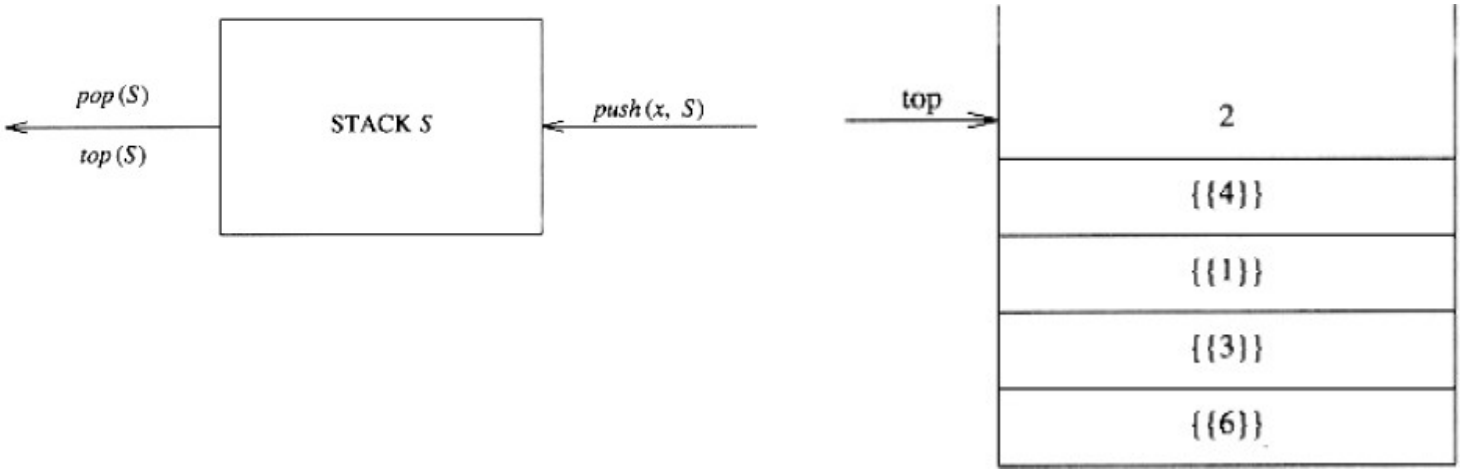
데이터 구조

강의 노트 4 스택과 큐

김유증, 박사 조교수
컴퓨터 과학 및 정보 공학부
한국 가톨릭 대학교, 대한민국

스택

- 스택은 삽입 및 삭제가 제한된 목록입니다.
  - 삽입 및 삭제는 목록의 맨 끝, 즉 *상단이라는* 한 위치에서만 수행할 수 있습니다.
  - 밀기: 상단에서 삽입
  - 팝: 상단에서 삭제
  - LIFO(라스트 인, 퍼스트 아웃): 가장 최근에 삽입된 항목부터 삭제해야 합니다.



스택: L 잉크 L은 구현 중입니다.

- 스택은 링크된 목록으로 쉽게 구현할 수 있습니다

```
1  struct Node;
2  typedef struct Node *PtrToNode;
3  typedef PtrToNode Stack;
4
5  struct Node {
6      ElementType Element;
7      PtrToNode   Next;
8  };
9
10 Stack CreateStack(void)
11 {
12     Stack S;
13     S = (Stack)malloc(sizeof(struct Node));
14     if( S == NULL ) FetalError("Out of Memory");
15     S->Next = NULL;
16     return S;
17 }
18
19 void Push(ElementType X, Stack S)
20 {
21     PtrToNode Tmp;
22     Tmp = (PtrToNode)malloc(sizeof(struct Node));
23     if( Tmp == NULL ) FetalError("Out of Memory");
24     Tmp->Element = X;
25     Tmp->Next = S->Next;
26     S->Next = Tmp;
27 }
```

