데이터 구조 실습

Project #1

학　　과: 컴퓨터공학과

팀　　장: 김태연

학　　번: 2013722016

팀　　원: 김동욱

학　　번: 2013722010

팀　　원: 황성영

학　　번: 2013722030

1) Introduction

이번 프로젝트에서는 영어 단어장 프로그램을 구현한다. 이 프로그램엔 3가지 단어장이 있다. 외워야 할 단어장(TO_MEMORIZE), 현재 외우고 있는 단어장(MEMORIZING), 다 외운 단어장(MEMORIZED) 이렇게 3가지로 구성된다. 이 3가지의 단어장은 각각 Queue, 이진 탐색 트리(Binary Search Tree, BST)와 환형 연결 리스트(Circular Linked List)로서 각각 다른 Linked-List를 이룬다. 그리고 각자의 자료구조의 WordNode에 영어단어와 한글 뜻이 저장되어 있는 데이터가 존재한다. command.txt 파일 안에 있는 각각의 명령어를 통하여 이 3가지 단어장 내에 존재하는 단어들을 명령어에 따라 명령을 수행한다.

* 외워야 할 단어장(TO_MEMORIZE)

자료구조 Queue를 이용하여 구현한다. 자료구조 Queue는 한쪽 끝에서 원소들이 Pop되고, 반대쪽 끝에서 노드가 Push되는 자료구조 이다. Push가 제일 먼저 된 순서대로 노드가 가장 먼저 Pop이 된다. word.txt 파일에 있는 새로운 단어를 Queue의Push를 이용하여 외워야 할 단어장(TO_MEMORIZE) 안에 단어를 넣는다. 그리고 Queue의 Pop을 이용하여 외워야 할 단어장(TO_MEMORIZE) 에서 현재 외우고 있는 단어장(MEMORIZING)으로 단어를 넘긴다. Print할 때에는 가장 먼저 Push한 Node의 Data부터 순차적으로 출력이 된다. 또 Save할 때에는 Print할 때처럼 가장 먼저 Push한 Node의 Data부터 순차적으로 to_memorize.txt파일에 저장된다. 이어서 Search를 수행할 때에는 가장 먼저 Push한 Node부터 순차적으로 찾고자 하는 Node의 Data와 비교하며 찾는다.

* 현재 외우고 있는 단어장(MEMORIZING)

자료구조 Binary Search Tree, BST를 이용해 구현한다. BST는 트리 구조에서 가장 위에 위치하는 root를 기준으로 root의 data값 보다 작으면 왼쪽, 크면 오른쪽으로 뻗어 나가는 트리 구조이다. 우선, AlphabetBST를 구현한다. A~Z의 각 AlphabetNode 내에 맨 앞의 알파벳이 동일한 WordBST가 구성되어 있다. TO_MEMORIZE에서 Queue의 Pop을 통하여 넘어온 Node를 먼저 AlphabetBST에서 맨 앞의 글자를 비교하여 어느 AlphbetNode로 들어갈지 먼저 찾는다. 그리고 그 안에 있는 WordBST에 넘어온 Node를 Insert 하게 된다. 그리고 외운 단어는 해당 WordNode를 트리 구조에서 Delete하고, 다 외운 단어장(MEMORIZED)으로 해당 Node를 넘긴다. 여기서 BST에서 Node를 제거할 때는 양쪽 자식이 모두 존재하는 경우, 왼쪽자식만 있는 경우, 오른쪽 자식만 있는 경우, 자식이 없는

