REPORT

보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과　　목 : 전자전기 프로그래밍실습
과 제 명 : Q2
담당교수 : 민 형 복
학　　과 : 전자전기공학부
학　　년 : 2
학　　번 : 2017311583
이　　름 : 정해진
제 출 일 : 2018.6.11

1. Introduction

[2]에 의하면, Linked List의 정의는 이렇다.

이것은 각 노드를 유용한 저장 공간에 그 위치에 상관없이 저장시키고, 각 노드의 관련성을 노드에 보관하여 1차원 배열 관계를 유지하도록 함으로써 중간 노드의 삽입, 제거를 손쉽게 할 수 있는 리스트로 각 노드는 링크(또는 포인터) 부분을 가지며, 그 노드와 관련 있는 다음 노드의 주소를 그 값으로 가진다. 다시 말하면 선형 리스트의 노드 배열이 어드레스와 일치하지 않고 기억 공간에 독립적으로 이루어진 리스트를 말한다.

[1]에 의하면, Doubly linked list는 다음과 같다.

• 수업에서 배웠던 linked list는 단일 linked list 입니다. 이 linked list는 머리부터 꼬리까지 연결됩니다.

• Doubly linked list의 연결은 머리에서 꼬리까지 그리고 꼬리에서 머리까지 양 방향으로 있습니다.

저번 실습 때 활용했던 linked list의 발전된 형태인 'Doubled Linked List"를 이용하여 주어진 조건 속에서 문제를 해결한다.

2. Problem Statement

① Describe what is the problem.

5개의 함수 코드를 작성한다.

• int headInsert(NODE **head, NODE **tail, int i_data);

• int headDelete(NODE **head, NODE **tail, int *i_data);

• int tailInsert(NODE **head, NODE **tail, int i_data);

• int tailDelete(NODE **head, NODE **tail, int *i_data);

• void deleteList(NODE **head, NODE **tail);

주의: 위의 4가지 int형 함수들은 데이터가 실제로 입력되거나 삭제되면 1을 return하고 어떤 이유로 데이터를 입력되지 않으면 0을 return한다.

두 개의 함수가 주어진다.
•printListfromHead(NODE *head);

•printListfromTail(NODE *tail);

② Describe how do you solve the problem.

- Main 함수

다음의 함수들을 모두 실행하여 조건에 맞는 결과를 도출한다.

int headInsert(NODE **head, NODE **tail, int i_data);
int headDelete(NODE **head, NODE **tail, int *i_data);
int tailInsert(NODE **head, NODE **tail, int i_data);
int tailDelete(NODE **head, NODE **tail, int *i_data);
void deleteList(NODE **head, NODE **);
static void printListfromHead(NODE *);
static void printListfromTail(NODE *);

- headInsert 함수

주어진 head pointer와 tail pointer를 가지는 linked list에 data를 list의 맨 앞에 삽입한다.

- headDelete 함수

주어진 head pointer와 tail pointer를 가지는 linked list에서 list의 맨 앞의 data를 삭제한다.

- tailInsert 함수

주어진 head pointer와 tail pointer를 가지는 linked list에 data를 list의 맨 뒤에 삽입한다

- tailDelete 함수

주어진 head pointer와 tail pointer를 가지는 linked list에서 list의 맨 뒤의 data를 삭제한다.

- deleteList 함수

주어진 head pointer와 tail pointer를 가지는 linked list의 모든 data를 삭제한다.

- printListfromHead 함수

주어진 head pointer를 가지는 linked list의 모든 data를 맨 앞에서부터 뒤까지 출력한다.

