Lab13. Stack & Queue

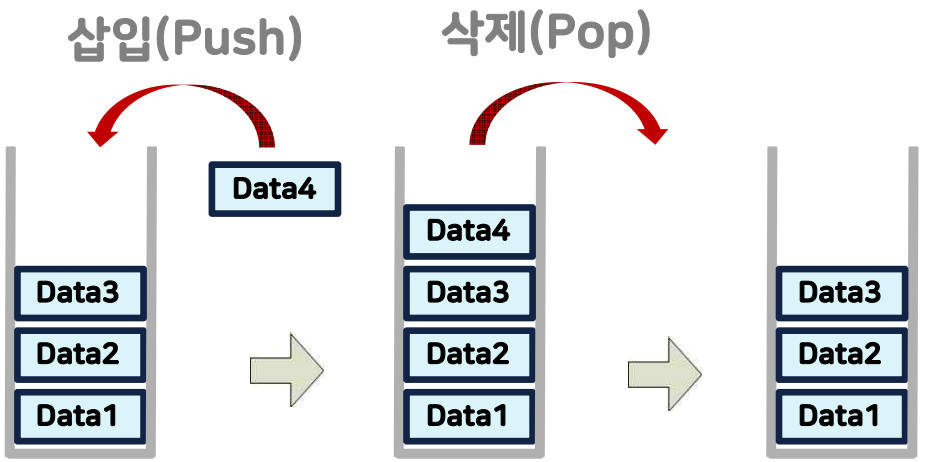
CSED101 LAB

STACK

Stack

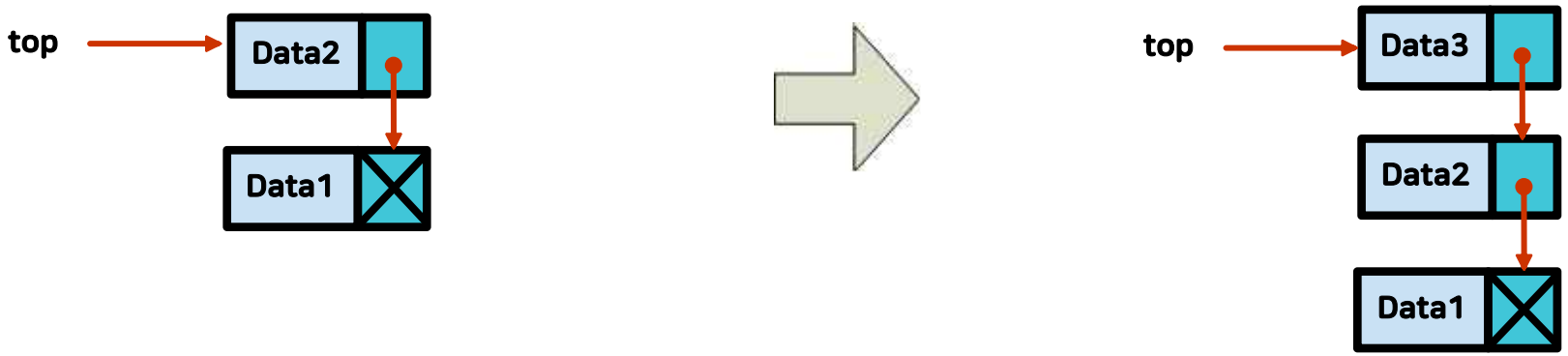
- Stack의 정의
  - 여러 데이터 항목을 일정한 순서로 나열한 자료 구조

- 기능 및 구조
  - 한쪽 끝에서만 새로운 항목 삽입(Push)하거나 기존 항목을 삭제(Pop)
  - LIFO(Last Input First Output)
    - 나중에 삽입된 데이터가 처음으로 삭제됨
  - Stack Primary Operation
    - Push(insert) & Pop(remove)

