1차 프로젝트 보고서

Data Structure Lab. Project #1

제출일자: 2016년 10월 07일 (금)

학 과: 컴퓨터공학과

담당교수: 이기훈 교수님

학 번: 2012722089

성 명: 신기환

학 번: 2012722086

성 명: 정병찬

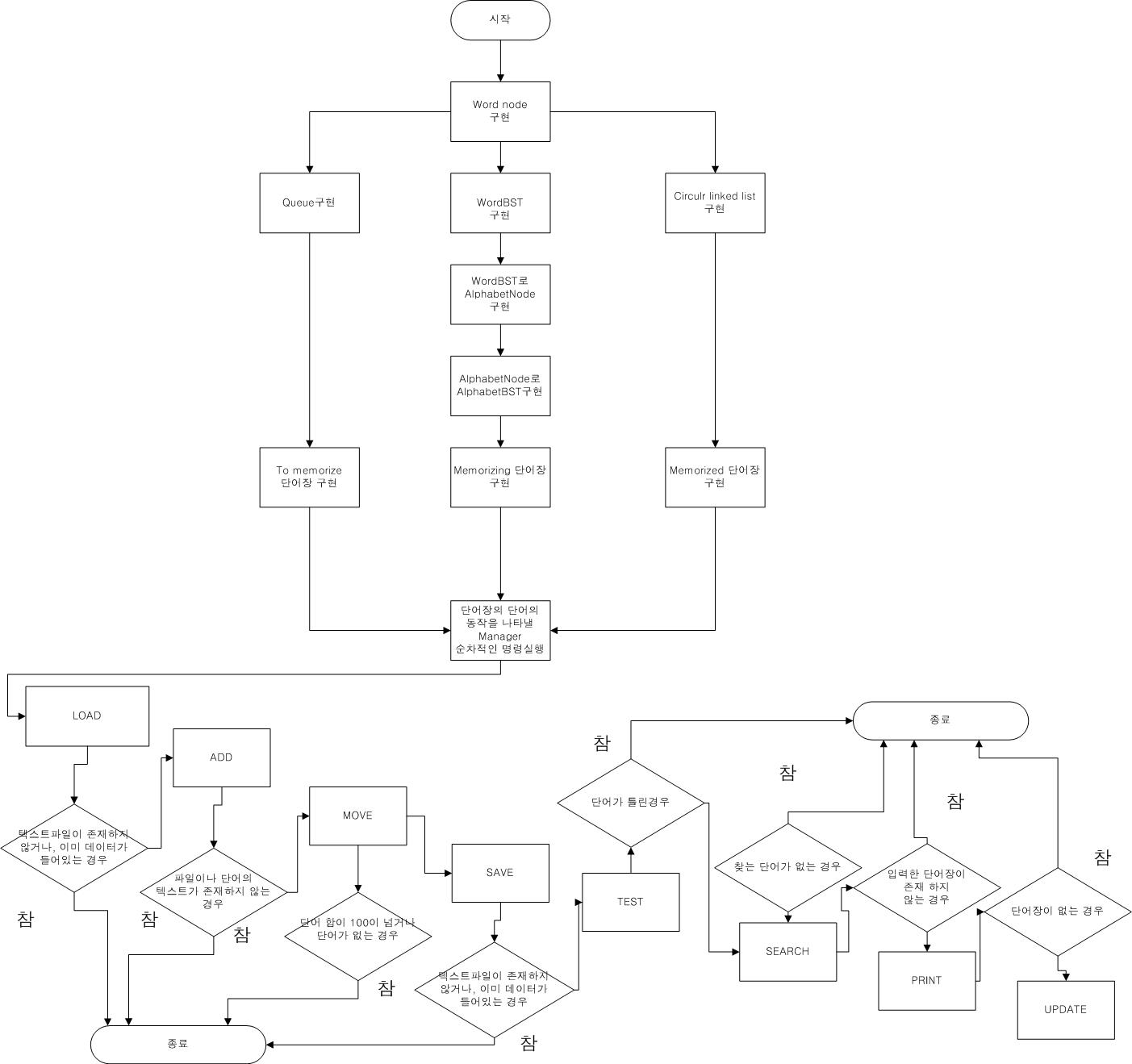
학 번: 2013722017

성 명: 정민호

Introduction

이번 프로젝트는 영어 단어장 프로그램을 구현하는 것이 목적입니다. 여기에는 총 3개의 단어장이 있는데 우선 앞으로 외워야 할 영어 단어와 뜻이 있는 “TO\_MEMORIZE”, 현재 외우고 있는 단어장인 “MEMORIZING”, 그리고 이미 외운 “MEMORIZED”로 구성되어있습니다. “TO\_MEMORIZE”는 queue로, “MEMORIZING”은 이진 탐색 트리로, “MEMORIZED”는 환형 연결 리스트로 구현합니다. 각 자료구조의 단어 노드에는 영어 단어와 한글 뜻이 데이터로 존재합니다. 