Object-Oriented Programming Report

Assignment 3-1

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2020202031
Name	Jaehyun Kim

Class (Design / Laboratory)	1 / B
Submission Date	2023. 05. 05

Program 1

❑ 문제 설명

공백으로 구분되는 단어들을 여러 개 입력 받고, Linked list 를 생성합니다.

이렇게 생성된 리스트 2 개 전체에서 단어들을 알파벳 오름차순으로 병합합니다. 이 때 알파벳 대소문자는 구분하지 않고 병합한 후 출력하는 문제입니다.

두 문자열의 크기를 비교하는 것은 각 문자열의 문자를 char 형으로 저장하고 대문자와 소문자의 아스키코드 차이가 32 인 것을 활용하여 소문자로 변환한 뒤 비교했습니다.

정렬의 경우 bubble sort 를 이용했습니다.

마지막으로 Merge_List 의 경우 p1 과 p2 의 노드들의 문자열 크기를 비교하여 p3 에 이동시키는 방식을 사용했습니다.

❑ 결과 화면

```
Input>>
Input list 1: Well done is better than well said
Input List 2: Blaze with the fire that is never extinguished
better Blaze done extinguished fire is is never said than that the Well well with
```

Input list1 과 list2 가 알파벳 오름차순으로 잘 정렬된 모습이다. 단순히 아스키코드만 두고 보자면 b 가 B 보다 크지만 대소문자는 구분하지 않기 때문에 다음 문자인 e 와 l 중 알파벳 순서가 빠른 better 이 Blaze 보다 먼저 출력되는

모습이다. Well 과 well 과 같이 완전히 같은 두 단어의 경우 bubbleSort 에서 같은 단어는 switching 하지 않기 때문에 먼저 입력된 쪽이 먼저 출력되는 것이다.

❑ 고찰

두 단어 간의 알파벳 크기를 구분하기 위해 strcmp 를 사용하려 했으나, strcmp 는 같은 알파벳이라도 대소문자에 따라 아스키코드가 달라 대소문자를 구분하게 되는 문제점이 있었습니다. 그래서 각 문자열의 문자 하나씩을 char 형 변수에 저장하여 대문자일 경우 소문자로 바꾼 뒤 비교했습니다. bubbleSort 에서는 매개변수를 Node* pHead 로 했었는데, sort 과정에서 pHead 를 변경할 때, call by value 되어 main 함수에 존재하는 p3 는 처음 가리키던 노드를 그대로 가리키게 되어 매개변수를 Node** pHead 로 바꿔줌으로써 call by reference 로 변경해줬습니다.

Program 2

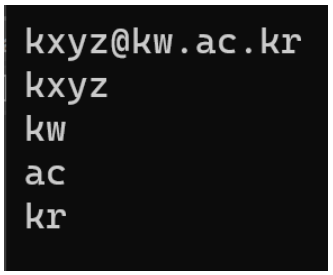
❑ 문제 설명

string.h 헤더파일에 있는 strtok 함수와 비슷한 기능의 my_strtok 함수를 구현하는 문제입니다.

함수의 매개변수로 전달되는 문자열은 처음 함수 호출할 때 한 번만 입력되기 때문에 문자열의 주소를 함수가 계속해서 기억할 수 있도록 static char 형 포인터로 문자열의 주소를 저장했습니다.

strtok 와 동일한 방법으로 구분문자를 NULL 로 채워주고 구분된 문자열의 주소를 반환하도록 코드를 작성했습니다.

❑ 결과 화면



구분기호인 '@', '.'에 따라 단어들이 각각 출력되는 것을 확인할 수 있습니다.

❑ 고찰

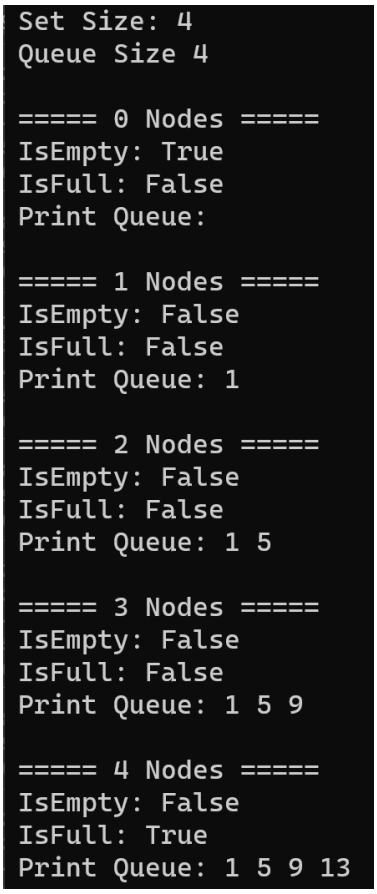
my_strtok 에서 문자열을 검사하기 위해 문자열의 인덱스를 저장할 idx 를 static int 형으로 선언했습니다. idx 를 1 씩 증가시키면서 입력 받은 문자열의 문자를 처음부터 하나씩 비교해가며 구분문자를 만나면 NULL 을 저장하고 idx 를 1 증가시킴으로써 idx 가 구분문자의 다음 인덱스를 참조하게 만들었습니다. 문자열 끝에 저장된 NULL 을 만나면 NULL 문자를 반환하도록 하기 위해 NULL 문자가 저장된 주소를 반환하도록 했는데, NULL 문자가 저장돼있을 뿐, 그 메모리 주소도 결국은 데이터이기 때문에 main 에서 무한루프가 실행되는 상황이 연출됐습니다. 따라서 문자열 끝에 저장된 NULL 을 만나면 return NULL; 하도록 코드를 수정했습니다.

