COSE213: Data Structure

Lecture 4 review

Minseok Jeon

2024 Fall

과제 1 관련 공지

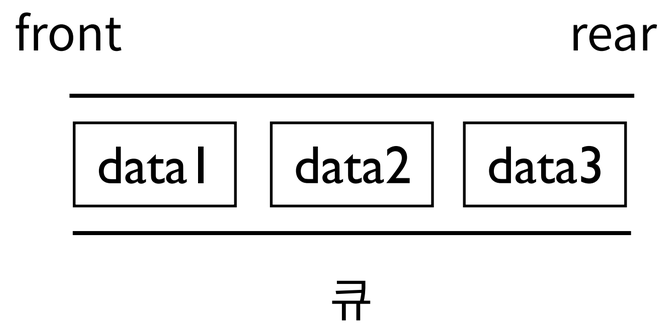
- 과제 제출은 3개의 헤더파일들을 압축하지 않고 제출해야 함

중간고사 공지

- 날짜: 10/25 (금)
- 시간: 12:00 ~ 13:15
- 장소: 과학도서관 409호
- 시험범위: Lecture 1 (배열) ~ Lecture 6 (리스트)

큐 (Queue)

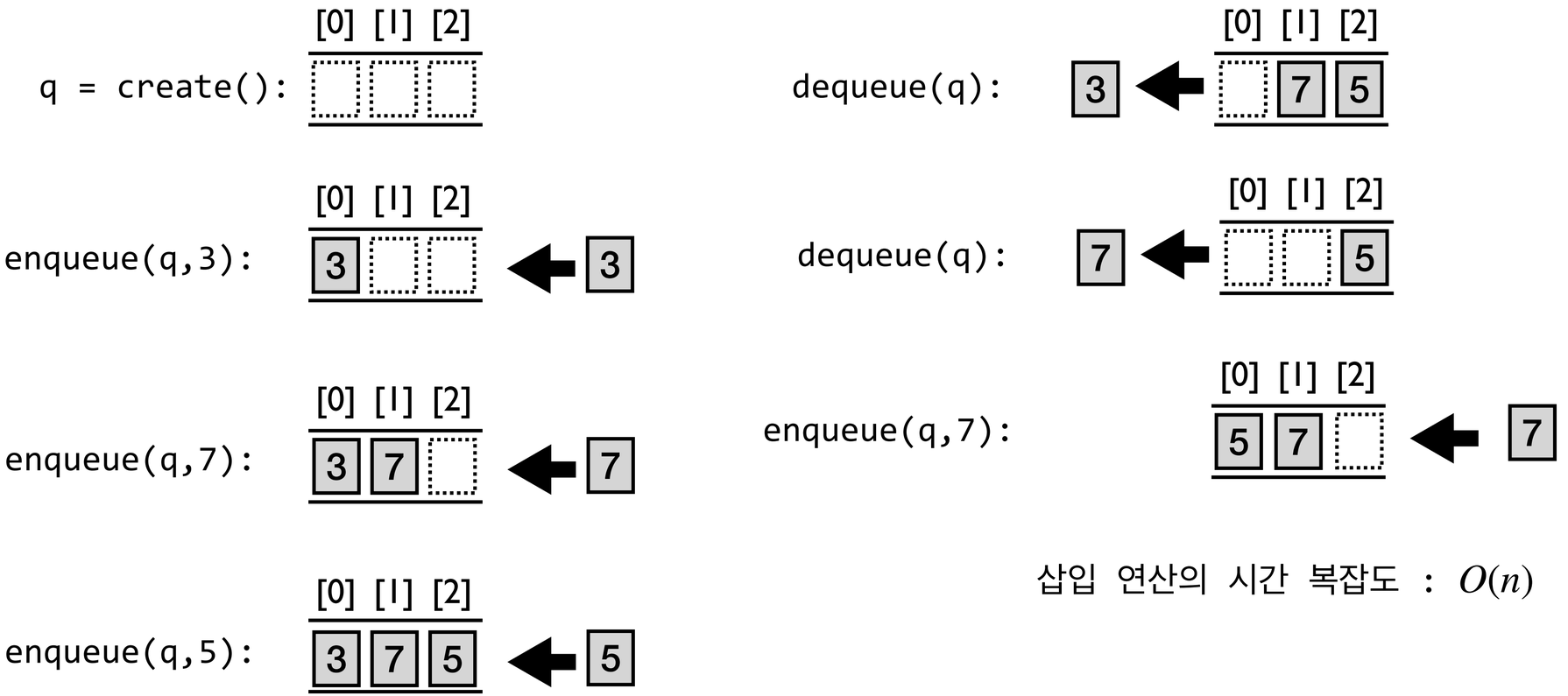
- 큐(Queue)는 선입선출(FIFO: Fist In, First Out) 원칙을 따르는 자료구조
- 큐의 추상 자료형:
  - create() : 비어있는 큐를 생성 후 반환
  - enqueue(q, e) : 큐 q에서 주어진 데이터 e를 큐의 맨 뒤에 추가
  - dequeue(q) : 큐 q가 비어있지 않으면 맨 앞 데이터를 삭제하고 반환
  - peek(q) : 큐 q가 비어있지 않으면 맨 앞 데이터를 제거하지 않고 반환
  - isEmpty(q) : 큐 q가 비어있으면 true를 아니면 false를 반환
  - isFull(q) : 큐 q가 가득 차 있으면 true를 아니면 false를 반환