우선 “TO\_MEMORIZE” queue는 새 단어가 들어올 때 push를 하고 “MEMORIZING”으로 옮길 때 pop동작을 합니다. “command.txt”을 통해 단어들이 추가되면 처음엔 “TO\_MEMORIZE”에 모두 들어가고 이후에 MOVE 명령어를 통해서 “MEMORIZING”으로 옮길 수 있습니다. 이때 “MEMORIZING”에는 최대 100개의 단어가 존재할 수 있습니다. “MEMORIZING”에는 알파벳 BST와 단어 BST가 존재하는데, 알파벳 BST는 알파벳 노드로 이루어져있고 각 노드에는 단어 BST를 가지고 있습니다. 그리고 TEST 명령어를 통해 단어를 외웠다면 “MEMORIZED”로 옮깁니다. “MEMORIZED”는 환형 연결 리스트이고 처음 단어가 들어오면 Head pointer가 그 단어를 가리킵니다. 그리고 그 단어는 Head pointer를 가리킵니다. 만약 이미 단어가 존재한다면 새로 들어온 단어를 Head pointer가 가리키는 노드와 그 뒤의 노드 사이에 연결해줍니다.

Flow chart

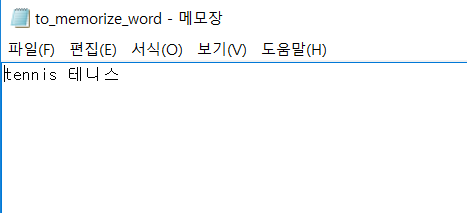
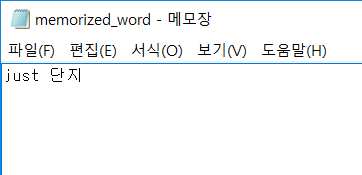
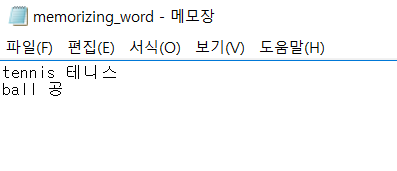