스택: 배열 구현

- 스택은 링크된 목록으로 쉽게 구현할 수 있습니다

```
29  ElementType Top(Stack S)
30  {
31      if( S->Next ) return S->Next->Element;
32      Error("Empty Stack");
33      return EmptyElement;
34  }
35
36  void Pop(Stack S)
37  {
38      PtrToNode Tmp = S->Next;
39      if( Tmp == NULL ) {
40          Error("Empty Stack");
41      } else {
42          S->Next = Tmp->Next;
43          free(Tmp);
44      }
45  }
```

스택: 배열 대 L 잉크 L입

- 운영 복잡성

- 확장성

- 구현 복잡성

- 실제 실행 시간

스택: 포스트픽스 표현

- 인픽스 표현식
  - 산술의 자연스러운 표현: 예: 4.99*1.06 + 5.99+6.99*1.06 = ?
  - 컴퓨터로 해석하기 쉽지 않음

- 컴퓨터로 접미사 식 계산하기
  - A1 = 4.99*1.06
  - A1 = A1 + 5.99
  - A2 = 6.99*1.06
  - A1+A2

- 컴퓨터의 작업을 순서대로 배치하기
  - 4.99 1.06 * 5.99 + 6.99 1.06 * + ➔ 포스트픽스 표현식

- 포스트픽스 표현식
  - 컴퓨터가 작동하는 순서대로 표현합니다.