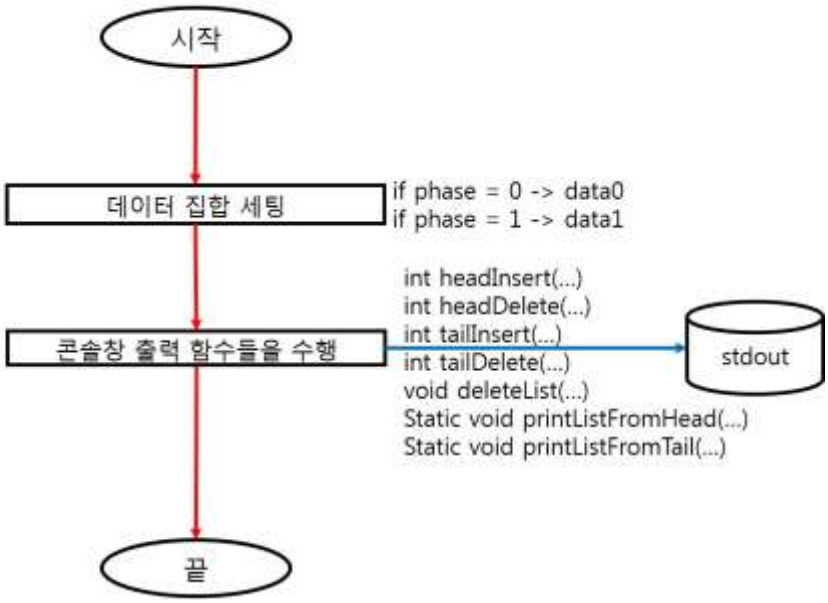
- printListfromTail 함수

주어진 tail pointer를 가지는 linked list의 모든 data를 멘 뒤에서부터 앞까지 출력한다.

③ Draw a flowchart of your algorithm

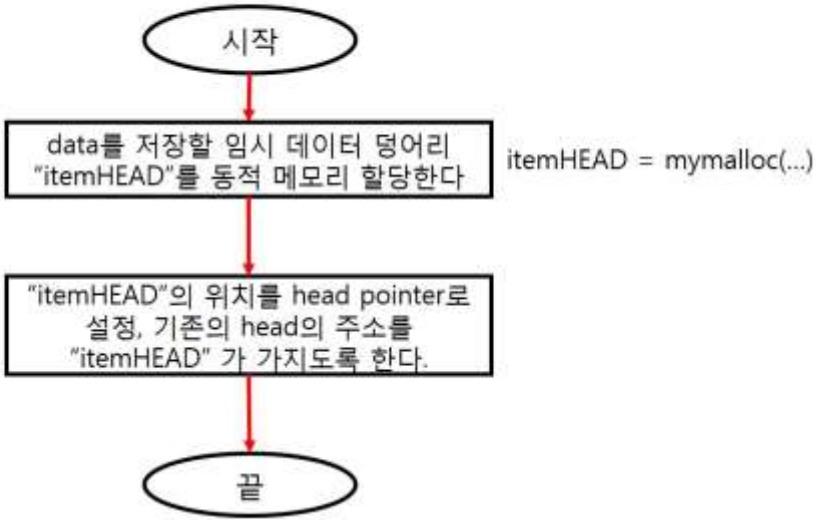
1. Main 함수

int main(void)



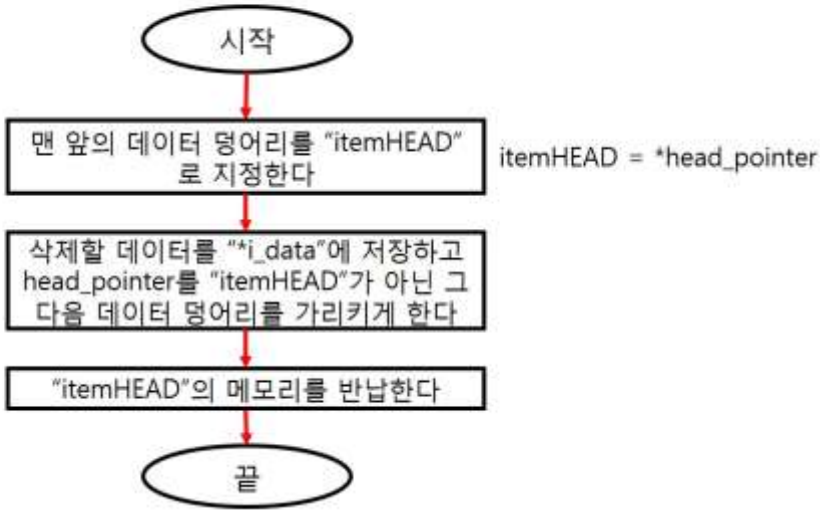
2. headInsert 함수

int headInsert (NODE **head, NODE **tail, int i_data)