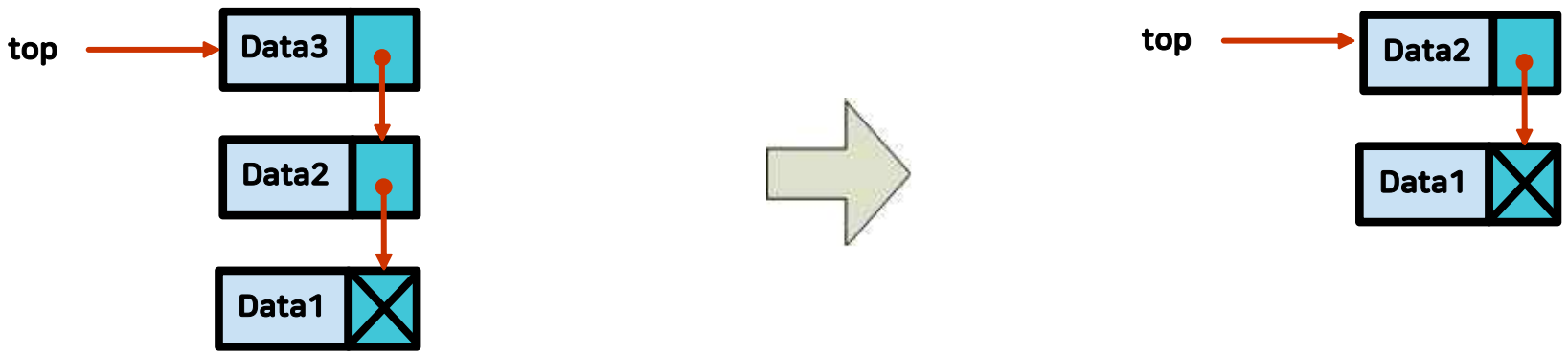


Stack - Linked List

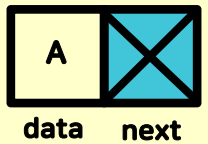
- Push
  - 새로운 node를 생성 후 top위에 연결



- Pop
  - top에 위치한 node를 제거 후 top을 다음 node로 이동

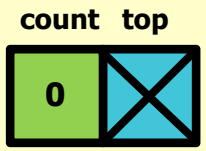


Stack (Linked list 구현)



스택 노드 구조

```
typedef struct node
{
        char data;
        struct node *next;
} NODE ;
```



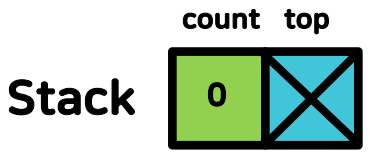
스택 헤드 구조

```
typedef struct
{
        int count;
        NODE *top;
} STACK ;
```

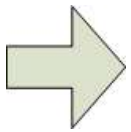
Stack Data Structure

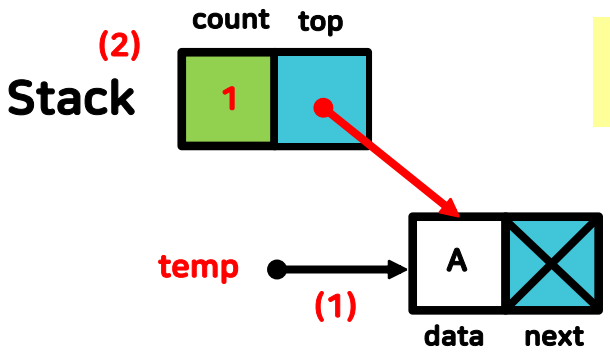
Linked List - Push

(1) 빈 스택에 삽입하는 경우




```
// 스택 생성 및 초기화
Stack = (STACK *)malloc(sizeof(STACK));
Stack->count = 0;
Stack->top = NULL;
```



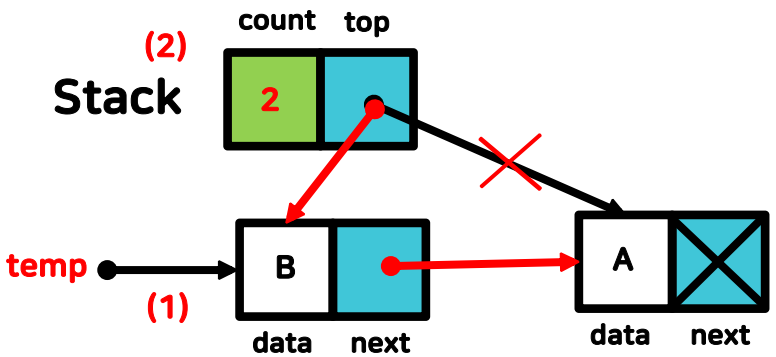



```
(2) Stack->top = temp;
    Stack->count++;
```

```
(1) temp = (NODE *)malloc(sizeof(NODE));
    temp->data = data;
    temp->next = Stack->top;
```