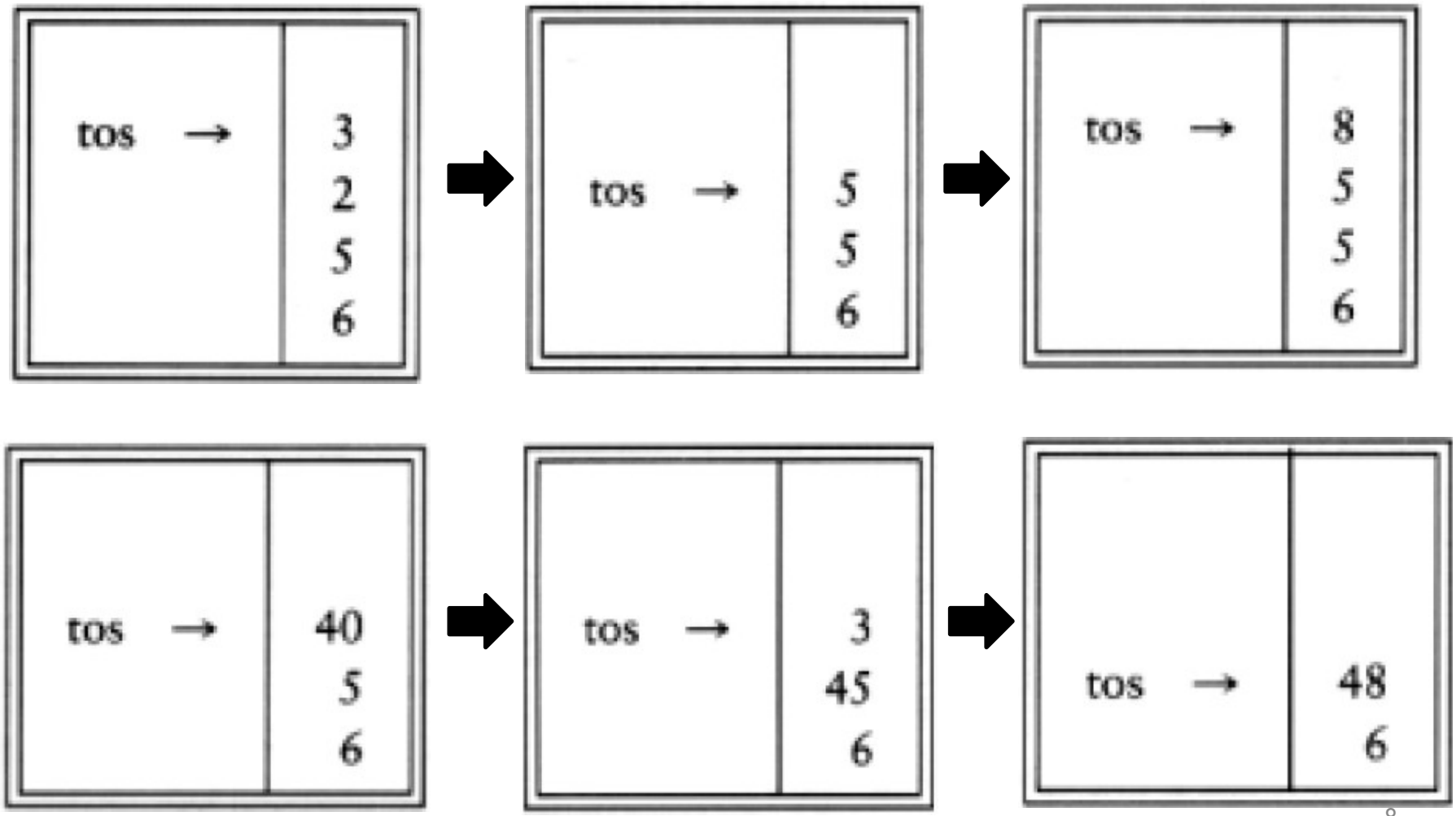
스택: 포스트픽스 표현

- 스택을 사용하여 후위 접두사 식 계산하기
  - 포스트픽스 식 계산6 5 2 3 + 8 * + 3 + *
  - 접미사 표현식은 무엇인가요?

- 계산 절차
  - 기호 하나씩 읽기
  - 숫자 읽기, 푸시
  - 연산자를 읽고, 두 개의 숫자를 팝업하고, 계산하고, 결과를 푸시합니다.
  - 정답인 숫자가 하나만 나올 때까지 반복합니다.

스택: 포스트픽스 표현

- 스택을 사용하여 후위 접두사 식 계산하기
  - 포스트픽스 식 계산6 5 2 3 + 8 * + 3 + *
  - 복잡성?