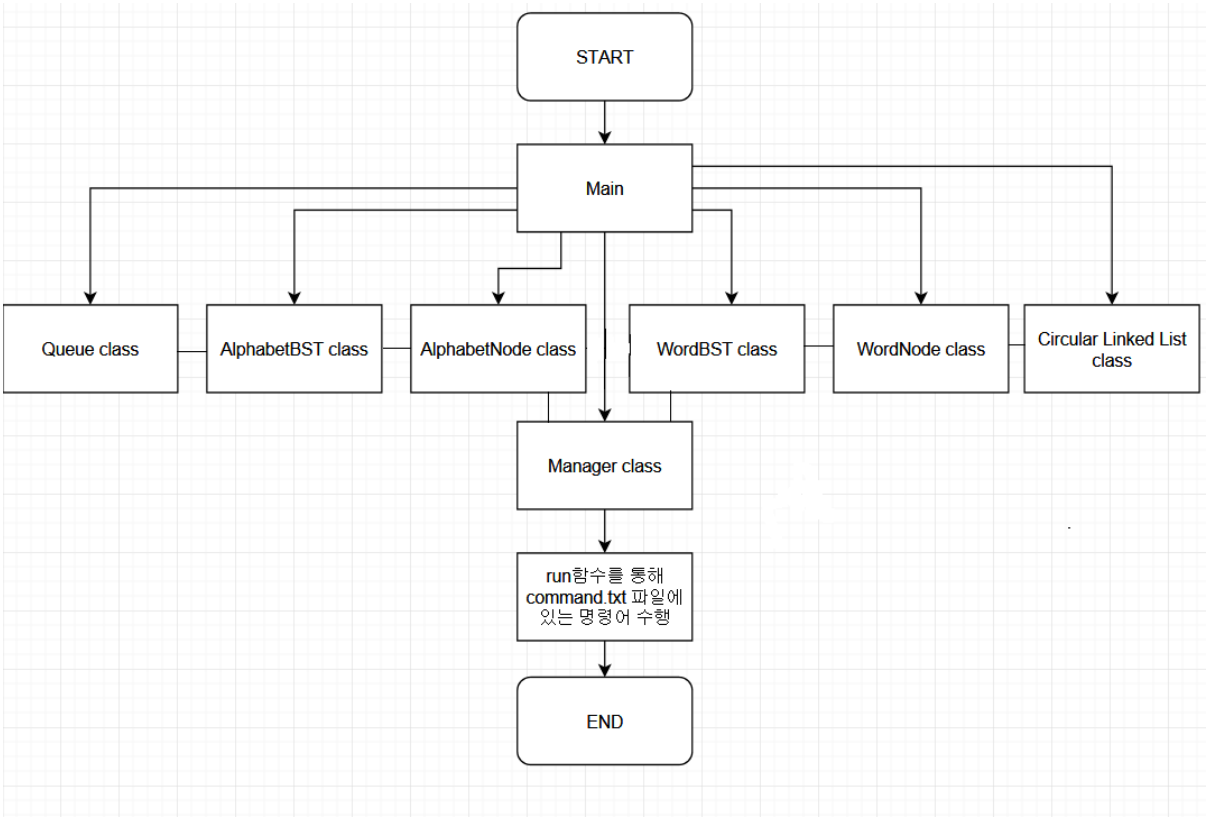
경우 이 경우의 수를 모두 생각하여야 한다. 그리고 BST Print 방법에는 Pre-Order, In-Order, Post-Order, Level-Order 등 여러 가지 순회방법이 존재하는데 여기서 Print 명령어를 통하여 어떠한 순회방식으로 순회하여 출력하는지 결정한다. 여기서 Print 출력할 때에는 AlphabetBST로 먼저 들어가서 알파벳을 순회하며 그 알파벳과 맨 앞 글자가 같은 WordBST에 들어가 WordNode의 Data를 순회하여 출력한다. 그 다음으로 Save를 할 때에는 텍스트파일에 알파벳 순으로 저장을 하게 되면 LOAD할 때 BST의 자식 노드가 한 쪽으로 쏠려 Skewed Binary Tree가 되므로 Pre-Order의 순회방법으로 WordNode의 Data를 memorizing.txt파일에 저장한다. 여기서 Search 검색 방법은 root인 Node부터 BST의 특징에 따라 찾고자 하는 Node의 Data와 비교하며 찾고자 하는 Node의 Data가 더 작으면 왼쪽 자식 Node로 더 크면은 오른쪽 자식 Node로 가며 비교한다.

