Data Structure Report

Project 1

Professor	이기훈 교수님
Department	Computer engineering
Student ID	
Name	
Class	설계 ( ) / 실습 ( )
Date	

Project Report

- Introdution
- Algorithm
- Flow Chart
- Result Screen
- Conclusion

1. Introdution
2. Flow Chart
   - Queue

   - Binary Search Tree
   - Circular Linked List
3. Algorithm
   - Queue

Push
pHead가 NULL일 경우 parameter인 node의 주소값을 pHead에 저장한다. pHead가 NULL이 아닐 경우 WordNode의 포인터형 변수 pTemp를 선언하여 pHead의 주소 값을 저장하고, pTemp를 Queue의 마지막까지 반복문을 사용하여 이동한다. 마지막 Node의 Next로 parameter인 node를 연결한다.

Pop
pHead의 Next Node가 NULL일 경우 WordNode의 포인터형 변수 pTemp를 선언하여 pHead의 주소값을 저장하고 pHead는 NULL로 초기화한 뒤 pHead의 메모리를 해제한다. 그 후 pTemp를 반환한다. 위 2가지 경우에 해당하지 않는 경우 WordNode의 포인터형 변수 pTemp를 선언하여 pHead의 주소값을 저장하고, pHead는 Next Node로 이동한다. pTemp의 Next Node는 NULL로 초기화하고 pTemp를 반환한다.

Search
WordNode의 포인터형 변수 pTemp를 선언하여 pHead의 주소값을 저장하고, pTemp를 반복문을 사용하여 마지막 Node까지 이동하면서 strcmp를 사용하여 현재 위치하고있는 Node와 입력받은 word를 비교하여 같을 경우 pTemp를 반환하고 pTemp가 마지막 Node까지 이동하여 NULL이 될 경우 NULL을 반환한다.

Print
pHead가 NULL일 경우 0을 반환한다. Log파일을 만들기 위해 파일입출력을 사용한다. 반복문을 사용하여 Queue를 Head부터 마지막 Node까지 순회하면서 현재 Node의 word와 mean을 출력한다. Queue가 비어있을 경우 Error를 출력한다.

Save
pHead가 NULL일 경우 저장할 Node가 없으므로 1을 반환한다. “to_memorized_word.txt”파일에 저장하기 위해 파일입출력을 사용하고, Print함수와 같은 방법으로 Queue를 Head부터 마지막 Node까지 순회하면서 텍스트 파일에 현재 Node의 word와 mean을 출력한다.

- Binary Search Tree
  ① Alphabet BST

Insert
Root가 NULL일 경우 root에 parameter인 node를 저장한 뒤, 반환한다. 반복문의 조건에 true를 입력하여 무한 loop로 만든다. 현재 Node의 alphabet이 입력 받은 node의 alphabet보다 작을 경우 현재 Node의 Right Node가 NULL일 경우 Right Node에 입력 받은 node의 주소 값을 저장한다. Right Node가 NULL이 아닐 경우 현재 Node를 Right Node가 NULL이 될 때까지 이동한 뒤, Right Node로 입력 받은 node를 설정한다.. 현재 Node의 alphabet이 입력 받은 node의 alphabet보다 크거나 같을 경우 현재 node의 Left Node가 NULL일 경우 Left Node에 입력 받은 node의 주소 값을 저장한다. Left Node가 NULL이 아닐 경우 현재 Node를 Left Node가 NULL이 될 때까지 이동한 뒤, Left Node로 입력 받은 node를 설정한다.

Print
Log파일을 만들기 위해 파일입출력을 사용한다. 입력 받은 order의 값을 “R_PRE”, “I_PRE”, “R_IN”, “I_IN”, “R_POST”, “I_POST”, “I_LEVEL”과 비교하여 같은 순회 방식의 함수를 호출한다. Order가 위의 순회 방식과 다를 경우 에러를 출력한다.

Search
Root가 NULL일 경우 NULL을 반환한다. R_Search함수를 호출하여 함수에서 반환된 값을 반환한다.