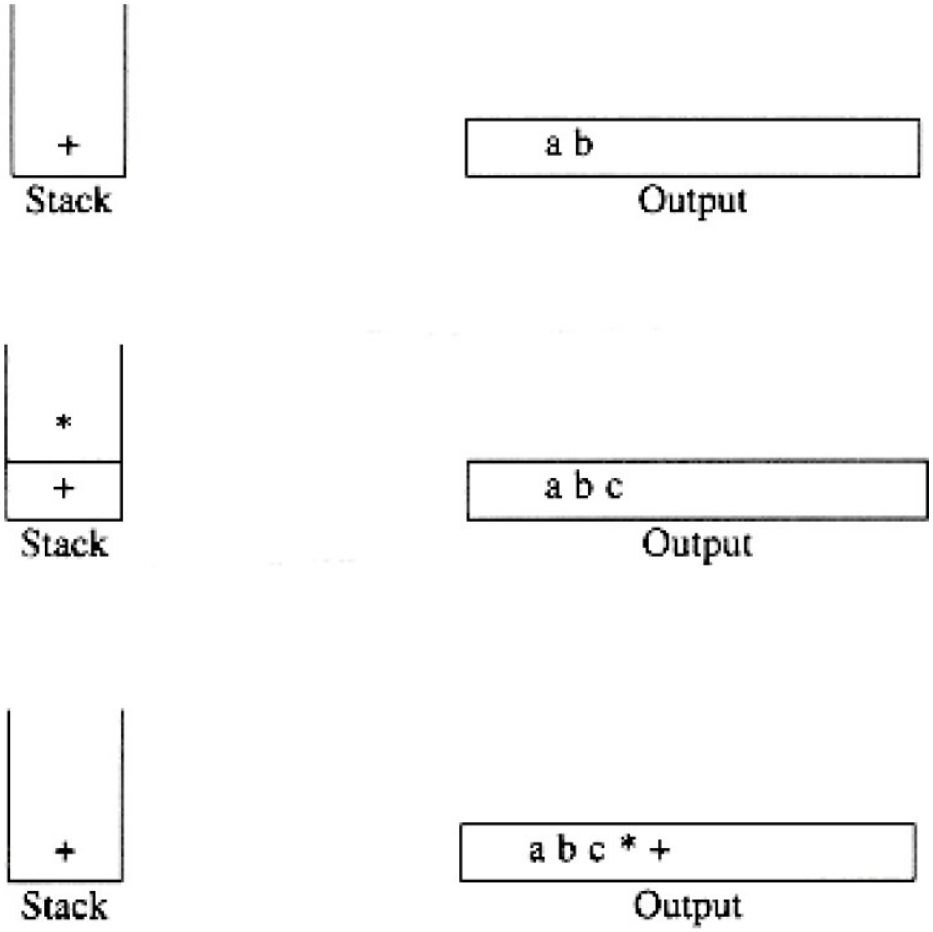
스택: 인픽스에서 포

- 스택을 사용하여 접두사 표현식을 접미사 표현식으로 변환하기
  - 일반화할 수 있지만 여기서는 +, *, (, 및)에 집중해 보겠습니다.
  - 접두사 표현식: a + b * c + (d * e + f) *g
  - 포스트픽스 표현식: a b c * + d e * f + g * +

- 절차
  - 오퍼레이터를 위한 스택과 출력을 위한 스토리지 준비하기
  - 접두사 식의 기호를 하나씩 읽기
  - 숫자를 읽고 출력에 입력합니다.
  - '+', '*', '('를 읽고 스택에서 우선순위가 낮은 항목을 찾을 때까지 팝한 다음 밀어 넣습니다.
  - ')'를 읽고 '("가 나올 때까지 출력을 팝업하고 씁니다.
  - 예외: '('는 ')'를 읽지 않으면 팝업되지 않습니다.

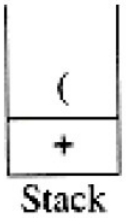
스택: 인픽스에서 포

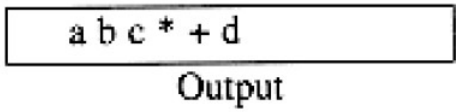
- 스택을 사용하여 접두사 표현식을 접미사 표현식으로 변환하기
  - 접두사 표현식: a + b * c + (d * e + f) *g

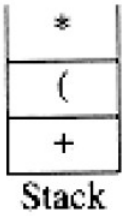


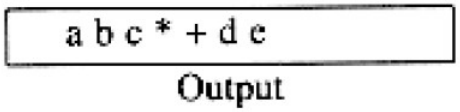
스택: 인픽스에서 포

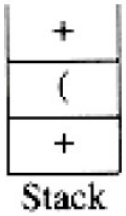
- 스택을 사용하여 접두사 표현식을 접미사 표현식으로 변환하기
  - 접두사 표현식: a + b * c + (d * e + f) *g

