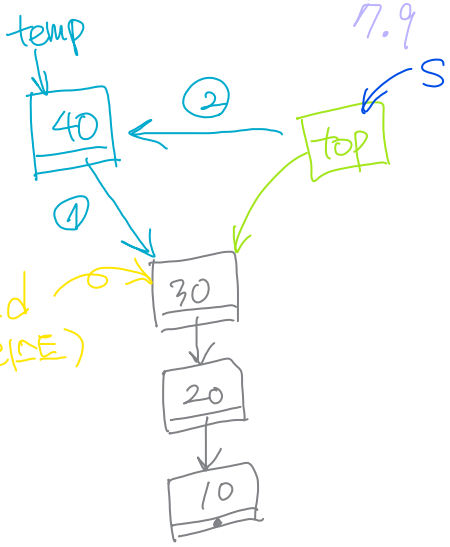
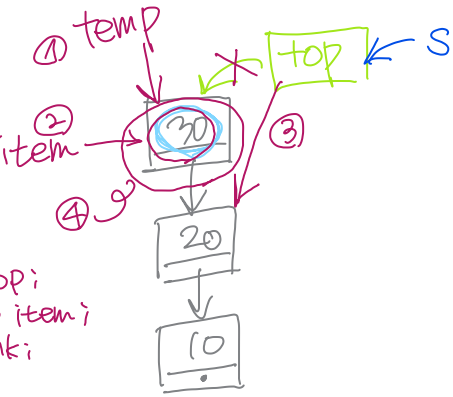
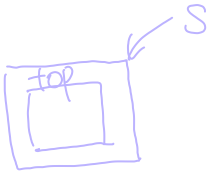
5장 스택: 연결리스트를 이용한 스택 ADT(교재 프로그램 5.4)

```
#include <stdio.h>
#include <malloc.h>

// 스택을 위한 타입 정의
typedef int element;

typedef struct StackNode {
        element item;
        struct StackNode *link;
} StackNode;

typedef struct {
        StackNode *top;
} LinkedStackType;

// 초기화 함수
void init(LinkedStackType *s)
{
        s->top = NULL;
}

// 공백 상태 검출 함수
int is_empty(LinkedStackType *s)
{
        return s->top == NULL;
}

// 포화 상태 검출 함수
int is_full(LinkedStackType *s) // Stack 연산중의 하나라 남겨두었지만 항상 거짓을 반환. 즉 꽉 차있을 때는 없음
{
        return 0;
}

// 삽입 함수
void push(LinkedStackType *s, element item)
{
    StackNode *temp=(StackNode *)malloc(sizeof(StackNode));
    if( temp == NULL ){
        fprintf(stderr, "메모리 할당에러₩n");
        return;
    }
    else{
        temp->item = item;
        temp->link = s->top;
        s->top = temp;
    }
}
```



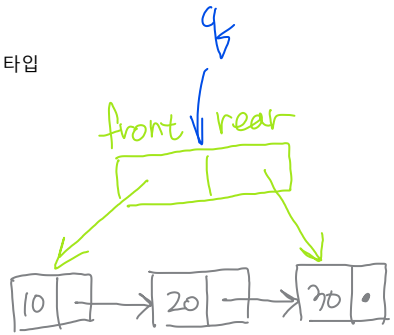

```
// 삭제 함수
element pop(LinkedStackType *s)
{
        if( is_empty(s) ) {
                fprintf(stderr, "스택이 비어있음₩n");
                exit(1);
        }
        else{
                StackNode *temp = s->top;
                element item = temp->item;
                s->top = s->top->link;
                free(temp);
                return item;
        }
}
// 피크 함수
element peek(LinkedStackType *s)
{
        if( is_empty(s) ) {
                fprintf(stderr, "스택이 비어있음₩n");
                exit(1);
        }
        else{
                return s->top->item;
        }
}
// 주 함수
void main()
{
        LinkedStackType s;

        init(&s);

        push(&s,1);
        push(&s,2);
        push(&s,3);

        printf("%d₩n", pop(&s));
        printf("%d₩n", pop(&s));
        printf("%d₩n", pop(&s));
        printf("%d₩n", is_empty(&s));
}
```

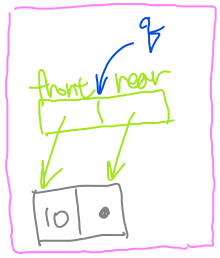




6장 큐: 연결리스트를 이용한 큐 ADT(교재 프로그램 6.4)

```
//        LinkedQueue.c

#include <stdio.h>
#include <malloc.h>

typedef int element;        // 요소의 타입

typedef struct QueueNode {        // 큐의 노드의 타입
        element item;
        struct QueueNode *link;
} QueueNode;

typedef struct {            // 큐 ADT 구현
        QueueNode *front, *rear;
} QueueType;

// 오류 함수
void error(char *message)
{
        fprintf(stderr,"%s₩n",message);
        exit(1);
}

// 초기화 함수
void init(QueueType *q)
{
        q->front = q->rear = NULL;
}

// 공백 상태 검출 함수
int is_empty(QueueType *q)
{
        return q->front == NULL;
}

// 포화 상태 검출 함수
int is_full(QueueType *q) // 항상 거짓, 포화상태일때가 없다. 연결리스트로 구현하였으므로
{
        return 0;
}
```






```
// 삽입 함수
void enqueue(QueueType *q, element item)
{
        QueueNode *temp=(QueueNode *)malloc(sizeof(QueueNode));
        if(temp == NULL )
                error("메모리를 할당할 수 없습니다");
        else {
                temp->item = item;              // 데이터 저장
                temp->link = NULL;              // 링크 필드를 NULL
                if( is_empty(q) ){              // 큐가 공백이면
                        q->rear = temp;
                        q->front = temp;
                }
                else {                          // 큐가 공백이 아니면
                        ① q->rear->link = temp;
                        ② q->rear = temp;
                }
        }
}

// 삭제 함수
element dequeue(QueueType *q)
{
        ① QueueNode *temp = q->front;
        element item;
        if( is_empty(q) )                       // 공백상태
                error("큐가 비어 있읍니다");
        else {
                ② item = temp->item;            // 데이터를 꺼낸다.
                ③ q->front = q->front->link;
                   if( q->front == NULL) // 노드가 하나 남았으면
                ④ free(temp);                   // 동적메모리 해제
                   return item;                 // 데이터 반환
        }
}

// 연결된 큐 테스트 함수
void main()
{
        QueueType q;

        init(&q);
        enqueue(&q, 1);
        enqueue(&q, 2);
        enqueue(&q, 3);
        printf("dequeue()=%d₩n",dequeue(&q));
        printf("dequeue()=%d₩n",dequeue(&q));
        printf("dequeue()=%d₩n",dequeue(&q));
}
```