDATE turbulence 난기류
UPDATE seed 씨앗
MOVE 120

EXIT

```

이것을 컴파일했고 print to-memorize의 경우 같은 결과 값을 출력하였다.


```

=====PRINT=====
activity      활동
autumn        가을
bird          새
clean         깨끗한
clothes       옷
course        강좌
different     다른
each          각각
enjoy         즐기다
example       예
face          얼굴
fire          불
fun           재미
hand          손
important     중요한
job           직업
letter        편지
light         빛
listen        듣다
=====

```

Memorizing을 inorder로 출력한 것이다.

```

=====PRINT=====
hand          손
different     다른
bird          새
activity      활동
autumn        가을
clothes       옷
clean         깨끗한
course        강좌
fire          불
face          얼굴
fun           재미
each          각각
example       예
enjoy         즐기다
letter        편지
listen        듣다
light         빛
job           직업
important     중요한
=====

```

Memorizing을 preorder로 출력한 것이다.

```

=====PRINT=====
autumn        가을
activity      활동
clean         깨끗한
course        강좌
clothes       옷
bird          새
enjoy         즐기다
example       예
each          각각
face          얼굴
fun           재미
fire          불
different     다른
important     중요한
job           직업
light         빛
listen        듣다
letter        편지
hand          손
=====

```

Memorizing을 postorder로 출력한 것이다.

```

===== SAVE =====
Success
=====
===== SEARCH =====
war    전쟁
=====
===== ERROR =====
800
=====
===== UPDATE =====
seed   씨 -> 씨앗
=====
===== ERROR =====
300
=====

```

위와 마찬가지로 save, search가 제대로 실행됨을 알 수 있고, 단어장의 turbulence라는 단어를 찾아 update하라는 명령을 보냈지만 단어가 없어 에러코드를 보낸 것을 알 수 있다. 또한, move명령어 시행 시 100개가 넘어가는 단어를 이동시키게 되면 에러코드가 뜨는 것을 확인할 수 있다.

5. Consideration

이름	프로젝트에서 맡은 역할	본인 스스로 생각하는 자신의 점수(10)
진세진	Momorizing 단어를 저장하는 2-D BST와 주석 및 전체 코드 이식 및 종합적인 디버깅 및 수정	8/10
고찰	초반에 여유를 가지고 생각하여 마지막에 서두른 결과를 만들다 보니 제대로 구현하지 못한 부분이 많았다. 자료들을 저장하고 큐에서 BST로, 다시 BST에서 Circular Linked List로 이동시키는 부분은 완벽하지만 해당 사항들을 명령어를 인식해서 출력하는 부분에서 결과화면으로 출력하거나 log.txt 파일에 출력하는게 서로 맞지않아 어려움이 있었다. 부족한 부분이 없지 않아 있던 팀프로젝트였다. 아쉬운 점으로는 조금 더 빠른 결정을 했다면 원활하게 완성할 수 있었을 거라 장담한다.	
이선희	Memorize에서 사용하는 CCL.h , CCL.cc와 Manager::run 함수, 보고서 Flowchart와 알고리즘을 맡았다.	7/10
고찰	팀과제를 하면서 역할의 분담과 시간의 적절한 분배의 중요성을 알게 되었다. 팀 프로젝트를 하면서 혼자 하는 것 보다 더 나은 방향으로 나아갈 수 있었고 자신의 어느 부분들이 부족했는지를 인지하는 계기가 되었다. 모르는 부분들을 알아갈 수 있었다.	
한승주	To-memorize에서 사용하는 Queue와 보고서의 Introduction, Result의 작성을 맡았다.	7/10
고찰	내가 맡았던 곳에서 어려웠던 점은 메모리를 할당시키고 소멸자에서 소멸시키는 것에 대한 이해가 부족하여 구현하는 것이 어려웠다. 그리고 리눅스 사용법이 익숙치 않아 string함수를 사용하기 위해서 기존의 비주얼 스튜디오에서는 cstring헤더함수를 사용하지만 리눅스에서는 표준함수인 string.h를 사용해야하는 것을 알았다. 프로그램을 run한 후 코드를 수정할 경우 debug하여 재설정해준다. 이때 새로 컴파일한 컴파일파일로 돌려야 다시 짠 코드가 돌아가고 이전에 돌렸던 run으로 할 경우 바뀌기 이전의 코드로 컴파일된다. 개인 사정으로 팀에 늦게 합류하게 되어 이미 완성되어 있는 코드가 어느 정도 있었고 남았던 Queue를 내가 짰다. 팀프로젝트의 좋은 점은 고쳐지지 않거나 어려운 부분을 서로 상의하여 채워줄 수 있다는 점이 좋았다. 현재의 팀에서는 소통이 빠르게 오고 가서 빠르고 원활하게 팀프로젝트가 진행되었지만 기존의 있던 팀에서는 팀원 간의 연락 부재로 인하여 소통의 어려움이 있었고 팀프로젝트시 팀원 간의 소통이 원활하지 않았다면 어렵게 진행된다는 점이 어려웠다.	

6. Reference

반복문을 통한 순회 /

http://www.nicklib.com/index.php?mid=algorithm&sort_index=readed_count&order_type=asc&document_srl=2011

레벨오더 순회 /

<http://www.geeksforgeeks.org/level-order-tree-traversal/>

트리삽입 및 소멸자 /

<http://www.cprogramming.com/tutorial/lesson18.html>