NULL

(2) 나머지 경우

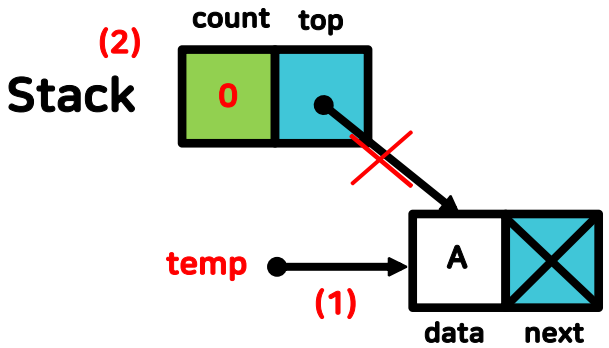



```
(1) temp = (NODE *)malloc(sizeof(NODE));
    temp->data = data;
    temp->next = Stack->top;
```

```
(2) Stack->top = temp;
    Stack->count++;
```

Linked List - Pop

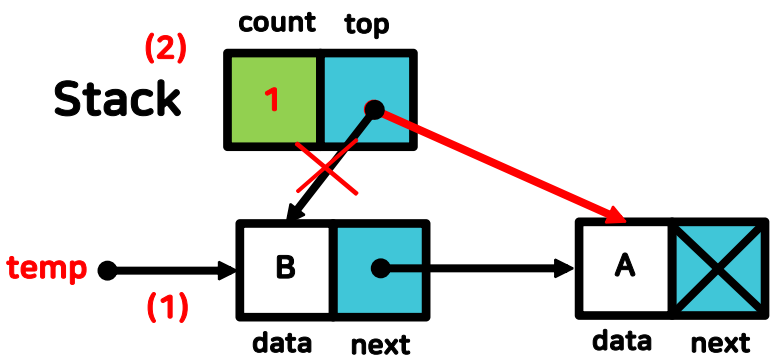
(1) 노드가 1개 남은 경우




```
(1) data = Stack->top->data;
    temp = Stack->top;

(2) Stack->top = temp->next;
    Stack->count--;
    free(temp);
```

NULL

(2) 나머지 경우




```
(1) data = Stack->top->data;
    temp = Stack->top;

(2) Stack->top = temp->next;
    Stack->count--;
    free(temp);
```

Problem 1

- Stack을 이용하여 입력된 문자열을 뒤집어 출력하는 프로그램을 구현하세요.

- 요구사항
  - Stack은 linked list 로 구현할 것.
  - 학번_Lab13_stack.c 를 다운로드 받아서 빈 칸을 채우시오.
  - 구현 기능
    - push
    - pop
    - isEmpty

- 실행 예제 (빨간색은 사용자 입력)

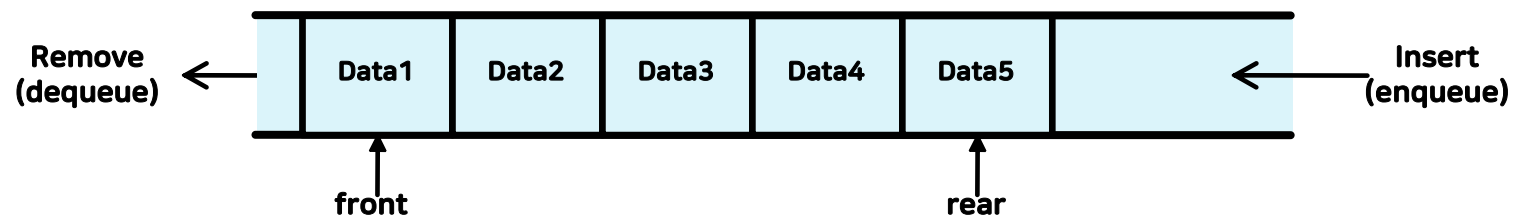
```
>> HELLO
   OLLEH

>> WORLD
   DLROW
```

