5.1 큐 추상 데이터 타입

큐 선입선출 ex) 매표소의 줄,

뒤에서 새로운 데이터 추가, 앞에서 데이터가 하나씩 삭제

후단 삽입이 일어나는 곳

전단 삭제가 일어나는 곳

ADT5.1 객체: 0개 이상의 요소들로 구성된 선형 리스트

연산:

create(max\_size)::= 최대 크기가 max\_size인 공백 큐를 생성한다

init(q) ::= 큐를 초기화한다

is\_empty(q)::=

if(size ==0) return true;

else return false;

is\_full(q)::=

if(size==max\_size) return true;

else return false;

//스택과는 달리 양 끝에서 삽입과 삭제 --> top, rear 라는 2개의 변수 존재

enqueue(q,e)::=

if(is\_full(q)) queue\_full 오류;

else q의 끝에 e를 추가한다

dequeue(q)::=

if(is\_empty(q)) queue\_empty 오류;

else q의 맨 앞에 있는 e를 제거해 반환한다

peek(q)::=

if(is\_empty(q) queue\_empty 오류;

else q의 맨 앞에 있는 e를 읽어서 반환한다

추상자료형 스택과 매우 유사

CPU와 주변기기의 속도 차이를 메꾸기 위해 큐 사용

Quiz 1. 선입선출

2. ‘a’

5.2 선형큐

1차원 배열을 이용한 큐 구현

삽입을 위한 변수 rear

큐의 마지막 요소

초기값 -1

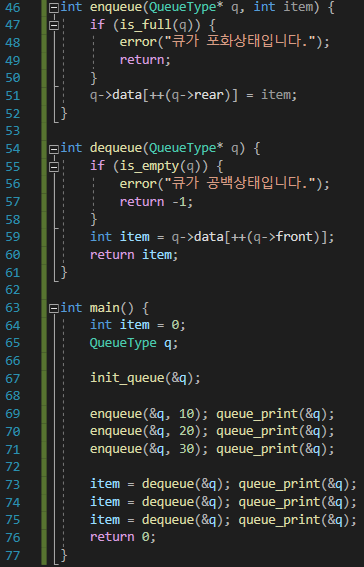
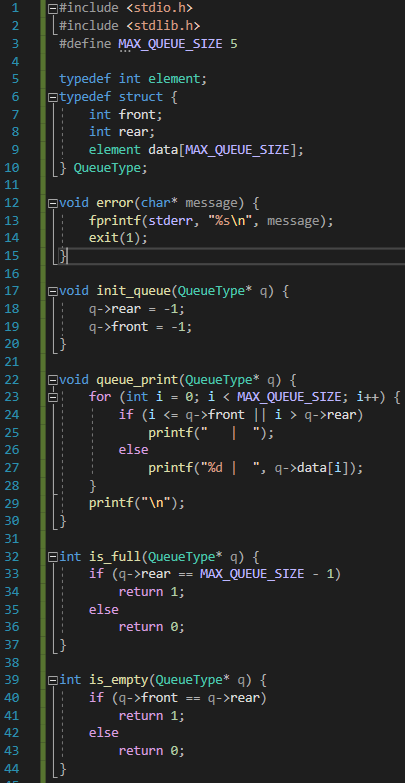
데이터가 들어오면 증가

삭제를 위한 변수 front

큐에 첫 번째 요소

초기값 -1

삭제할 때 증가

선형큐의 구현 프로그램5.1 

선형 큐의 응용: 작업 스케줄링 작업들이 운영체제에 들어간 순서대로 처리됨 🡪 큐 이용

Quiz 우선순위 고려 불가

5.3 원형 큐

선형 큐의 문제점 front, rear 값이 계속 증가 🡪 배열 끝에 언젠가 도달

🡪 배열 앞 부분이 비어 있어도 사용 불가 🡪 주기적으로 모든 요소들을 왼쪽으로 이동시켜야 함

원형 큐 front, rear의 값이 배열의 끝에 도달하면 다음의 인덱스는 0으로 함

front 초기값 0

첫번째 요소의 하나 앞(비어있음)

rear 초기값 0

마지막 요소

삽입 rear 증가 후, 증가된 위치에 데이터 삽입

삭제 front 증가 후, 증가된 위치에 데이터 삭제

공백상태, 포화상태 구별 공백상태 front==rear

포화상태 front == rear+1

공백, 포화상태를 구별하기 위해 한 자리를 꼭 비워둬야 함

원형 큐의 삽입, 삭제 알고리즘 알고리즘 5.1 enqueue(Q, x) :