* 다 외운 단어장(MEMORIZED)

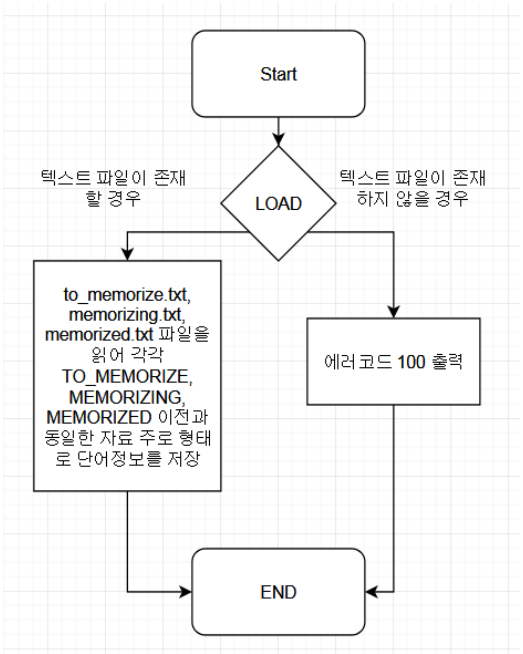
자료구조 Circular Linked-list를 이용하여 구현한다. Circular Linked-list는 Single Linked-list 자료 구조에서 제일 마지막으로 Insert한 Node의 다음 Node가 제일 먼저 Insert한 Node를 가리킨다. 이러한 방식으로 Circular을 이룬다. Circular Linked-list에서 Insert는 Queue의 Push와 매우 유사하다. 거기서 제일 마지막으로 Insert한 Node의 다음 Node를 제일 먼저 Insert한 Node로 연결하면 된다. MEMORIZING 단어장에서 단어를 다 외우게 되면 MEMORIZING 단어장에서 BST의 Delete를 이용하여 MEMORIZED 단어장으로 단어정보를 넘겨받는다. 여기서 Print 출력 방법은 Queue의 Print 출력 방법과 동일하게 제일 먼저 Insert한 Node부터 순차적으로 Node의 Data를 출력한다. 여기서 Save 저장 방법 역시 Queue의 Save 저장 방법과 동일하다 제일 먼저 Insert한 Node의 Data부터 순차적으로 memorized.txt파일에 저장한다. 이어서 Search 검색 방법 역시 제일 먼저 Insert한 Node부터 찾고자 하는 Node의 Data와 비교하며 검색을 시행한다.