- ADT of array in Java: https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html
  ADT of array in Ocaml: https://ocaml.org/manual/5.2/api/Queue.html

배열을 이용한 큐의 구현

- 선형 큐 (linear queue): 1차원 배열을 이용하여 큐를 구현하는 경우
  - 선형 큐에서는 삽입에 추가적인 이동이 필요할 수 있음

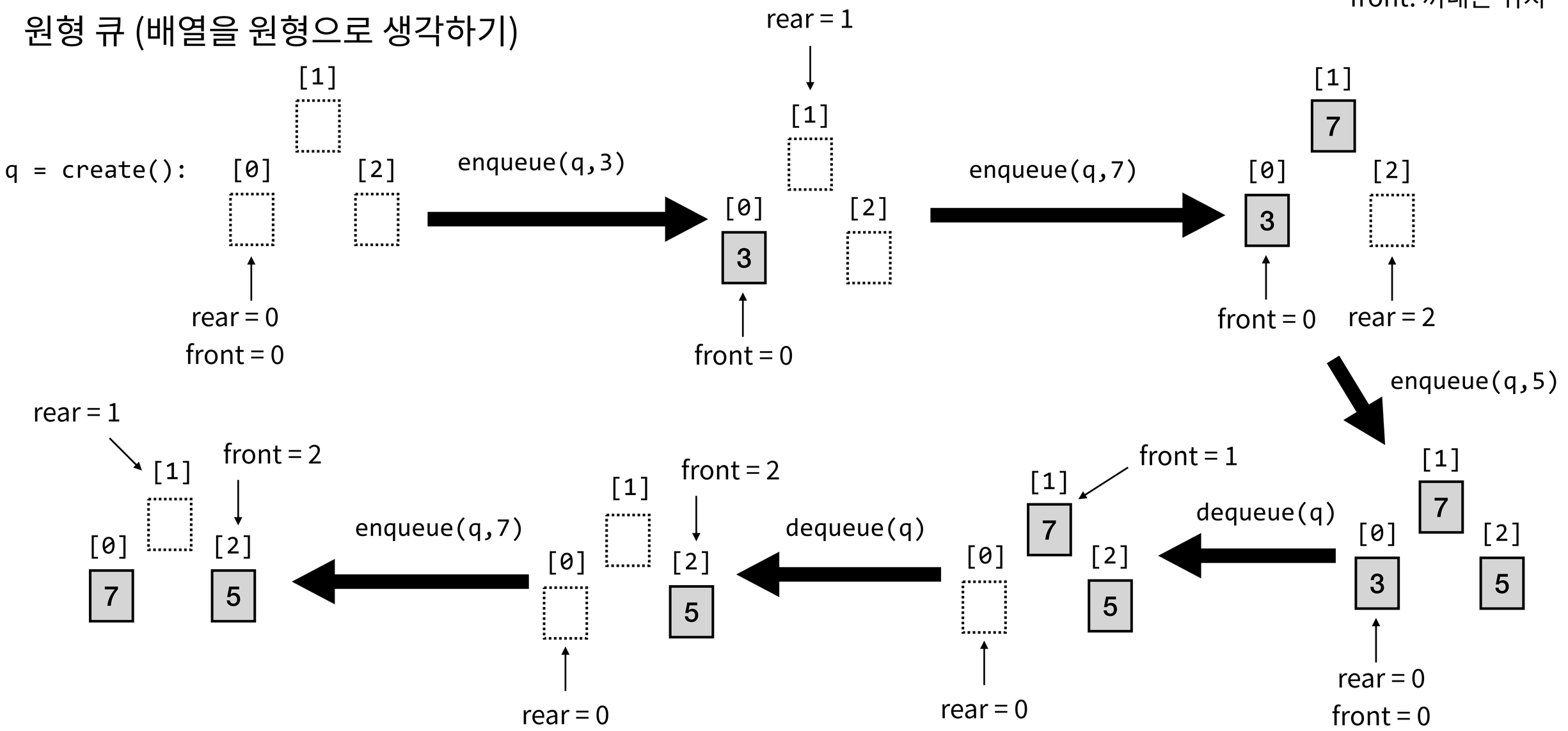


삽입 연산의 시간 복잡도 : $O(n)$

배열을 이용한 큐의 구현

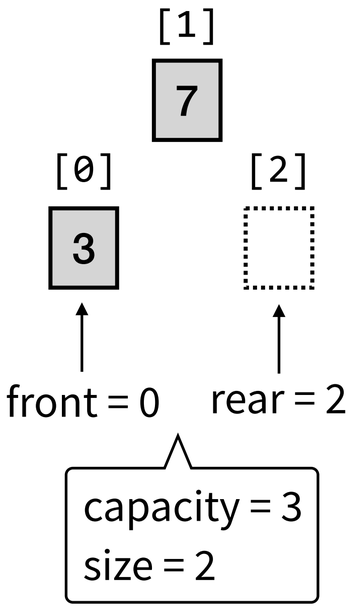
rear: 저장하는 위치
front: 꺼내는 위치

- 원형 큐 (배열을 원형으로 생각하기)



Enqueue