- 제출파일명: 학번_Lab13_stack.c

QUEUE

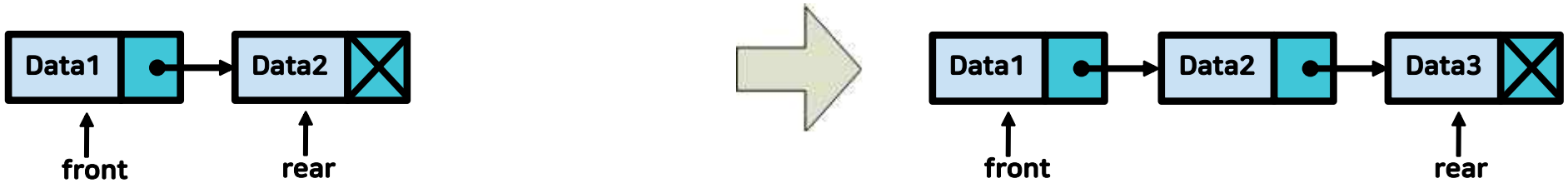
Queue

- Queue의 정의
  - Stack과 유사하게 여러 데이터 항목을 일정한 순서로 나열한 자료 구조
- 기능 및 구조
  - 뒤(rear)에서 새로운 항목의 삽입(enqueue)을 하고 앞(front)에서 기존 항목을 삭제(dequeue)
  - FIFO (First Input First Output)
    - 삽입한 순서대로 먼저 삽입된 데이터가 가장 먼저 삭제됨
  - Queue Primary Operation
    - Enqueue(insert) & Dequeue(remove)

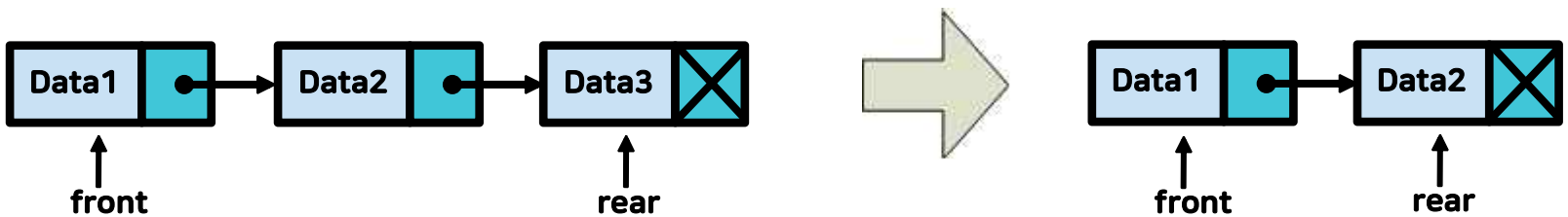


Queue - Linked List

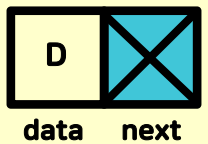
- Enqueue
  - 새로운 node를 생성 후 rear의 뒤에 연결



- Dequeue
  - front에 위치한 node를 제거 후 front를 다음 node로 이동

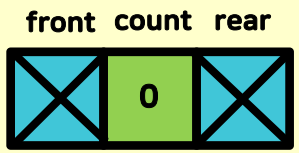


Queue (Linked list 구현)



큐 노드 구조

```
typedef struct node
{
        char data;
        struct node *next;
} NODE ;
```



큐 헤드 구조

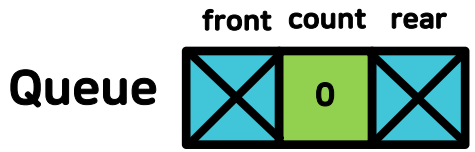
```
typedef struct
{
        int count;
        NODE *front;
        NODE *rear;
} QUEUE ;
```