2) Flow Chart

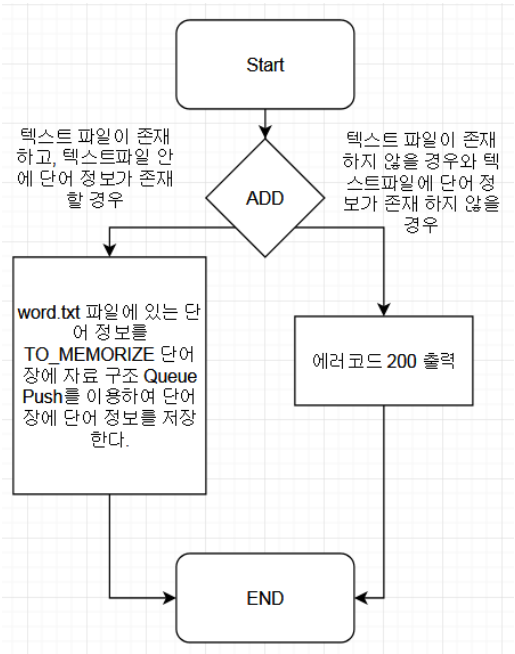
-MAIN



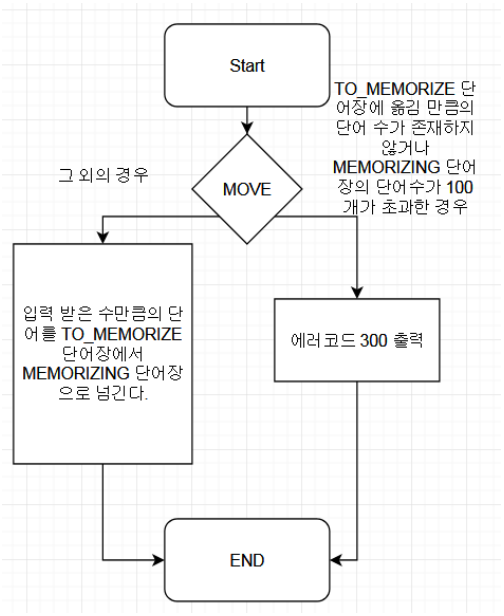
-LOAD



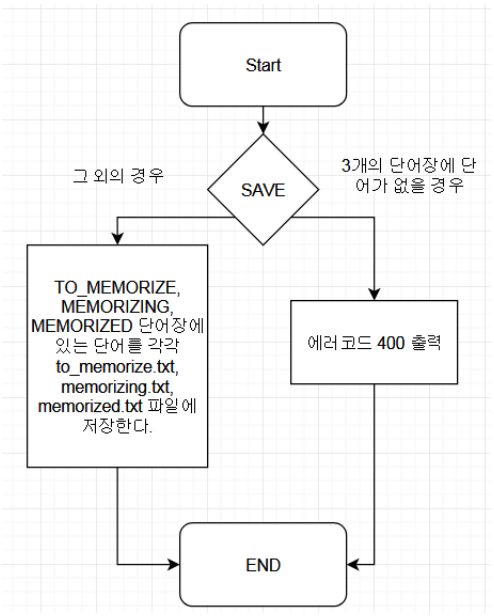
-ADD



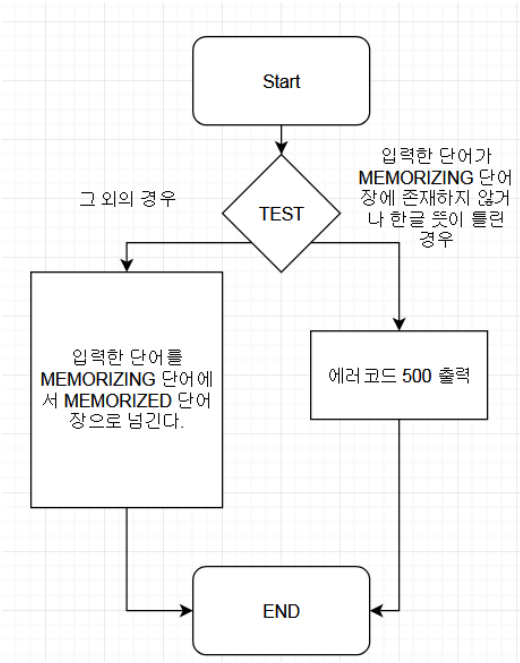
-MOVE



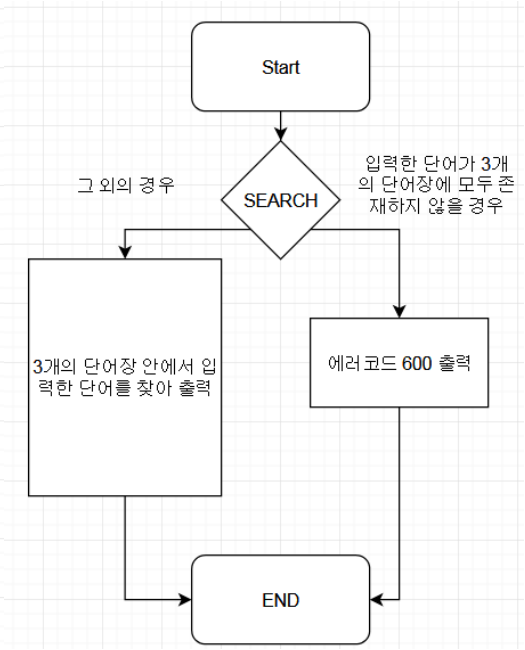
-SAVE



-TEST

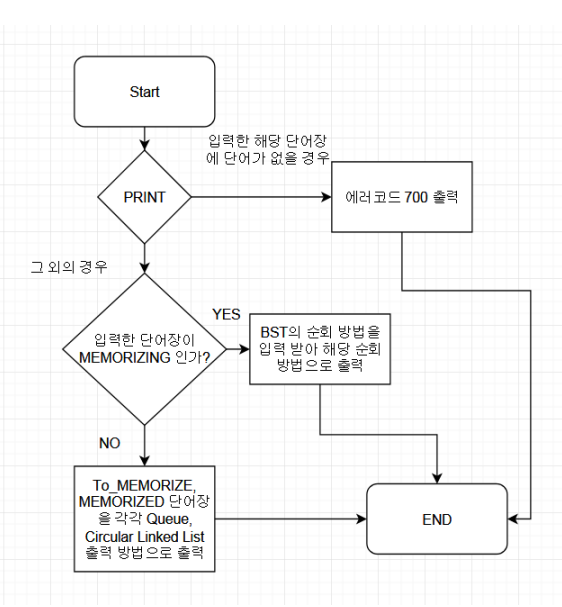


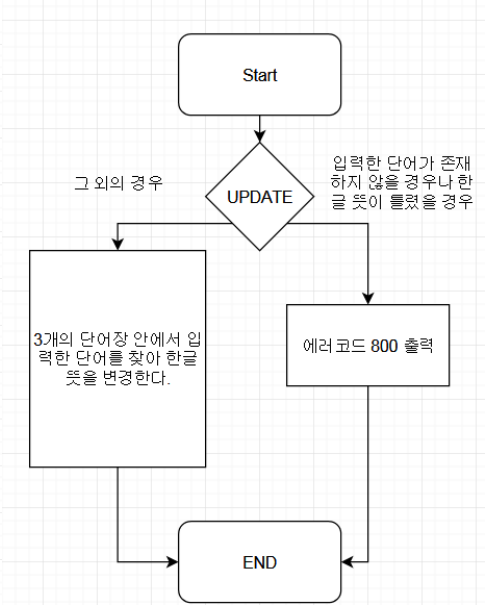
-SEARCH



-PRINT

-UPDATE





3) Algorithm

이번 프로젝트에서는 3가지 영어 단어장이 있다. 외워야 될 단어장(TO_MEMORIZE) 외우고 있는 단어장(MEMORIZING) 다 외운 단어장(MEMORIZED) 이렇게 3가지가 있다. 이 3가지 단어장의 단어들을 각각의 LOAD, ADD, MOVE, SAVE, TEST, SEARCH, PRINT, UPDATE, EXIT 명령어들을 통해 다룬다. 각각의 명령어들을 수행하여 성공하였을 시에는 명령을 수행하고 알맞은 결과를 출력한다. 그리고 실패하였을 시에는 알맞은 에러코드를 출력한다.

- LOAD : 기존의 단어장을 불러오는 명령어이다. To_memorize.txt, memorizing.txt, memorized.txt 의 텍스트 파일 안에 단어장 정보가 존재할 경우에 단어장 정보를 읽어 각각 TO_MEMORIZE, MEMORIZING, MEMORIZIED 단어장에 이전과 동일한 Linked-List 자료 구조에 단어장 정보를 저장한다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있을 경우에 에러코드를 출력한다. LOAD를 성공하였을 시에는 Success라는 문구를 출력한다.