Algorithm

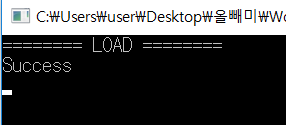
|  |  |
| --- | --- |
| LOAD | 우선 “to\_memorize\_word” 텍스트 파일을 읽어옵니다. 그리고 한 줄씩 getline으로 읽어와서 임시 배열에 저장하는데, 파일에 단어와 뜻 사이에 공백이 있으므로 토큰으로 끊어서 단어와 뜻을 따로 분리해서 word node의 단어와 뜻에 저장해줍니다. 그리고 노드를 Queue에 push해줍니다. 이것을 파일의 끝을 도달할 때까지 반복해서 Queue를 채웁니다. 그 다음에는 “memorizing\_word” 텍스트 파일을 열고 “to\_memorize\_word”에서 한 것처럼 반복해서 BST에 노드를 채웁니다. “memorized\_word” 파일을 열어서도 똑같이 수행하여 환형 연결 리스트를 구성합니다. |
| ADD | ADD에서는 단어 정보를 읽어오기 위해서 “word.txt” 파일을 우선 읽어옵니다. 그리고 파일 안에 존재하는 모든 단어를 “TO\_MEMORIZE”에 저장하기 위해서 파일의 끝을 도달할 때까지 “LOAD”에서 한 것처럼 한 줄씩 읽어와서 토큰으로 단어와 뜻을 잘라줍니다. 하지만 “word.txt”에서는 단어와 뜻이 tab을 두고 나뉘어져 있기 때문에 토큰으로 나눌 때 스페이스 한 칸 대신에 tab(\t)을 넣어줘야 합니다. 그리고 노드를 Queue에 push하기 전에 단어가 Queue에 이미 존재하는지 판별해서 존재 한다면 continue문을 통해서 다시 반복문의 처음으로 돌아가 다음 줄의 단어를 읽어옵니다. |
| MOVE | “command.txt”에 입력된 수만큼 “TO\_MEMORIZE”의 단어들을 “MEMORIZING”으로 옮겨야 하기 때문에 우선 문자열 형태로 읽어온 숫자를 정수형으로 바꿔줘야 합니다. 이 때 입력받은 수와 “MEMORIZING”의 단어 수의 합이 100을 넘으면 에러를 출력합니다. 그리고 Queue에 존재하는 단어의 수보다 더 많은 수를 입력 받았을 때에도 에러를 출력하도록 해줬습니다. 옮겨줄 때에는 Queue에서 pop을 해서 임시 노드에 저장해 줬다가 그것을 다시 “MEMORIZING”에 넣어주기 위해 BST에 그 노드를 삽입해주었습니다. |
| SAVE | circular linked list의 save는 Head부터 파일들을 텍스트 파일에 단어와 뜻을 집어 넣습니다. Queue도 Head부터 차례대로 다음으로 넘어가면서 집어 넣습니다. 더 이상 노드가 존재하지 않을 때까지 계속합니다. BST는 AlphabetBST에 노드에 저장된 모든 노드를 뿌리부터 탐색합니다. 탐색할 노드를 찾고 AlphabetNode에 있는 BST에서 파일에 저장되어 있는 모든 값을 텍스트 파일에 옮깁니다. |
| TEST | 우선 입력된 단어와 뜻을 다 읽어와서 단어가 “MEMORIZING”에 존재하는지를 판단하기 위해서, 단어를 BST에서 search를 합니다. 만약 단어가 없다면 에러를 출력합니다. 단어가 존재한다면 입력된 단어의 뜻과 BST에서 찾은 단어가 가지고 있는 뜻이 동일한지 “strcmp”함수로 비교합니다. 만약 뜻이 같지 않다면 에러를 출력합니다. 만약 뜻이 동일하면 단어를 외웠다는 것이 되기 때문에 “MEMORIZED”로 옮겨주기 위해서 우선 그 노드를 환형 연결 리스트에 삽입해줍니다. 그리고 “MEMORIZING”에서 없애주기 위해 BST에서 그 노드를 delete함수를 통해 삭제합니다. |
| SEARCH | “TO\_MEMORIZE”, “MEMORIZING” 또는 “MEMORIZED”에 단어가 존재하는지를 판단해주기 위해서 각각 “queue”, “bst”, “cll”의 search함수를 통해서 입력받은 단어를 가진 노드를 찾습니다. 노드를 찾으면 단어와 뜻을 출력해줍니다. 만약 3곳 모두에서 찾지 못한다면 에러를 출력합니다. |
| PRINT | Queue의 print는 노드를 하나 만들어서 해드부터 차례대로 Queue가 끝날 때까지 차례대로 출력을 해줍니다. BST 에서는 command 에서 입력 받은 순회로 입력을 받아서 출력을 합니다. circular linked list에서는 Head부터 차례대로 콘솔 창에 단어와 의미를 입력합니다. |
| UPDATE | “SEARCH”에서 수행한 것처럼 “TO\_MEMORIZE”, “MEMORIZING” 또는 “MEMORIZED”에 단어가 있는지 찾습니다. 만약 찾으면 그 노드의 SetMean함수를 사용해 뜻을 바꿔줍니다. 예를 들어서 “MEMORIZING”에 업데이트해줘야 할 단어가 존재한다면 search함수를 통해 BST의 root에서부터 타면서 해당 단어를 가지고 있는 노드를 찾아서 그 노드의 뜻을 바꿔줍니다. |