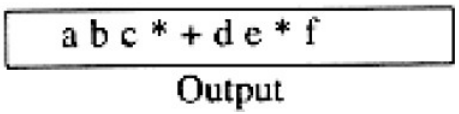






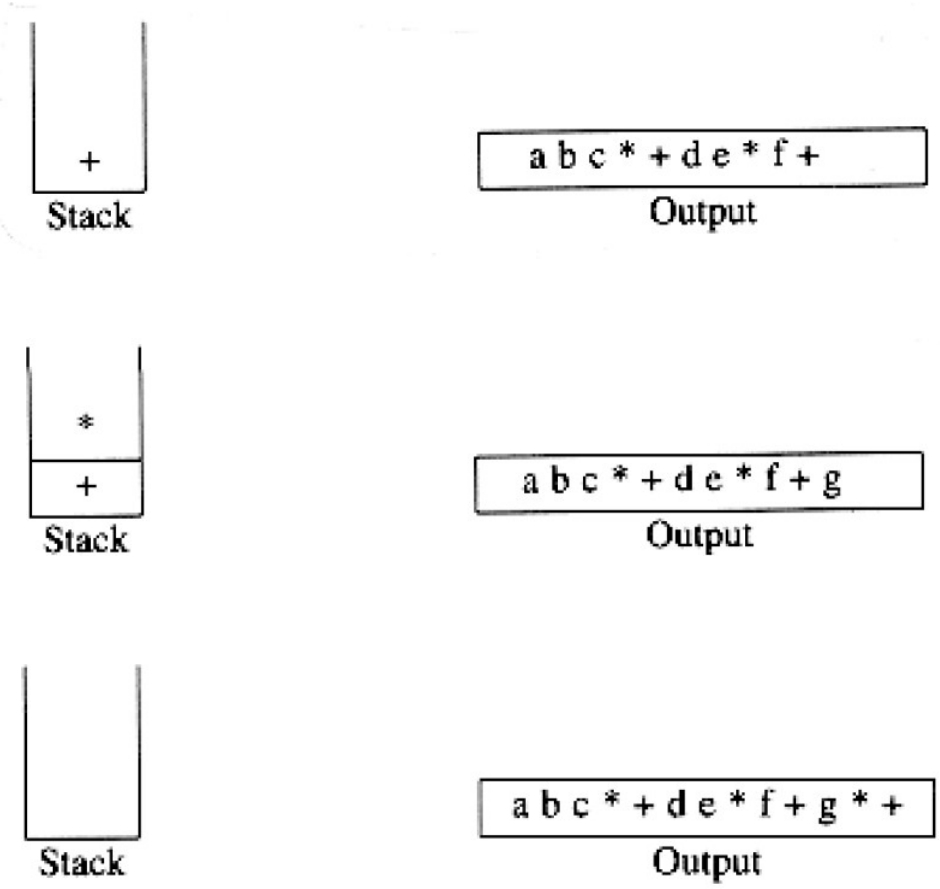






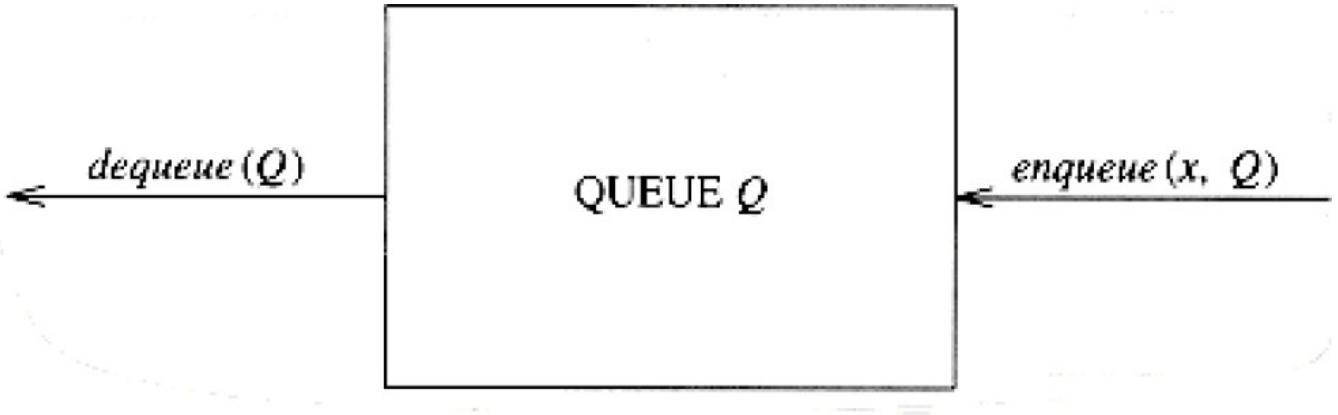
스택: 인픽스에서 포

- 스택을 사용하여 접두사 표현식을 접미사 표현식으로 변환하기
  - 접두사 표현식: a + b * c + (d * e + f) *g



대기

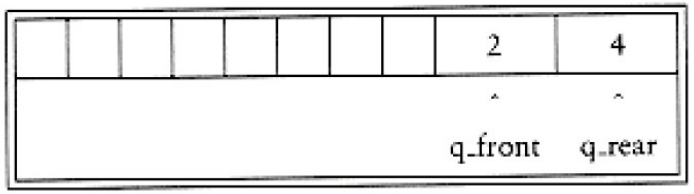
- 대기열은 삽입 및 삭제가 제한된 목록입니다.
  - 목록 끝에 요소를 삽입하고 목록의 시작 부분에서 삭제하기
  - 대기열: 마지막에 삽입
  - 대기열: 시작 시 삭제
  - FIFO(선입선출) / LILO(라스트 인, 라스트 아웃)