- ADD : 단어 텍스트 파일에 있는 단어 정보를 읽어오는 명령어이다. word.txt 파일 안에 있는 단어 정보들을 TO_MEMORIZE 단어장에 Queue의 Push로 순차적으로 추가시킨다. 텍스트파일이 존재하지 않거나 텍스트파일에 단어가 존재하지 않을 경우 에러코드를 출력한다. Word.txt 파일에 있는 단어들을 ADD하는데에 성공할 시에는 Success라는 문구를 출력한다.

- MOVE : TO_MEMORIZE 단어장에서 MEMORIZING 단어장으로 단어를 옮기는 명령어이다. 1 과100사이에 있는 정수를 입력 받아 입력 받은 수만큼 TO_MEMORIZE 단어장에서 MEMORIZING 단어장으로 단어를 옮긴다. 여기서 입력한 수만큼 넘길 단어가 존재하지 않는다면 에러코드를 출력한다. 그리고 MEMORIZIG 단어장에 있는 단어의 수와 입력 받은 수의 합이 100을 넘지 않도록 하고, 여기서 합이 100이 넘어갈 경우 역시 에러코드를 출력한다. 즉 MEMORIZING 단어장 안에 단어가 100개가 넘어가면 안 된다. 그리고 MOVE를 성공하였을 시에는 Success라는 문구를 출력한다.

- SAVE : 현재 단어장 정보를 저장하는 명령어이다. 각각 TO_MEMORIZE, MEMORIZING, MEMORIZED 단어장 안에 있는 단어들을 to_memorize.txt, memorizing.txt, memorized.txt 텍스트파일에 저장한다. 저장할 단어장 정보가 없을 시 에러코드를 출력한다. 그리고 각 텍스트 파일에 단어장 정보를 SAVE하는데 성공하였을 시에는 Success라는 문구를 출력한다.