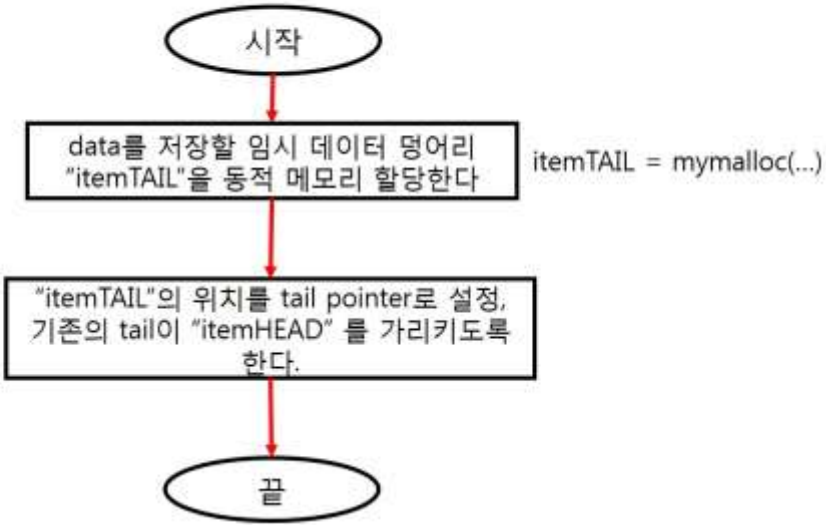
3. headDelete 함수

int headDelete (NODE **head, NODE **tail, int *i_data)



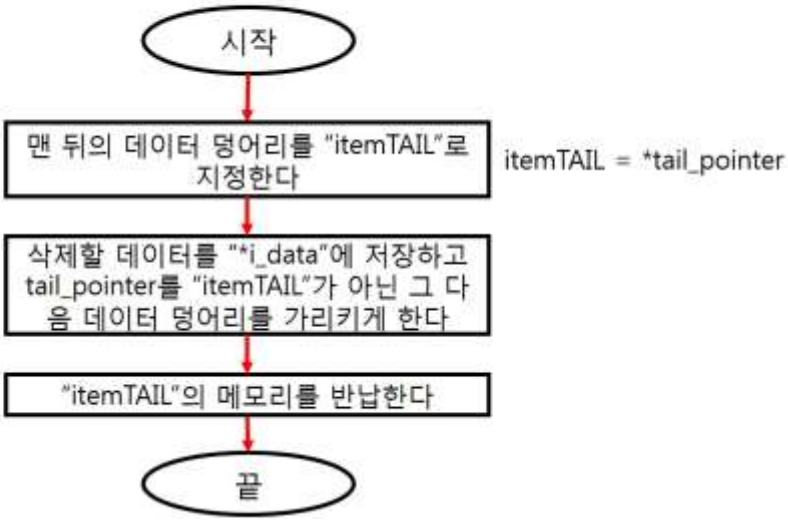
4. tailInsert 함수

int tailInsert (NODE **head, NODE **tail, int i_data)



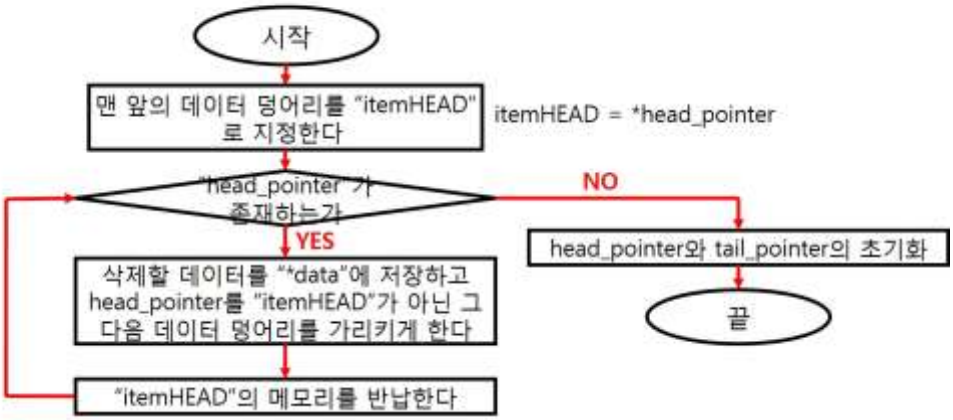
5. tailDelete 함수

int tailDelete (NODE **head, NODE **tail, int *i_data)