Result Screen

LOAD

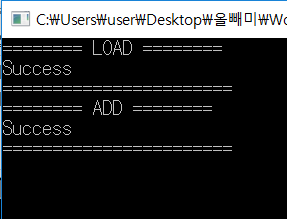
 

다음과 같이 처음 LOAD를 하기 전에 3개의 텍스트 파일 있으면 3개의 단어장의 단어를 집어 넣을 수 있으므로 다음과 같이 LOAD가 성공합니다.



ADD

다음은 ADD입니다. Word.txt에 있는 단어를 to memorized파일에 넣어줍니다. ADD를 실행해서 성공한 결과 다음과 같이 됨을 알 수 있습니다.





처음에 단어장에 단어가 하나밖에 없었지만 pNext로 계속들어가면 단어가 연결되어 있음을 볼 수 있습니다.

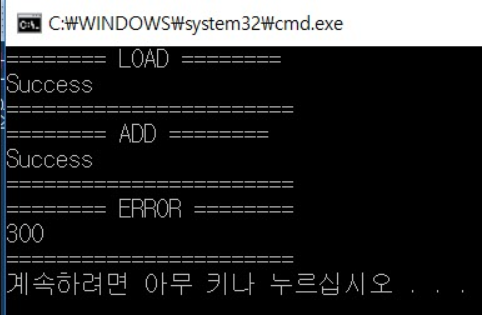
MOVE

그 다음에 설명할 동작은 MOVE입니다.

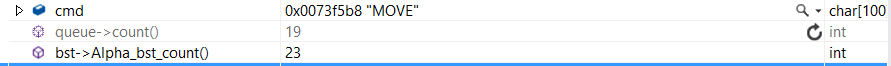
To memorize 에 있는 단어와 memorizing에 있는 단어의 합이 100이 넘어가면 에러가 뜹니다.

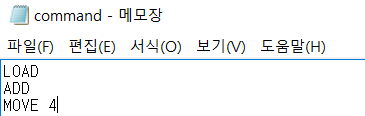


다음 조사식에 보듯이 queue의 단어와 bst의 노드 개수가 100이 넘어가 버리면 에러가 뜨는 것 입니다.

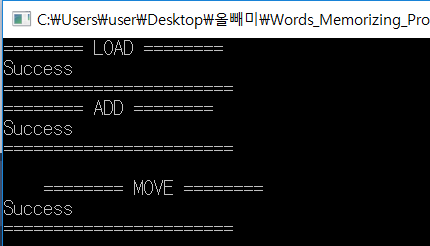


다음 과 같이 에러가 뜨게 됩니다.



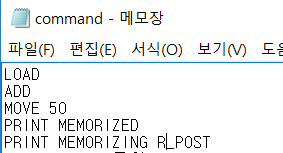


다음과 같이 to memorize와 memorizing의 합이 100이하이고 움직이는 단어의 수도 4개 이하이기 때문에 위의 경우는 성공하게 됩니다.

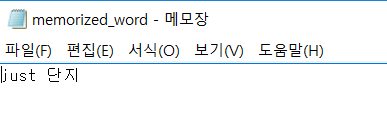


다음과 같이 성공한 것을 알 수 있습니다.