- TEST : 단어를 외웠는지 테스트하는 명령어이다. 즉, 입력할 단어를 찾아 그 단어의 뜻이 맞으면 MEMORIZING단어장에 있는 단어를 MEMORIZED 단어장으로 넘어간다. 여기서 입력한 단어가 MEMORIZING단어장에 존재하지 않거나, 단어의 뜻이 틀렸을 경우 에러코드를 출력한다. 그리고 TEST가 성공하였을 시에는 Pass라는 문구를 출력한다.

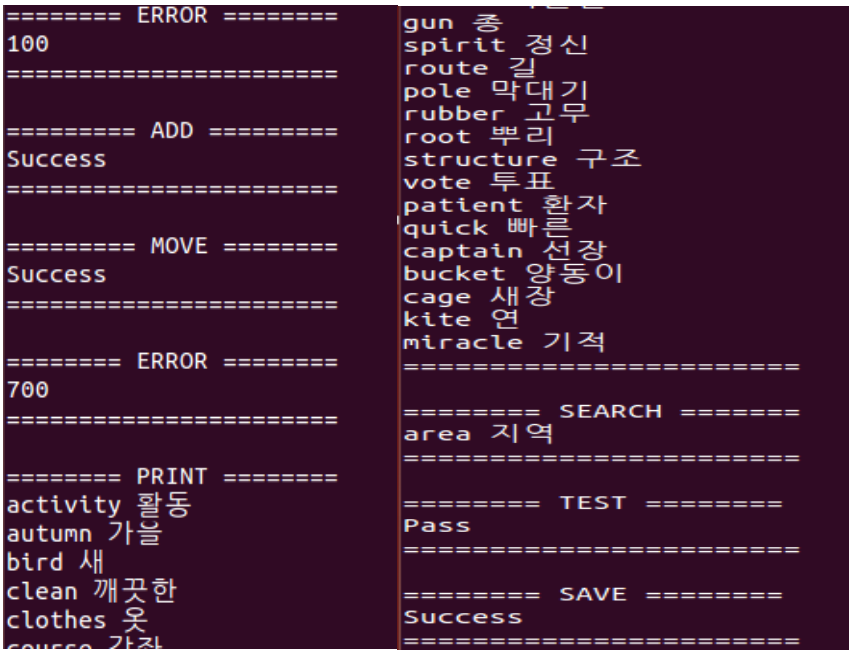
- SEARCH : 단어의 뜻을 찾아 출력하는 명령어이다. 입력한 단어가 TO_MEMORIZE, MEMORIZING, MEMORIZED 단어장에서 입력한 단어를 찾아 있는 존재할 경우 영어단어와 한글 뜻을 출력한다. 입력한 단어가 3개의 단어장 안에 존재하지 않을 경우에는 에러코드를 출력한다.

- PRINT : 입력한 단어장에 있는 단어들을 출력하는 명령어이다. 입력한 단어장이 TO_MEMORIZE일 경우 자료 구조 Queue의 출력처럼 단어의 단어정보를 출력한다. 그리고 입력한 단어가 MEMORIZING일 경우 자료 구조 BST의 출력처럼 출력한다. 여기서 출력 방법이 여러 가지가 있다. 재귀함수를 이용한 In-Order 순회방법, 재귀함수를 사용하지 않은 In-Order 순회방법, 재귀함수를 이용한 Pre-Order 순회방법, 재귀함수를 사용하지 않은 Pre-Order 순회방법, 재귀함수를 이용한 Post-Order 순회방법, 재귀함수를 사용하지 않은 Post-Order 순회방법, Level-Order 순회방법이 있다. 각각의 어떠한 순회 방법을 출력할지 입력을 받는다. 그 입력 받은 순회방법으로 MEMORIZING 단어장 안에 있는 단어의 단어정보를 출력한다. 그리고 입력한 단어가 MEMORIZIED일 경우 자료구조 Circular Linked List의 출력처럼 단어장의 단어 정보들을 출력한다. 만약 입력한 단어장에 정보가 없을 경우 에러코드를 출력한다. 그리고 단어가 존재할 경우에는 각각의 자료구조에 따른 출력 방법에 따라 단어 정보를 출력한다.