  - 배열뿐만 아니라 링크된 목록으로도 구현할 수 있습니다.



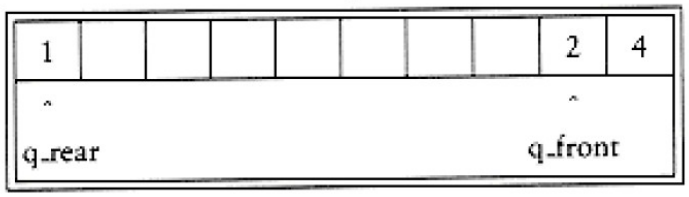
큐: 원형 배열 구현

- 원형 배열: 길이 L의 배열에서 L -1은 마지막 셀의 인덱스이고 마지막 셀의 다음 셀은 인덱스가 0인 첫 번째 셀입니다.
  - 대기열에는 일반적으로 앞쪽과 뒤쪽이 있습니다.

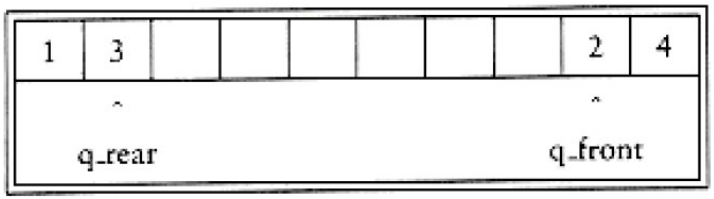
Initial State



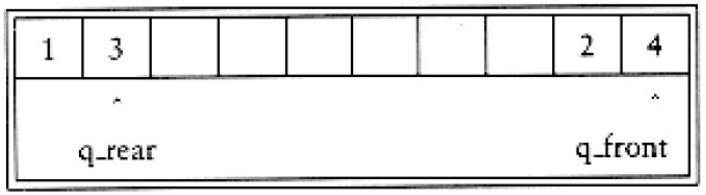
After *Enqueue*(1)



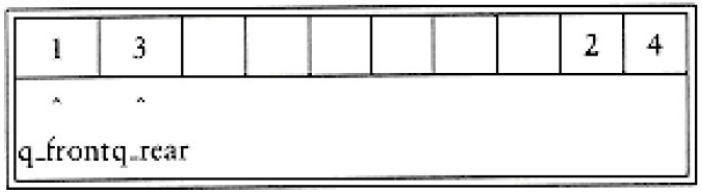
After *Enqueue* (3)