6. deleteList 함수

void deleteList (NODE **head, NODE **tail)

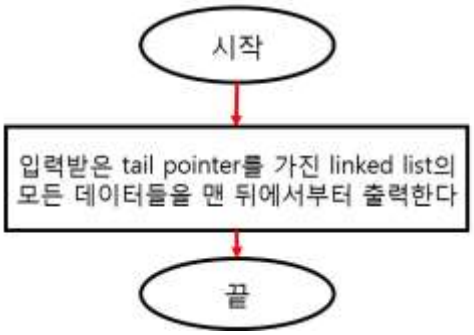


7. printListfromHead 함수

static void printListFromHead (NODE *head)

8. printListfromTail 함수

static void printListFromTail (NODE *tail)



3. Implementation

- Main 함수

다음의 함수들을 모두 실행하여 조건에 맞는 결과를 도출한다. 각 함수들은 데이터 집합 2개에 대하여, linked list를 생성하고 데이터를 삽입하거나 삭제하는 작업을 수행한다(printListfromHead, printListfromTail 함수는 해당 linked list 출력).

```
int headInsert(NODE **head, NODE **tail, int i_data);
int headDelete(NODE **head, NODE **tail, int *i_data);
int tailInsert(NODE **head, NODE **tail, int i_data);
int tailDelete(NODE **head, NODE **tail, int *i_data);
void deleteList(NODE **head, NODE **);
static void printListfromHead(NODE *);
static void printListfromTail(NODE *);
```

- headInsert 함수

주어진 head pointer와 tail pointer를 가지는 linked list에 data를 list의 맨 앞에 삽입한다. 삽입할 때, 임시 데이터 덩어리 "itemHEAD"를 dynamic memory allocation을 통해 생성하고, 그 데이터 덩어리를 기존의 linked list에 연결하여 새로운 linked list를 만들도록 한다.

- headDelete 함수

주어진 head pointer와 tail pointer를 가지는 linked list에서 list의 맨 앞의 data를 삭제한다.

- tailInsert 함수

주어진 head pointer와 tail pointer를 가지는 linked list에 data를 list의 맨 뒤에 삽입한다. 삽입할 때, 임시 데이터 덩어리 "itemTAIL"을 dynamic memory allocation을 통해 생성하고, 그 데이터 덩어리를 기존의 linked list에 연결하여 새로운 linked list를 만들도록 한다.

- tailDelete 함수

주어진 head pointer와 tail pointer를 가지는 linked list에서 list의 맨 뒤의 data를 삭제한다. 데이터가 하나인 경우와 존재하지 않는 경우 같은 예외도 생각한다. 데이터가 하나인 경우에는 list의 맨 뒷부분에 해당되는 tail이자 맨 앞부분에 해당되는 head 부분이므로, tail data를 삭제한다. 데이터가 존재하지 않는 경우에는 0을 return한다.

- deleteList 함수

주어진 head pointer와 tail pointer를 가지는 linked list의 모든 data를 삭제한다. headDelete 함수를 반복 실행하는 것과 같다.