Save
Iterative Preorder 순회 방식으로 Node를 Stack의 역할을 하는 Alphabet 포인터형의 배열에 저장한 뒤, 배열에 저장된 주소 값의 WordBST의 Save 파일을 호출한다.

Binary Search Tree 순회	
R_PRE	Reculsive Preorder 순회 방식으로 재귀 함수를 사용하는 방식이다. 현재 순회하는 node가 NULL일 경우 반환하는 탈출 조건을 만든다. 현재 순회하는 node의 WordBST의 R_PRE함수를 호출한다. R_PRE함수의 parameter에 node의 Left Node를 입력하여 호출한다. R_PRE함수의 parameter에 node의 Right Node를 입력하여 호출한다.
I_PRE	Iterative Preorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 첫 번째 인자로 입력한다. 배열의 마지막 인자의 WordBST의 I_PRE함수를 호출하고 Right Node가 NULL이 아닐 경우 배열의 마지막에 Right Node를 저장한다. Left Node가 NULL이 아닐 경우 배열의 마지막에 Left Node를 저장한다.
R_IN	Reculsive Inorder 순회 방식으로 재귀 함수를 사용하는 방식이다. 현재 순회하는 node가 NULL일 경우 반환하는 탈출 조건을 만든다. R_IN함수의 parameter에 node의 Left Node를 입력하여 호출한다. 현재 순회하는 node의 WordBST의 R_IN함수를 호출한다. R_IN함수의 parameter에 node의 Right Node를 입력하여 호출한다.
I_IN	Iterative Inorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 0번째 인자로 입력한다. BST의 Left Leaf Node로 이동하면서 배열에 주소 값을 저장하고 배열의 첫 번째 인자가 NULL일 경우 반복을 중단한다. pTemp에 배열의 마지막 주소 값을 저장하고 Node의 WordBST의 I_IN 함수를 호출한다. pTemp에 배열의 마지막 주소의 Right Node를 저장하고 배열의 마지막을 NULL로 초기화하고 다시 pTemp의 주소값을 저장한다. 위의 과정이 모두 끝났을대 stack의 값이 -1이면 반복문을 탈출한다.
R_POST	Reculsive Postorder 순회 방식으로 재귀 함수를 사용하는 방식이다. 현재 순회하는 node가 NULL일 경우 반환하는 탈출 조건을 만든다. R_POST함수의 parameter에 node의 Left Node를 입력하여 호출한다. R_POST함수의 parameter에 node의 Right Node를 입력하여 호출한다. 현재 순회하는 node의 WordBST의 R_POST함수를 호출한다.
I_POST	Iterative Postorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 첫 번째 인자로 입력한다. Root의 Right Node가 NULL이 아닐 경우 배열의 두 번째 인자에 root의 Right Node의 주소 값을 저장한다. Left Node의 주소 값을 저장하기 위해 AlphabetNode 포인터형 배열 left를

	선언하고 초기화한다. 반복문을 사용하여 pTemp에 배열의 마지막 주소 값을 저장하고, pTemp의 Right Node가 NULL이 아닐 경우 배열에 pTemp의 Right Node의 주소 값을 저장한다. Node의 Left Node가 NULL이 아닐 경우 left 배열에 주소 값을 저장한다. pTemp의 Right Node가 NULL일 경우 left 배열의 마지막 인자 값을 arr 배열의 마지막에 저장한다. arr배열의 마지막 인자의 Left Node, Right Node가 NULL이 아니거나, arr 배열의 마지막 인자가 leaf Node이며 left배열이 NULL이 아닐 경우 계속 반복한다. 위와 같은 방식을 root의 Left Node로 이동하여 순회한다. 반복문을 사용하여 arr배열의 뒤에서 처음으로 이동하면서 WordBST의 I_POST함수를 호출한다.
I_LEVEL	Iterative Levelorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 첫 번째 인자로 입력한다. 배열의 인자의 주소 값이 NULL이 아닐 경우 주소 값의 WordBST의 I_LEVEL함수를 호출한다. pTemp의 Left Node가 NULL이 아닐 경우 배열의 마지막에 저장한다. pTemp의 Right Node가 NULL이 아닐 경우 배열의 마지막에 주소 값을 저장한다. I의 값과 stack의 값이 같을 경우 함수를 탈출한다.