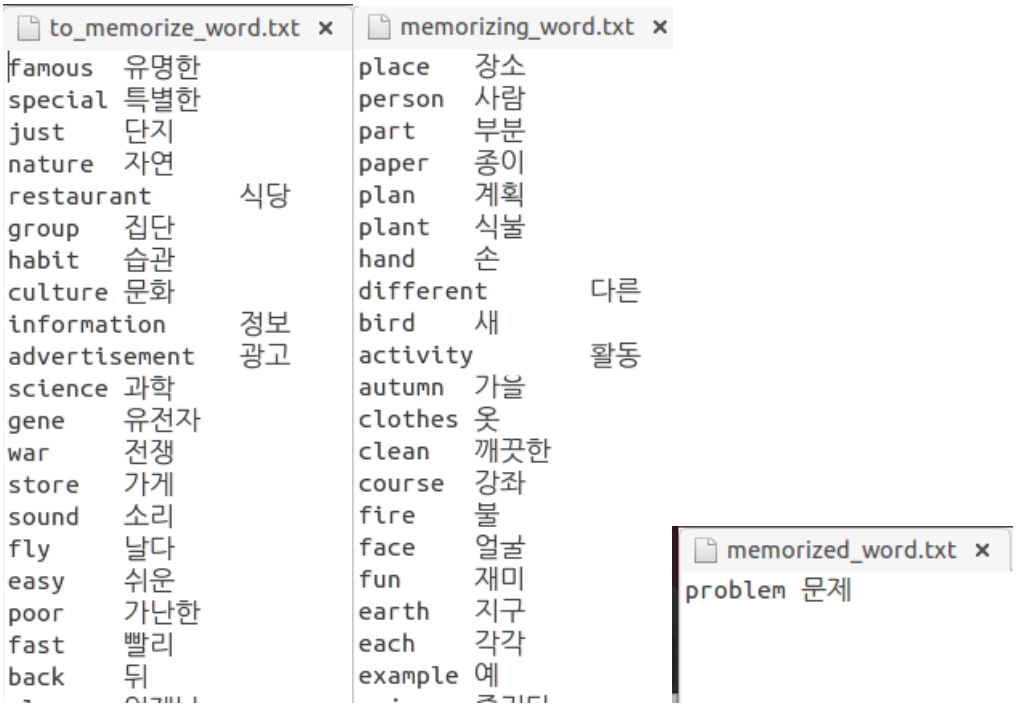
- UPDATE : 단어의 뜻을 변경하는 명령어이다. TO_MEMORIZE, MEMORIZING, MEMORIZED 단어장에 입력한 단어가 존재할 경우 단어의 한글 뜻을 입력한 뜻으로 변경한다. 그리고 변경된 결과를 출력한다. 만약 입력한 단어가 3개의 단어장 안에 존재하지 않을 경우 에러코드를 출력한다.

이와 같은 명령어들에 의해 3개의 단어장을 관리한다. 이러한 관리는 모두 Manager class에서 작동한다. 총지휘관 역할을 하는 셈이다.

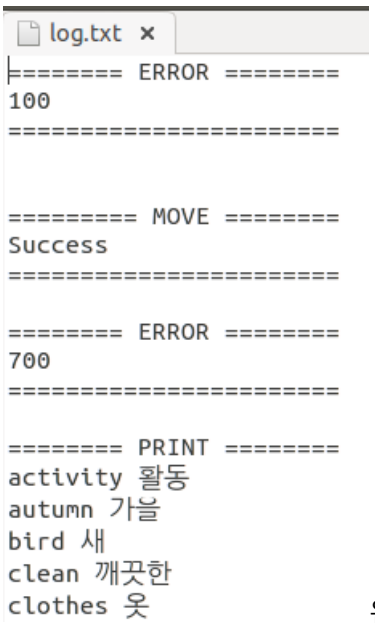
4) Result Screen



위의 그림은 command.txt파일에 적힌 명령어를 수행 후 출력한 화면이다.