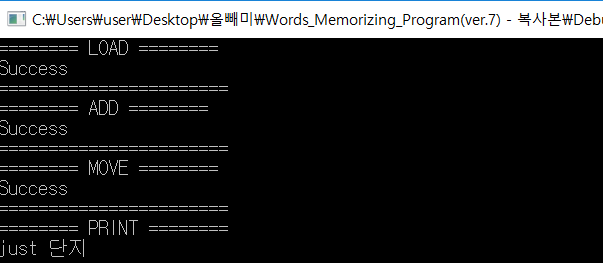
PRINT



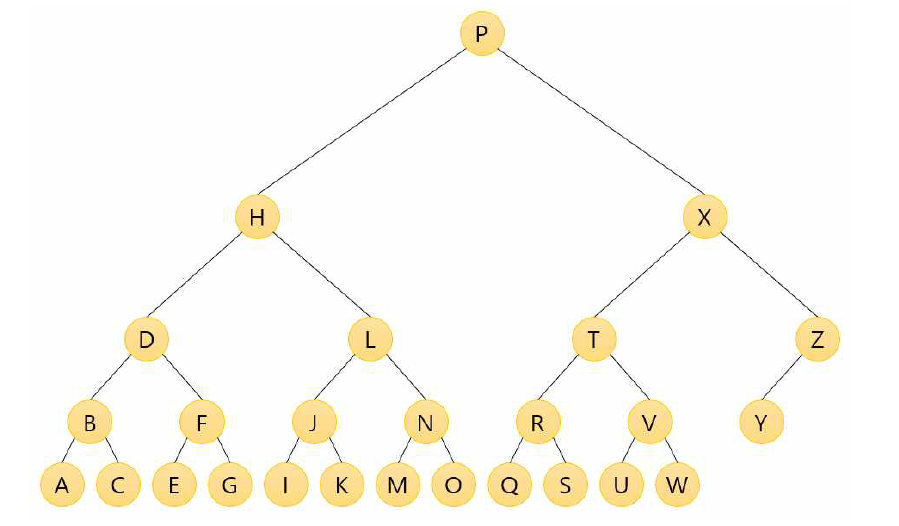
처음에 MOMORIZED를 출력합니다. MEMORIZED를 출력하면 다음과 같습니다.



단어가 단 하나가 저장되어 있으므로

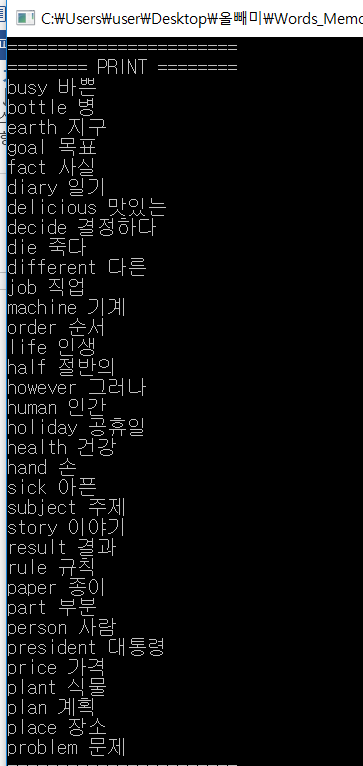


다음과 같이 출력이 되고, 그다음 PRINT로 MEMORIZING 즉 BST를 iterateor poset order 순으로 출력 이됩니다.

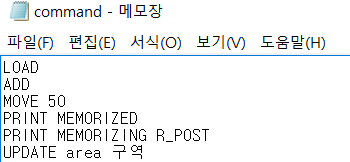


다음과 같은 AlphabetBST이므로 post order는 left->right->root 순으로 이므로 맨 왼쪽부터 찍게 됩니다. 다음 프린트와 같이 b->e->g->f->d->j->m->o->l->h->s->r->p순으로 잘 순회하는 것을 볼 수 있습니다.