Queue Data Structure

Linked List - Enqueue

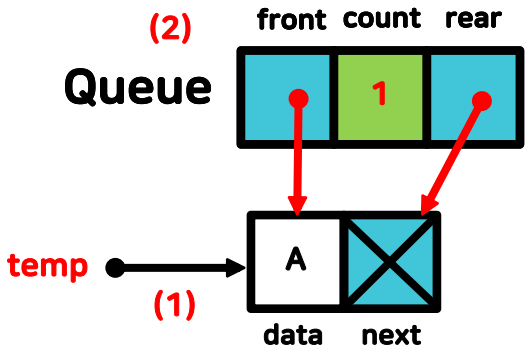
** Enqueue: rear 뒤에 삽입

(1) 빈 큐에 삽입하는 경우



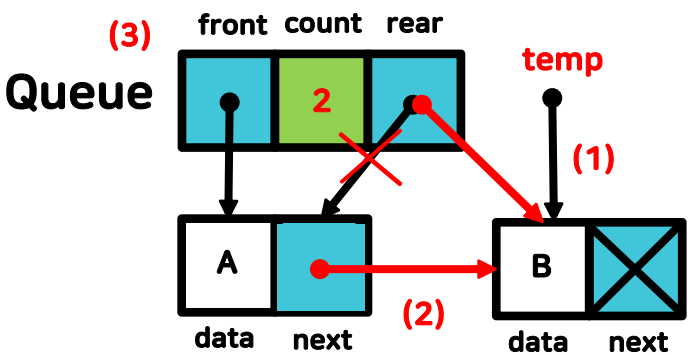

```
// 큐 생성 및 초기화
Queue = (QUEUE *)malloc(sizeof(QUEUE));
??
```

(1) temp = (NODE *)malloc(sizeof(NODE));
temp->data = data;
temp->next = NULL;



(2) Queue->front = temp;
Queue->rear = temp;
Queue->count++;

(2) 나머지 경우



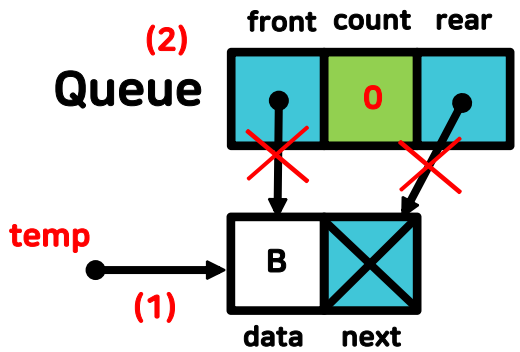
(2) Queue->rear->next = temp;

(3) Queue->rear = temp;
Queue->count++;

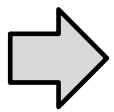
Linked List - Dequeue

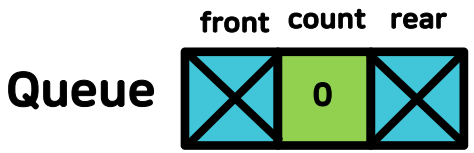
** Dequeue: front에 위치한 노드를 제거

(1) 노드가 1개 남은 경우

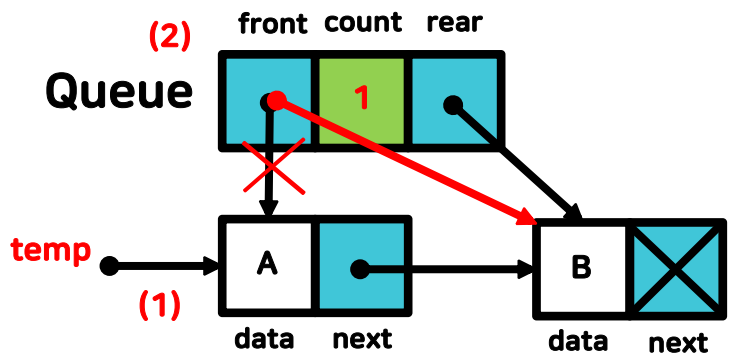



```
(2) Queue->front = NULL;
    Queue->rear  = NULL;
    Queue->count--;
    free(temp);
```





(2) 나머지 경우




```
(1) data = Queue->front->data;
    temp = Queue->front;
```

```
(2) Queue->front = Queue->front->next;
    Queue->count--;
    free(temp);
```

Problem 2

- Queue를 linked list로 구현하세요.

- 요구사항
  - 학번_Lab13_queue.c 를 다운로드 받아서 빈 칸을 채우시오.
  - 구현 기능
    - enqueue
    - dequeue
    - isEmpty

- 실행 예제 (빨간색은 사용자 입력)

```
>> HELLO
   HELLO

>> WORLD
   WORLD
```

- 제출파일명: 학번_Lab13_queue.c