위의 그림은 TO_MEMORIZE, MEMORIZING, MEMORIZED 이 3개의 단어장의 단어정보를 각각 to_memorize.txt, memorizing.txt, memorized.txt 파일에 저장한 화면이다.



왼쪽의 그림은 log.txt 파일에 출력된 화면이다.

5) Consideration

김태연 (팀장): print를 할 때 원하는 방법으로 순회하면서 visit함수를 통해 그 순서대로 출력하는 방법을 사용했다. 결과화면을 출력했을 때 reculsive traversal을 했을 때와 nonreculsive traversal을 비교해보니 nonreculsive traversal이 단어 수가 더 적게 출력되 것을 보고 어디가 잘못됐는지 고민하게 됐다. 디버깅을 하면서 이유를 알아보니, stack에 노드들을 넣을 때 BST의 root들을 넣은 것이 이유였다. 비어있는 root들이 있었기 때문에 stack에 push할 때 무시를 하게 됐고, 따라서 무시되고 출력되지 않는단어들이 생겼다. 이를 해결하기 위해서 AlphabetNode에 sample을 추가하게 됐고, 생성자에서 그 AlphabetBST에 맞는 첫 글자를 넣어주었다. stack에 push를 할 때는 sample을 이용해서 push하고 이를 통해 print할 때는 root를 통해 출력하는 방법으로 문제를 해결할 수 있었다.

김동욱 (팀원): 이번 프로젝트에서 가장 헷갈렸던 문제는 MEMORIZING 단어장을 Print할 때에 AlphabetBST에 들어가서 WordNode로 들어가 단어장의 단어 정보를 출력한다. 그런데 말처럼 잘 코드가 짜여지지 않고, 이중 BST여서 너무나 헷갈렸다. 그런데 AlphabetBST에서 순회를 할 때 Visit함수에서 WordBST로 들어가 WordBST에 있는 단어 정보를 출력하면 되었다. 그리고 리눅스에서 컴파일을 처음으로 해보았는데 신기하기도 하였고, 익숙하지 않은 방법이라 어려웠다. 그리고 군대 전역 후 복학하여서 이러한 프로젝트를 아주 오랜만에 코딩을 해보았는데 많은 어려움을 겪었다. 하지만 팀원들의 도움을 많이 받아 예전에 알았는데 까먹었던 부분도 많이 생각이 나고 전보다 많이 코딩실력이 향상된 것 같아서 기분이 좋았다. 다음 프로젝트도 열심히 하여 잘 마무리 해야겠다.

황성영 (팀원): 이번 프로젝트를 진행하면서 처음으로 팀 프로젝트를 진행하게 되었는데, 걱정이 되었지만 그래도 잘 마무리 되어서 다행이였다. 확실히 혼자 하는 것 보다 더 문제가 발생 하였을 때, 서로 얘기를 하며 방안을 찾을 수 있다는 점과 역할 분담을 하여서 진행을 할 수 있다는 점이 좋았다. 하지만 역할 분담을 하여서 본인이 비중이 덜 한 곳은 잘 모를 수 있다는게 단점인 것 같기도 하다. 프로젝트 관해서는 기존에 배웠던 링크드 리스트와 큐. 그리고 이번 학기 부터 배운 트리 3개의 자료구조를 합쳐서 단어장을 구성하였는데 먼저 프로그램의 틀을 미리 조교님들이 만들어놔서 편하기도 하였고, 코드를 보면서 너무 깔끔하고 잘 정리 되어있고, 필요한 것만 잘 정해져 있는걸 보고 아직 내가 많이 코딩실력이 많이 부족하다는 것과 배울 점이 많다는걸 느꼈다. 그리고 문제점도 많이 있었는데. 먼저 BST구조가 2개로 연결 되어 있으니깐 단순히 생각하기에는 연결만 잘 되있으면 문제없을 거라 생각 했지만 실제로 프린트나 순회를 하는 부분에서 많은 문제가 있었다. 트리가 구성되어 있어도 word가 없는 노드도 있을텐데 이 상황에서 프린

트 문제가 있었다. 샘플이라는 임시 노드들을 다 추가하여 해결했지만 더 좋은 방법이 있을 수 있겠지만 시간상 그렇게 하진 않았다. 처음 팀별 과제 였지만 많이 배우고 좋게 끝나서 다행이였다.