R_Search
재귀 함수를 사용한 Search함수이다. Alphabet이 0일 경우 NULL을 반환한다. 현재 Node의 Left Node가 NULL이고 Right Node가 NULL이고 현재 node의 alphabet이 입력 받은 alphabet과 같지 않을 경우 NULL을 반환한다. 현재 Node의 alphabet이 입력 받은 alphabet과 같을 경우 현재 Node를 반환한다. 현재 Node의 alphabet이 입력 받은 alphabet보다 클 경우 현재 Node를 Left Node로 이동하여 R_Search 함수를 호출한다. 현재 Node의 alphabet이 입력 받은 alphabet보다 작을 경우 현재 Node를 Right Node로 이동하여 R_Search 함수를 호출한다.

② Word BST

Insert
Root가 NULL일 경우 root에 parameter인 node를 저장한 뒤, 반환한다. 반복문의 조건에 true를 입력하여 무한 loop로 만든다. 현재 Node의 word가 입력 받은 node의 word보다 작을 경우 현재 Node의 Right

Node가 NULL일 경우 Right Node에 입력 받은 node의 주소 값을 저장한다. Right Node가 NULL이 아닐 경우 현재 Node를 Right Node가 NULL이 될 때까지 이동한 뒤, Right Node로 입력 받은 node를 설정한다.. 현재 Node의 word가 입력 받은 node의 word보다 크거나 같을 경우 현재 node의 Left Node가 NULL일 경우 Left Node에 입력 받은 node의 주소 값을 저장한다. Left Node가 NULL이 아닐 경우 현재 Node를 Left Node가 NULL이 될 때까지 이동한 뒤, Left Node로 입력 받은 node를 설정한다.

Delete
Root가 NULL일 경우 NULL을 반환한다. R_Delete함수를 호출하여 함수에서 반환된 값을 반환한다.

Search
Root가 NULL일 경우 NULL을 반환한다. R_Search함수를 호출하여 함수에서 반환된 값을 반환한다.

Print
Log파일을 만들기 위해 파일입출력을 사용한다. 입력 받은 order의 값을 “R_PRE”, “I_PRE”, “R_IN”, “I_IN”, “R_POST”, “I_POST”, “I_LEVEL”과 비교하여 같은 순회 방식의 함수를 호출한다. Order가 위의 순회 방식과 다를 경우 에러를 출력한다.

Save
Root가 NULL일 때 저장할 값이 없으므로 true를 반환한다. Memorizing_word txt file에 출력하기 위해 파일입출력을 사용한다. Iterative Preorder 순회 방식으로 WordBST의 Node들의 word와 mean을 텍스트 파일에 출력한다.

Binary Search Tree 순회	
R_PRE	Reculsive Preorder 순회 방식으로 재귀 함수를 사용하는 방식이다. 현재 순회하는 node가 NULL일 경우 반환하는 탈출 조건을 만든다. 현재 순회하는 node의 word와 mean을 출력한다. R_PRE함수의 parameter에 node의 Left Node를 입력하여 호출한다. R_PRE함수의 parameter에 node의 Right Node를 입력하여 호출한다.
I_PRE	Iterative Preorder 순회 방식으로 반복문을 사용하는 방식이다.

	Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 첫 번째 인자로 입력한다. 배열의 마지막 인자의 word와 mean을 출력한다. Right Node가 NULL이 아닐 경우 배열의 마지막에 Right Node를 저장한다. Left Node가 NULL이 아닐 경우 배열의 마지막에 Left Node를 저장한다.
R_IN	Reculsive Inorder 순회 방식으로 재귀 함수를 사용하는 방식이다. 현재 순회하는 node가 NULL일 경우 반환하는 탈출 조건을 만든다. R_IN함수의 parameter에 node의 Left Node를 입력하여 호출한다. 현재 순회하는 node의 word와 mean을 출력한다. R_IN함수의 parameter에 node의 Right Node를 입력하여 호출한다.
I_IN	Iterative Inorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 0번째 인자로 입력한다. BST의 Left Leaf Node로 이동하면서 배열에 주소 값을 저장하고 배열의 첫 번째 인자가 NULL일 경우 반복을 중단한다. pTemp에 배열의 마지막 주소 값을 저장하고 Node의 word와 mean을 출력한다. pTemp에 배열의 마지막 주소의 Right Node를 저장하고 배열의 마지막을 NULL로 초기화하고 다시 pTemp의 주소값을 저장한다. 위의 과정이 모두 끝났을대 stack의 값이 -1이면 반복문을 탈출한다.
R_POST	Reculsive Postorder 순회 방식으로 재귀 함수를 사용하는 방식이다. 현재 순회하는 node가 NULL일 경우 반환하는 탈출 조건을 만든다. R_POST함수의 parameter에 node의 Left Node를 입력하여 호출한다. R_POST함수의 parameter에 node의 Right Node를 입력하여 호출한다. 현재 순회하는 node의 word와 mean을 출력한다.
I_POST	Iterative Postorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 첫 번째 인자로 입력한다. Root의 Right Node가 NULL이 아닐 경우 배열의 두 번째 인자에 root의 Right Node의 주소 값을 저장한다. Left Node의 주소 값을 저장하기 위해 AlphabetNode 포인터형 배열 left를 선언하고 초기화한다. 반복문을 사용하여 pTemp에 배열의 마지막 주소 값을 저장하고, pTemp의 Right Node가 NULL이 아닐 경우 배열에 pTemp의 Right Node의 주소 값을 저장한다. Node의 Left Node가 NULL이 아닐 경우 left 배열에 주소 값을 저장한다. pTemp의 Right Node가 NULL일 경우 left 배열의 마지막 인자 값을 arr 배열의 마지막에 저장한다. arr배열의 마지막 인자의 Left Node, Right

	Node가 NULL이 아니거나, arr 배열의 마지막 인자가 leaf Node이며 left배열이 NULL이 아닐 경우 계속 반복한다. 위와 같은 방식을 root의 Left Node로 이동하여 순회한다. 반복문을 사용하여 arr배열의 뒤에서 처음으로 이동하면서 node의 word와 mean을 출력한다.
I_LEVEL	Iterative Levelorder 순회 방식으로 반복문을 사용하는 방식이다. Stack의 역할을 하는 Alphabet 포인터 형 배열을 선언하고 root를 배열의 첫 번째 인자로 입력한다. 배열의 인자의 주소 값이 NULL이 아닐 경우 주소 값의 word와 mean을 출력한다. pTemp의 Left Node가 NULL이 아닐 경우 배열의 마지막에 저장한다. pTemp의 Right Node가 NULL이 아닐 경우 배열의 마지막에 주소 값을 저장한다. I의 값과 stack의 값이 같을 경우 함수를 탈출한다.

R_Delete
R_Delete함수는 Reculsive 방식으로 설계한 Delete함수이다. Root를 삭제하는 경우에는 root가 양 쪽의 Node를 가질 경우, 한 쪽 Node만 가질 경우, 양 쪽 Node가 없을 경우가 있다. Root가 양 쪽 Node를 가질 때 삭제하는 경우는 root의 Left Node에서 가장 오른쪽 Node의 부모 Node로 이동하여 leaf Node를 pTemp_right에 저장한 뒤, pTemp의 Right Node를 NULL로 초기화한다. Root의 주소 값은 return_node에 저장한 뒤, root에 pTemp_right Node의 주소 값을 저장하고 return_node를 반환한다. Root가 한 쪽 Node만 가질 경우에는 Root를 pTemp를 이용하여 root값을 저장하고, 가지고 있는 방향의 Node로 이동하고 pTemp를 반환한다. Root가 leaf Node일 경우에는 Root를 pTemp를 이용하여 root값을 저장하고, root를 NULL로 초기화하고 pTemp를 반환한다. 두 번째 case로 현재 Node의 Left Node의 양 쪽 Node가 NULL이 아닐 경우이다. 현재 Node의 양 쪽 Node를 각각 pLeft, pRight에 저장한다. 현재 Node는 Left Node로 이동한 뒤 가장 오른쪽 Node의 부모 Node로 이동한다. 삭제하는 방법은 Root의 양 쪽 Node가 NULL이 아닐 경우와 같다. 세 번째 case는 현재 Node의 Left Node의 한 쪽 Node만 NULL이 아닐 경우이다. Root가 한 쪽 Node만 가질 경우 삭제하는 방법과 동일하다. 네 번째 case는 현재 Node의 Left Node의 양 쪽 Node가 NULL일 경우이다. Root의 양 쪽 Node가 NULL일 경우 삭제하는 방법과 동일하다. 다섯 번째 case는 현재 Node의 Right Node의 양 쪽 Node가 NULL일 경우이다. Root의 양 쪽 Node가 NULL이 아닐 경우와 같다.