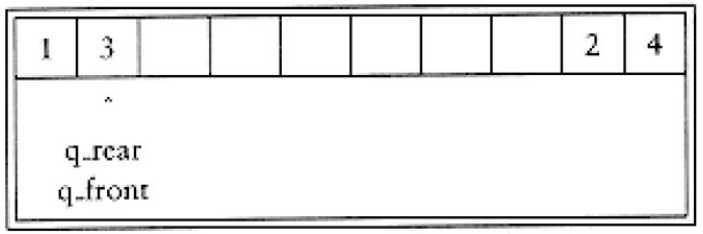
After *Dequeue*, Which Returns 2



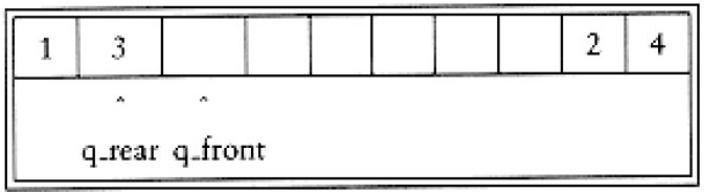
After *Dequeue*, Which Returns 4



After *Dequeue*, Which Returns 1



After *Dequeue*, Which Returns 3 and Makes the Queue Empty



큐: 원형 배열 구현

- 원형 배열: 길이 L의 배열에서 L -1은 마지막 셀의 인덱스이고 마지막 셀의 다음 셀은 인덱스가 0인 첫 번째 셀입니다.
  - 대기열에는 일반적으로 앞쪽과 뒤쪽이 있습니다.

```
 1  struct QueueRecord;
 2  typedef struct QueueRecord *Queue;
 3
 4  int IsEmpty(Queue Q);
 5  int IsFull(Queue Q);
 6  Queue CreateQueue(int MaxElements);
 7  void DisposeQueue(Queue Q);
 8  void MakeEmpty(Queue Q);
 9  void Enqueue(ElementType X, Queue Q);
10  ElementType Front(Queue Q);
11  void Dequeue(Queue Q);
12  ElementType FrontAndDequeue(Queue Q);
13
14  struct QueueRecord {
15      int Capacity;
16      int Front;
17      int Rear;
18      int Size;
19      ElementType *Array;
20  };
```

큐: 원형 배열 구현

- 원형 배열: 길이 L의 배열에서 L -1은 마지막 셀의 인덱스이고 마지막 셀의 다음 셀은 인덱스가 0인 첫 번째 셀입니다.
  - 대기열에는 일반적으로 앞쪽과 뒤쪽이 있습니다.

```
22  int IsEmpty(Queue Q)
23  {
24      return( Q->Size==0 );
25  }
26
27  is IsFull(Queue Q)
28  {
29      return( Q->Size == Q->Capacity );
30  }
31
32  void MakeEmpty(Queue Q)
33  {
34      Q->Size = 0;
35      Q->Front = 1;
36      Q->Rear = 0;
37  }
38
39  int Succ(int Value,Queue Q)
40  {
41      if( ++Value == Q->Capacity ) Value == 0;
42      return Value;
43  }
```

큐: 원형 배열 구현

```
45  void Enqueue(ElementType X,Queue Q)
46  {
47      if( IsFull(Q) ) {
48          Error("Full Queue");
49      } else {
50          Q->Size++;
51          Q->Rear = Succ(Q->Rear,Q);
52          Q->Array[Q->Rear] = X;
53      }
54  }
55
56  void Dequeue(Queue Q)
57  {
58      if( IsEmpty(Q) ) {
59          Error("Empty Queue");
60      } else {
61          Q->Size--;
62          Q->Front = Succ(Q->Front,Q);
63      }
64  }
65
66  ElementType Front(Queue Q)
67  {
68      if( !IsEmpty(Q) ) return Q->Array[Q->Front];
69      Error("Empty Queue");
70      return NULL_ELEMENT;
71  }
```

대기열: 애플리케이

- 대기열은 어디에나 있습니다.
  - 영화 매표소: 대기열, 선착순 입장.
  - 웹 서버는 큐를 사용하여 액세스 요청을 처리합니다.
  - OS 스케줄러는 대기열을 사용하여 리소스를 할당합니다.

- 큐 이론
  - 대기열의 모든 종류의 문제 해결
  - 대기 시간, 최적의 대기열 크기, 손실 확률 계산 등...