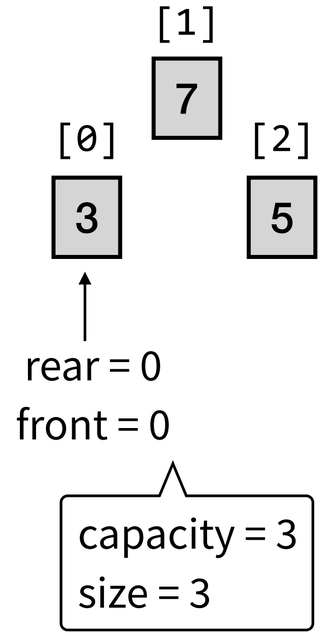
- enqueue : 큐의 맨 뒤에 주어진 새로운 정수 데이터를 추가



5

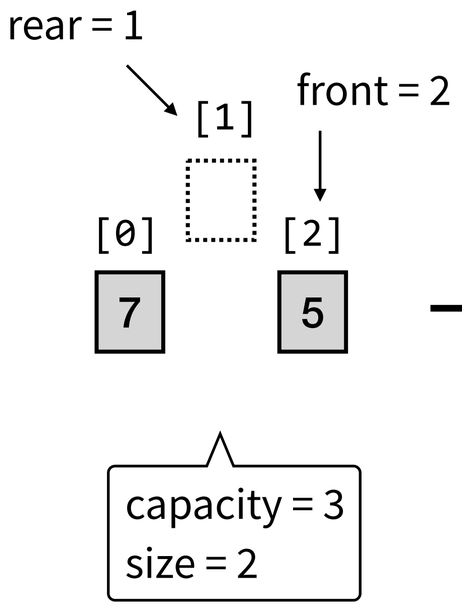
data

```
procedure enqueue(queue, data)
  if isFull(queue) then
    print("Cannot enqueue. Queue is full.")
    return error()                                  ▷ failed
  end if
  idx ← queue.rear
  queue.items[idx] ← data
  queue.rear ← (queue.rear + 1) mod queue.capacity
  queue.size ← queue.size + 1
  return queue
end procedure
```