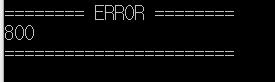
UPDATE



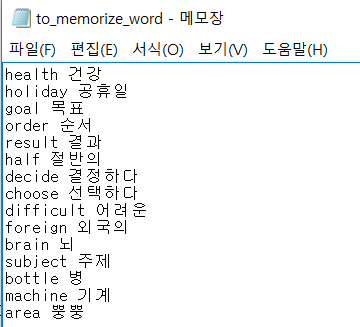
다음 명령어는 UPDATE입니다. 의미를 새로 바꿔 줄 단어를 찾이 못하면 다음과 같이 출력이 됩니다.



단어가 모든 단어장에 존재하지 않는 단어 입니다. 따라서



다음과 같은 ERROR를 출력하게 됩니다.



다음과 같이 area 뿡뿡 이라고 뜻이 잘못 된 단어를 to momorize에 저장 하였습니다.

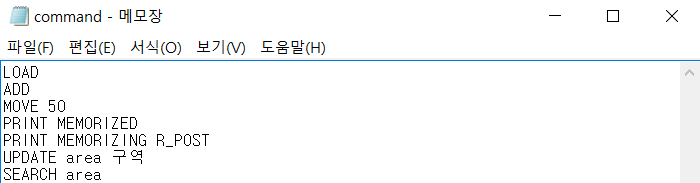
이를 다시 컴파일링 하면



다음과 같은 창이 뜨면서 고쳐지는 것을 알 수 있습니다.

SEARCH

그 다음은 SEARCH입니다.



아까 의미를 고쳐주었던 단어 area을 찾아 보겠습니다.



다음과 같이 SEARCH가 잘 이루어졌음을 잘 알 수 있습니다.

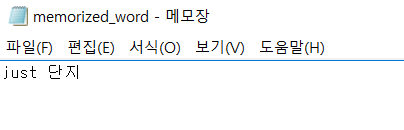
찾을 단어를 단어장에 지워주어서 없으면



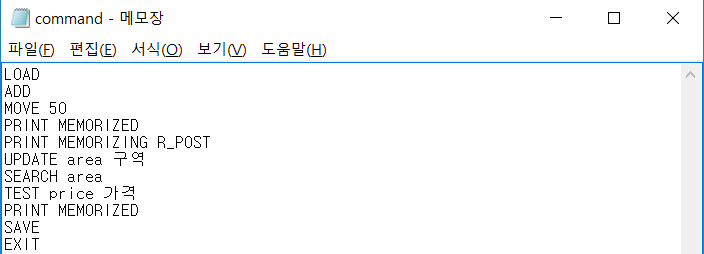
다음과 같은 에러가 뜸을 잘 알 수 있습니다.

이 다음은 TEST 함수 입니다.

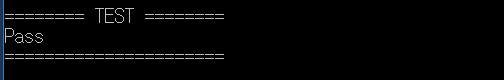




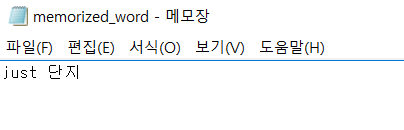
다음과 같이 memorized에 저장되어 있기 때문에 ERROR 발생한것을 볼 수 있습니다. Meomorizing에 있는 것을 테스트 하는 것이기 때문입니다.