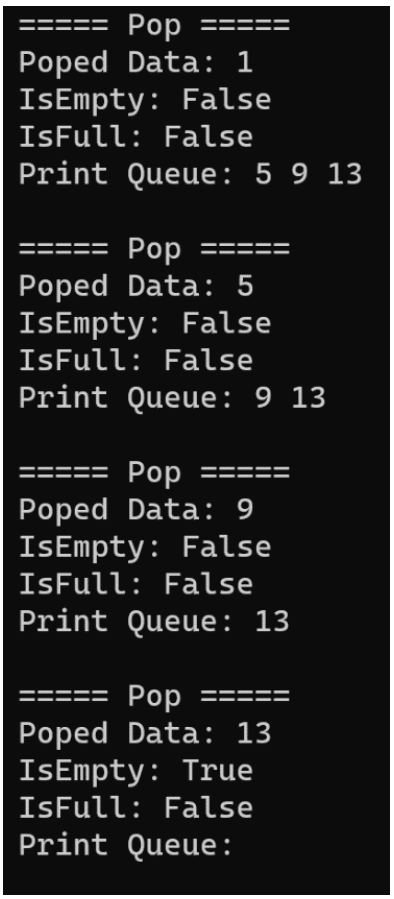
Program 3

❑ 문제 설명

Linked list 를 활용하여 Queue 를 구현하는 문제입니다. 문제에서 주어진 class 들의 선언들을 정의하고, 작성한 클래스 코드에 대한 테스트 코드를 main 함수에 작성합니다.

❑ 결과 화면





main 함수의 코드를 보면, Queue 객체를 생성하고 Queue 객체에 size 만큼 노드를 하나씩 차례로 Push 하고, 다시 노드를 하나씩 Pop 한다. 이 과정에서 queqe 가 비어있는지, 가득 차있는지 여부와 모든 노드의 데이터를 출력한다.

노드가 0 개일 때는 IsEmpty 가 true, IsFull 이 false 고, 노드가 존재하지 않으니 데이터가 출력되지 않는다.

노드가 1~3 개일 때는 IsEmpty, IsFull 모두 false 고, 추가된 노드 데이터도 잘 출력되는 모습이다.

노드가 4 개일 때는 IsEmpty 가 false, IsFull 이 true 고, 4 개의 노드 데이터가 잘 출력된다.

❑ 고찰

지금까지 Linked List 를 사용할 때 List 에 포함된 Node 의 개수를 멤버변수에 추가하지 않고 오로지 pHead 와 pTail 만을 사용하여 코드를 작성하곤 했는데 이번 과제를 통해 List 의 멤버변수에 Node 의 개수를 추가하니 class 메소드를 작성할 때 편리하다는 느낌을 받았고, 앞으로 이러한 방식을 사용하여 코드를 보다 더 간결하고 가독성 있게 작성해야겠다는 생각을 했습니다.

Program 4

❑ 문제 설명

Linked list 를 활용하여 Stack 을 구현하는 문제입니다. 문제에서 주어진 class 들의 선언들을 정의하고, 작성한 클래스 코드에 대한 테스트 코드를 main 함수에 작성합니다.

❑ 결과 화면

```
Set Size: 5
stack Size 5

===== 0 Nodes =====
IsEmpty: True
IsFull: False
Print stack:

===== 1 Nodes =====
IsEmpty: False
IsFull: False
Print stack: 1

===== 2 Nodes =====
IsEmpty: False
IsFull: False
Print stack: 5 1

===== 3 Nodes =====
IsEmpty: False
IsFull: False
Print stack: 9 5 1

===== 4 Nodes =====
IsEmpty: False
IsFull: False
Print stack: 13 9 5 1

===== 5 Nodes =====
IsEmpty: False
IsFull: True
Print stack: 17 13 9 5 1
```

```
===== Pop =====
Poped Data: 17
IsEmpty: False
IsFull: False
Print stack: 13 9 5 1

===== Pop =====
Poped Data: 13
IsEmpty: False
IsFull: False
Print stack: 9 5 1

===== Pop =====
Poped Data: 9
IsEmpty: False
IsFull: False
Print stack: 5 1

===== Pop =====
Poped Data: 5
IsEmpty: False
IsFull: False
Print stack: 1

===== Pop =====
Poped Data: 1
IsEmpty: True
IsFull: False
Print stack:
```

main 함수의 코드를 보면, Stack 객체를 생성하고 Stack 객체에 size 만큼 노드를 하나씩 차례로 Push 하고, 다시 노드를 하나씩 Pop 한다. 이 과정에서 stack 이 비어있는지, 가득 차있는지 여부와 모든 노드의 데이터를 출력한다.

노드가 0 개일 때는 IsEmpty 가 true, IsFull 이 false 고, 노드가 존재하지 않으니 데이터가 출력되지 않는다.

노드가 1~4 개일 때는 IsEmpty, IsFull 모두 false 고, 추가된 노드 데이터도 잘 출력되는 모습이다.

노드가 5 개일 때는 IsEmpty 가 false, IsFull 이 true 고, 5 개의 노드 데이터가 잘 출력된다.

❑ 고찰

Stack class 도 결국은 Linked List 를 활용하여 구현했기 때문에 Queue class 와 거의 똑같았습니다. 하지만 Stack 과 Queue 가 Node 를 추가하는 방식에 차이가 있어서 Queue class 에서 Push 와 Pop 메소드에 약간의 수정을 거쳐 구현했습니다. PrintStack 의 경우에 반목문의 조건을 i < m_NumElement 으로 설정했는데 이미 Full 인 리스트의 사이즈를 더 작은 사이즈로 설정하면 리스트의 사이즈보다 더 많은 데이터가 출력될 수 있으므로 i < m_NumElement && i < m_Size 로 수정하여 예외상황에 대처했습니다.