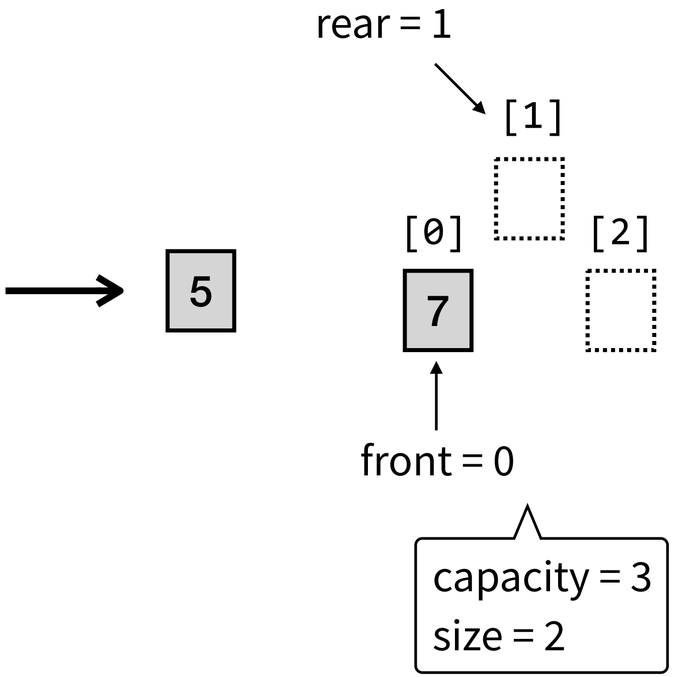


Dequeue

- dequeue : 큐의 가장 앞에 있는 데이터를 삭제하고 반환



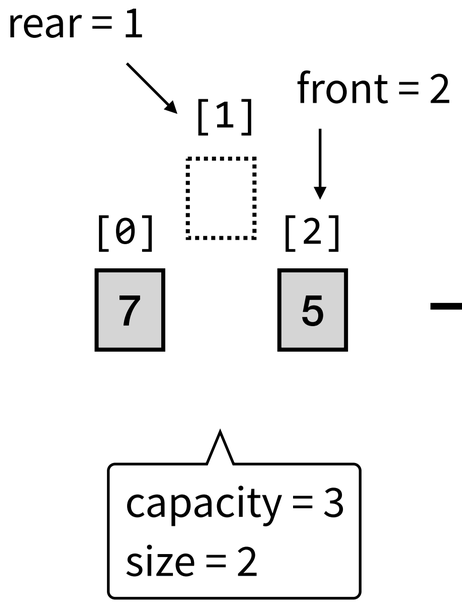

```
procedure dequeue(queue)
  if isEmpty(queue) then
    print("Cannot dequeue. Queue is empty.")
    return error()                                  ▷ failed
  end if
  item ← queue.items[queue.front]
  queue.size  ← queue.size - 1
  queue.front ← (queue.front + 1) mod queue.capacity
  return item                                       ▷ succeed
end procedure
```



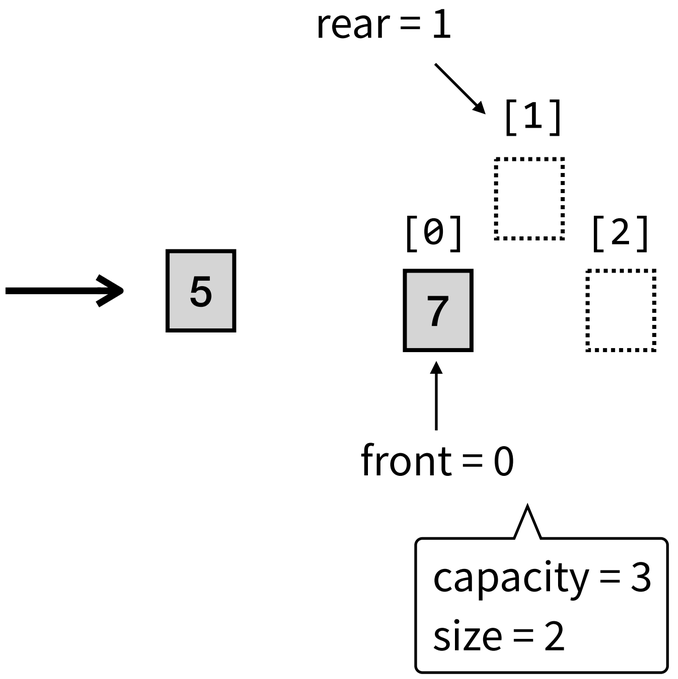
- dequeue : 큐의 가장 앞에 있는 데이터를 삭

```c
typedef struct {
    int *items;
    int front;
    int rear;
    int size;
    int capacity;
} Queue;
```

질문:
front 변수 없이 큐를 구현할 순 없을까?



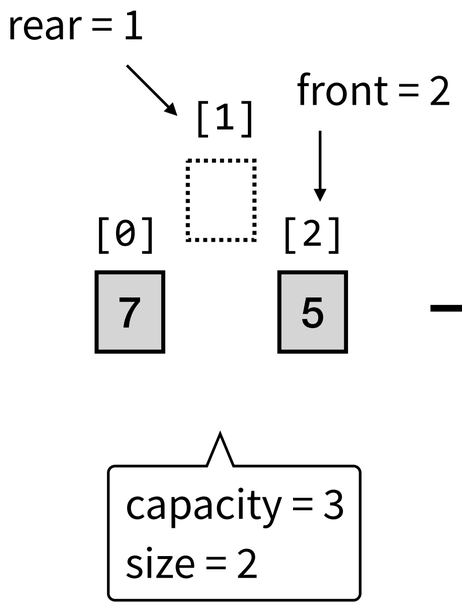

```
procedure dequeue(queue)
  if isEmpty(queue) then
    print("Cannot dequeue. Queue is empty.")
    return error()                                  ▷ failed
  end if
  item ← queue.items[queue.front]
  queue.size  ← queue.size - 1
  queue.front ← (queue.front + 1) mod queue.capacity
  return item                                       ▷ succeed
end procedure
```



- dequeue : 큐의 가장 앞에 있는 데이터를 삭

```
typedef struct {
    int *items;
    int front;
    int rear;
    int size;
    int capacity;
} Queue;
```

e




```
procedure dequeue(queue)
  if isEmpty(queue) then
    print("Cannot dequeue. Queue is empty.")
    return error()                          ▷ failed
  end if
  item ← ?
  queue.size ← queue.size - 1
 return item                                ▷ succeed
end procedure
```