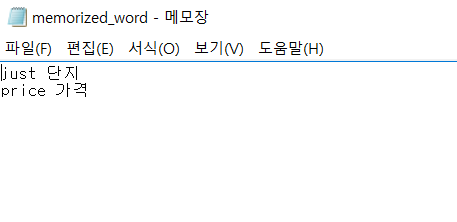


다음과 같이 TEST에 단어에 알맞은 뜻을 작성하면



가 떠서 잘 저장되고

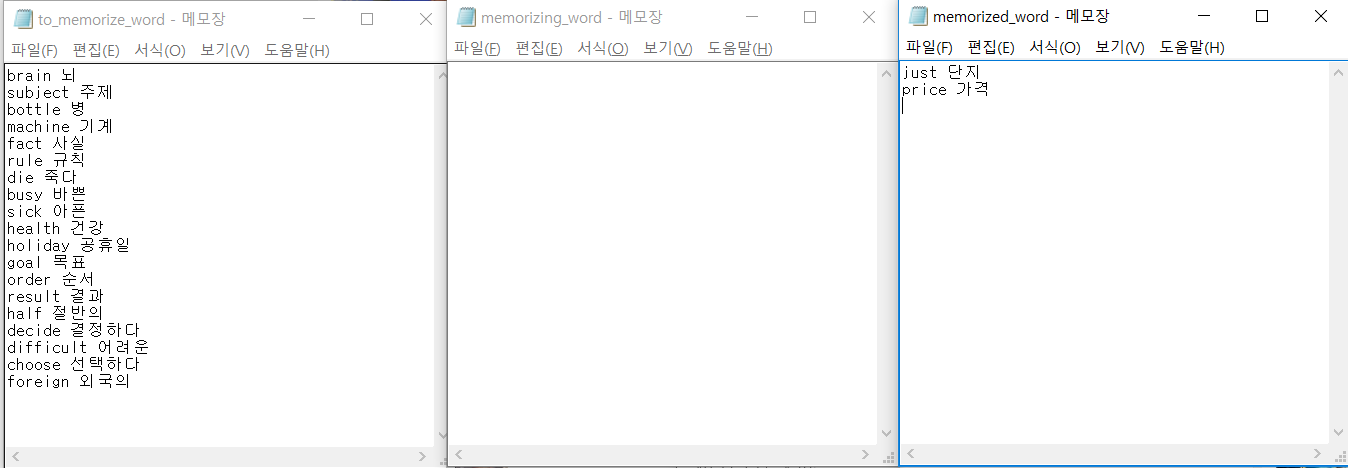




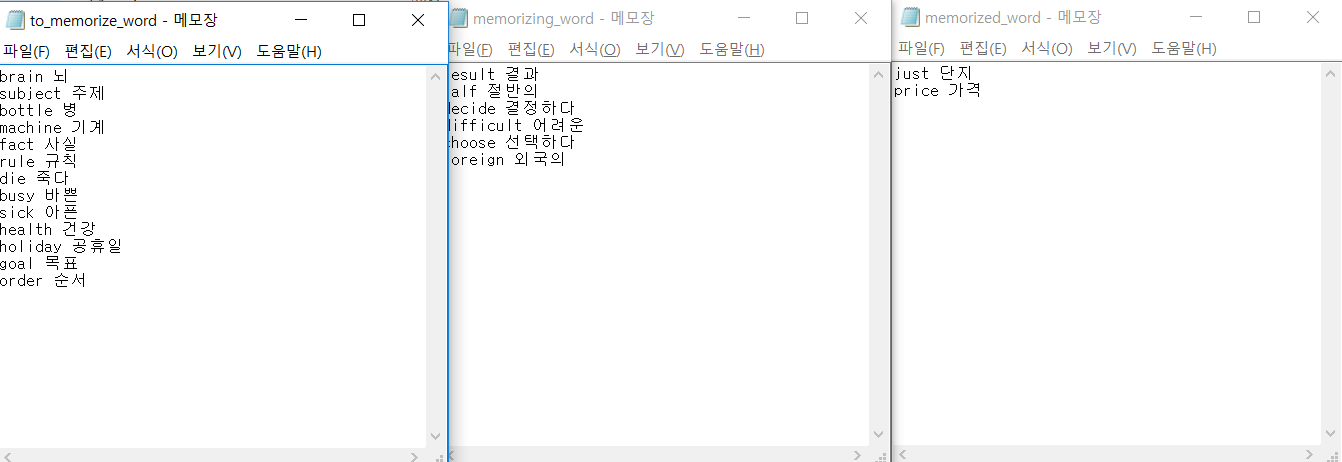
에 잘 추가 됨을 볼 수 있습니다.