여섯 번째 case는 현재 Node의 Right Node의 한 쪽 Node만 NULL일 경우이다.
Root가 한 쪽 Node만 가질 경우 삭제하는 방법과 동일하다.
일곱 번째 case는 현재 Node의 Right Node의 양 쪽 Node가 NULL일 경우이다. 양 쪽 Node가 NULL일 경우 삭제하는 방법과 동일하다.
현재 Node의 word가 입력 받은 word보다 작을 경우 현재 Node를 Right Node로 이동한 뒤 R_Delete 함수를 호출한다.
현재 Node의 word가 입력 받은 word보다 클 경우 현재 Node를 Left Node로 이동한 뒤 R_Delete 함수를 호출한다.

R_Search
재귀 함수를 사용한 Search함수이다. Word가 0일 경우 NULL을 반환한다. 현재 Node의 Left Node가 NULL이고 Right Node가 NULL이고 현재 node의 word가 입력 받은 word과 같지 않을 경우 NULL을 반환한다. 현재 Node의 word가 입력 받은 word와 같을 경우 현재 Node를 반환한다. 현재 Node의 word가 입력 받은 word보다 클 경우 현재 Node를 Left Node로 이동하여 R_Search 함수를 호출한다. 현재 Node의 word가 입력 받은 word보다 작을 경우 현재 Node를 Right Node로 이동하여 R_Search 함수를 호출한다.

- Circular Linked List

Insert
pHead가 NULL일 경우 pHead에 입력 받은 node를 저장하고 pHead Next Node로 pHead를 저장한다. pHead의 Next Node가 pHead일 경우 pHead의 Next Node에 입력 받은 node를 저장하고 Next Node의 Next Node로 pHead를 저장한다. 위의 두 조건에 부합하지 않을 경우 pTemp에 pHead를 저장하고 Circular Linked List의 마지막 Node로 이동한 뒤, pTemp의 Next Node로 입력 받은 node를 저장하고 pTemp의 Next Node의 Next Node로 pHead를 저장한다.

Search
pHead가 NULL일 경우 0을 반환한다. pTemp에 pHead를 저장하고, pTemp의 Next Node가 pHead일 경우 pTemp의 Next Node의 word와 입력 받은 word를 비교하여 같을 경우 pTemp를 반환하고, pTemp의 Next Node가 pHead가 아닐 경우 pTemp를 Circular Linked List의 tail node까지

이동하며 pTemp의 word와 입력 받은 word가 같을 경우 pTemp를 반환한다. pTemp의 Next Node가 pHead일 경우 NULL을 반환한다.

Print
pHead가 NULL일 경우 0을 반환한다. pTemp에 pHead의 값을 저장한다. pTemp의 Next Node가 pHead가 될 때까지 Next Node로 이동하면서 pTemp의 word와 mean을 출력한다. 모두 출력하면 1을 반환한다.