4. Result

"2017311583.정해진.Q2.c" 를 컴파일 후 실행한 결과이다.

```
Perform headInsert at phase 0
  headInsert expected  : 9 18 23 43 51 65 76 89 99
  headInsert performed : 9 18 23 43 51 65 76 89 99
             reversed list of the above : 99 89 76 65 51 43 23 18 9

Perform headDelete at phase 0
  deleting a head node : 9 18 23 43 51 65 76 89 99
             reversed list of the above : 99 89 76 65 51 43 23 18 9
  deleting a head node : 18 23 43 51 65 76 89 99
             reversed list of the above : 99 89 76 65 51 43 23 18
  deleting a head node : 23 43 51 65 76 89 99
             reversed list of the above : 99 89 76 65 51 43 23
  deleting a head node : 43 51 65 76 89 99
             reversed list of the above : 99 89 76 65 51 43
  deleting a head node : 51 65 76 89 99
             reversed list of the above : 99 89 76 65 51
  deleting a head node : 65 76 89 99
             reversed list of the above : 99 89 76 65
  deleting a head node : 76 89 99
             reversed list of the above : 99 89 76
  deleting a head node : 89 99
             reversed list of the above : 99 89
  deleting a head node : 99
             reversed list of the above : 99
  headDelete performed okay.

Perform tailInsert at phase 0
  tailInsert expected  : 99 89 76 65 51 43 23 18 9
  tailInsert performed : 99 89 76 65 51 43 23 18 9
             reversed list of the above : 9 18 23 43 51 65 76 89 99

Perform tailDelete at phase 0
  deleting a tail node : 99 89 76 65 51 43 23 18 9
             reversed list of the above : 9 18 23 43 51 65 76 89 99
  deleting a tail node : 99 89 76 65 51 43 23 18
             reversed list of the above : 18 23 43 51 65 76 89 99
  deleting a tail node : 99 89 76 65 51 43 23
             reversed list of the above : 23 43 51 65 76 89 99
  deleting a tail node : 99 89 76 65 51 43
             reversed list of the above : 43 51 65 76 89 99
  deleting a tail node : 99 89 76 65 51
             reversed list of the above : 51 65 76 89 99
  deleting a tail node : 99 89 76 65
             reversed list of the above : 65 76 89 99
  deleting a tail node : 99 89 76
             reversed list of the above : 76 89 99
  deleting a tail node : 99 89
             reversed list of the above : 89 99
  deleting a tail node : 99
             reversed list of the above : 99
  tailDelete performed okay.

Deleting current list at phase 0
deleteList performed okay
```

첫번째 데이터 집합 {99, 89, 76, 65, 51, 43, 23, 18, 9}에 대해 잘 실행되었다.

```
Perform headInsert at phase 1
  headInsert expected  : 128 256 328 415 512
  headInsert performed : 128 256 328 415 512
             reversed list of the above : 512 415 328 256 128

Perform headDelete at phase 1
  deleting a head node : 128 256 328 415 512
             reversed list of the above : 512 415 328 256 128
  deleting a head node : 256 328 415 512
             reversed list of the above : 512 415 328 256
  deleting a head node : 328 415 512
             reversed list of the above : 512 415 328
  deleting a head node : 415 512
             reversed list of the above : 512 415
  deleting a head node : 512
             reversed list of the above : 512
  headDelete performed okay.

Perform tailInsert at phase 1
  tailInsert expected  : 512 415 328 256 128
  tailInsert performed : 512 415 328 256 128
             reversed list of the above : 128 256 328 415 512

Perform tailDelete at phase 1
  deleting a tail node : 512 415 328 256 128
             reversed list of the above : 128 256 328 415 512
  deleting a tail node : 512 415 328 256
             reversed list of the above : 256 328 415 512
  deleting a tail node : 512 415 328
             reversed list of the above : 328 415 512
  deleting a tail node : 512 415
             reversed list of the above : 415 512
  deleting a tail node : 512
             reversed list of the above : 512
  tailDelete performed okay.

Deleting current list at phase 1
deleteList performed okay
+++++ Checking memory... +++++
+++++ Memory is okay. +++++

Process returned 0 (0x0)   execution time : 0.070 s
Press any key to continue.
```

두번째 데이터 집합 {512, 415, 328, 256, 128}에 대해 잘 실행되었다.

5. Conclusion & Evaluation

실습을 통해 새로운 형식의 linked list인 doubly linked list를 다루는 방법에 대해 잘 알게 되었다.

6. 참고 문헌

[1] Min, H. B. and SKKU, “quiz2.pdf”
[2]컴퓨터인터넷 IT 용어대사전, “연결된 리스트”, 2011. 1. 20.