SAVE

마지막으로 SAVE 함수 입니다. 각각 단어장에 있는 단어들을 텍스트 파일로 다시 출력을 하는 방식입니다.



다음과 같이 단어 장이 있는 SAVE함수를 작동하면



넘어가서 memorizing에 잘 들어감을 알 수 있습니다.

Consideration

* 정병찬

전에 배운 개념들을 이번 프로젝트에서 모두 다뤘기 때문에 Queue, BST, circular linked list를 구현하는 것을 더 익히고 그것들에 더 익숙해질 수 있었습니다. 또 이번에 구현한 AlphabetBST는 각 노드 안에 또다른 WordBST가 존재하는 이진 탐색 트리 안의 이진 탐색 트리의 구조였기 때문에 처음에 많이 헷갈렸습니다. 또 저는 팀 프로젝트여서 좋았던 점이 팀원 별로 각자 취약한 부분과 자신 있는 부분이 다르기 때문에 상호보완적인 관계를 가져서, 혼자서는 구현하기 어려운 프로그램을 같이 고민하고 여러 시도를 해서 문제를 풀 수 있었다고 생각합니다.

* 정민호

이번 프로젝트를 통해서 다양한 경우를 생각하고 짤 수 있어서 좋았습니다. 특히 BST쪽은 짜면서 BST에 대한 이해도를 높일 수 있었습니다. 그리고 팀으로 코딩하는 것은 처음이여서 낯설긴 했지만 매우 좋은 경험이였습니다. 다 같이 협동해서 하니깐 혼자서 할 때는 보이지 않았던 점이 보이고, 팀원들간에 끈끈한 결속이 생기고, 한명 한명 각각 짜는 것 보다는, 여러이서 같이 하니 윈윈 되고 시너지 효과가 생기는 것 같았습니다. 그래서 전 좋았는데 임의뢰 짜진 팀을 보면 어떤 팀은 한 팀원이 잠수를 타고 , 서로 어색한 사이에서 팀원을 해서 합도 안맞고 갈등이 있는 것도 종종 보았습니다. 마지막 프로젝트 하는 동안에도 안보이고 잠수를 하는 등 안 좋은 모습등이 보였습니다. DS 설계 두번 째 시간에 왔떤 해커톤 이야기를 해주셨던 선배님 이야기가 생각 났습니다. 코딩 실력도 중요하지만 정말 잘 맞는 팀원들과 합이 맞아서 하는 것도 정말 중요 하다는 것을 느꼈습니다. 앞으로도 자주 팀프로젝트로 진행하였으면 좋겠습니다.

* 신기환 – 프로그래밍을 배우기 시작한 후 처음으로 팀 프로젝트를 진행했습니다. 이번 과제를 통해 하나의 프로그램을 만들기 위해 여러가지 함수들을 다른 사람들이 만들어 다시 큰 하나의 프로그램으로 만들 수 있다는 부분에서 협업의 이로움과 큰 프로그램을 구현하는 효율적인 방법에 대해서 이해할 수 있었다. 하지만 문제점이 없던 건 아니었다. 우선 가장 큰 문제점은 역시 디버깅에서 찾을 수 있었다. 여러 팀원들이 작성한 코드들을 한번에 모아 다시 하나의 프로그램으로 사용했기 때문에 각각의 버그들을 잡아야 하지만 이것을 하나의 프로그램으로 만들었을 때 생기는 연결 문제들이 많이 발생하는 것을 느낄 수 있었다. 이런 문제를 해결함으로써 앞으로의 큰 프로젝트를 수행하여 내가 하나의 부분을 만들 때 모든 프로그램을 생각해여 코드를 작성해야 되며, 먼저 큰 그림을 생각하여 각 계층에 존재하는 알고리즘을 파악하는 것이 중요하다는 것을 느낄 수 있었다.