Save
pHead가 NULL일 경우 1을 반환한다. Memorized_word 텍스트 파일에 출력하기 위해 파일 입출력을 사용해서 텍스트 파일을 여는데 파일을 열 때 삭제하고 다시 생성하기 위해 ios::trunc를 사용한다. pHead의 Next Node가 pHead일 경우 pTemp의 word와 mean을 출력한다. pHead의 Next Node가 pHead가 아닐 경우 pTemp를 마지막으로 이동하며 pTemp의 word와 mean을 출력한다.

- Manager

Run
Command.txt 파일을 라인단위로 읽어와서 ₩r을 기준으로 나누고 " "로 나눈 뒤, 명령어에 따라 조건을 나눠서 명령어가 일치할 경우 함수를 호출한다. 함수가 실행에 실패했을 경우 Error를 출력한다.

LOAD
to_memorize_word 텍스트 파일에서 단어와 의미를 읽어와서 strtok를 사용하여 분리하고 Queue 에 중복되지 않을 경우 Queue에 저장한다. Memorizing_word 텍스트 파일에서 단어와 의미를 읽어와서 strtok를 사용하여 분리하고 Binary Search Tree 에 중복되지 않을 경우 Binary Search Tree에 저장한다. Memorizied_word 텍스트 파일에서 단어와 의미를 읽어와서 strtok를 사용하여 분리하고 Circular Linked List 에 중복되지 않을 경우 Circular Linked List에 저장한다.

ADD
Word 텍스트 파일에서 단어와 의미를 읽어와서 strtok를 사용하여 단어와 의미를 분리하고 Queue와 Binary Search Tree, Circular Linked List에 중복되지 않을 경우 Queue에 저장한다.

MOVE
Queue에서 pop된 Node의 단어의 첫 번째 알파벳을 Alphabet BST에서 Search함수를 호출하여 반환된 Node의 WordBST에 저장한다. 입력할 때 bst의 WordCnt변수를 1 증가시킨다. WordCnt가 100이 넘을 경우 0을 반환한다.

SAVE
Queue와 Binary Search Tree, Circular Linked List의 Save함수를 호출하여 세 함수의 반환 값이 모두 1일 경우 1을 반환하고, 그렇지 않을 경우 0을 반환한다.

TEST
입력 받은 단어를 모두 소문자로 변환한 뒤, bst의 Search함수를 호출한다. 반환된 Node가 NULL이 아닐 경우 입력 받은 단어의 의미가 반환된 Node의 의미와 같을 경우 bst에서 해당 단어를 삭제하고 CLL에 저장한다.

SEARCH
Queue의 Search함수를 호출하여 반환된 Node가 NULL이 아닐 경우 반환된 Node의 단어와 의미를 출력한다. NULL일 경우 bst의 Search함수를 호출한다. 반환된 Node가 NULL이 아닐 경우 반환된 Node의 단어와 의미를 출력한다. NULL일 경우 CLL의 Search함수를 호출한다. 반환된 Node가 NULL이 아닐 경우 반환된 Node의 단어와 의미를 출력한다. NULL일 경우 단어가 저장되있지않음으로 0을 반환하여 Error를 출력한다.

PRINT
입력 받은 명령어에 따라서 “TO_MEMORIZE”일 경우 Queue의 Print함수를 호출한다. “MEMORIZING”일 경우 입력 받은 order에 따라 bst에 알맞을 트리순회방식의 Print 함수를 호출한다. “MEMORIZED”일 경우 Circular Linked List의 Print함수를 호출한다.

UPDATE
Search함수와 마찬가지로 Queue의 Search함수를 호출하고 반환된 값이 NULL이 아닐 경우 반환된 Node의 의미를 변경하고 출력한다. NULL일 경우 bst의 Search함수를 호출하고 반환된 값이 NULL이 아닐 경우 반환된 Node의 의미를 변경하고 출력한다. NULL일 경우 Circular Linked List의 Search함수를 호출한다. 반환된 Node가 NULL이 아닐 경우 반환된 Node의 의미를 변경하고 출력한다. NULL일 경우 Error를 출력한다

EXIT
run함수를 탈출하여 프로그램을 종료한다.

4. Result Screen
5. Conclusion