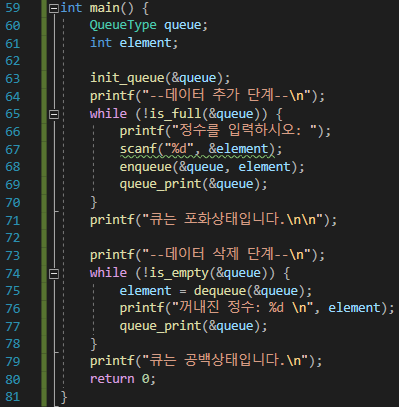
rear < --(rear + 1) % MAX\_QUEUE\_SIZE;

Q[rear] < --x;

알고리즘 5.2 dequeue(Q) :

front < --(front + 1) % MAX\_QUEUE\_SIZE;

return Q[front];

원형 큐의 구현 프로그램5.2  

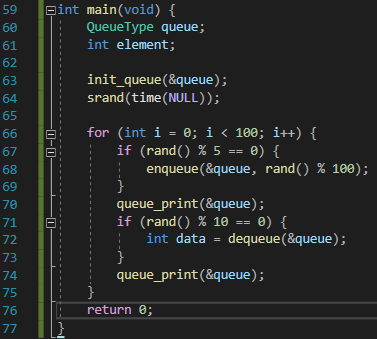
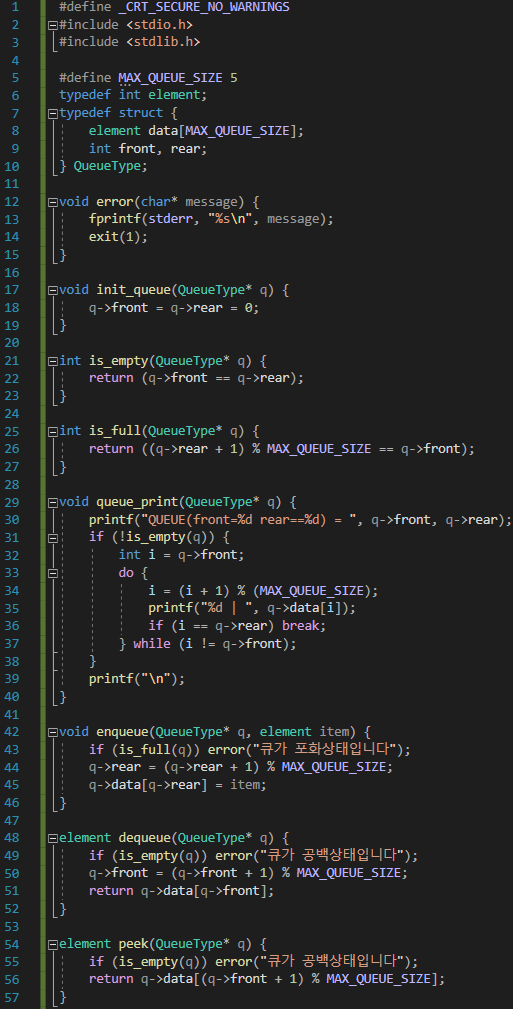
Quiz 1. front = 데이터가 삭제되는 곳, rear = 데이터가 삽입되는 곳

2. 1, 4

5.4 큐의 응용: 버퍼

버퍼 서로 다른 속도로 실행되는 두 프로세스 간의 상호작용을 조화시킴

큐로 제작

프로그램5.3 

5.5 덱이란?

덱 double-ended queue

큐의 전단과 후단에서 모두 삽입∙삭제 가능

덱의 추상 자료형 ADT5.2 객체: n개의 element형의 요소들의 순서있는 모임

연산:

create() ::= 덱 생성

init(dq) ::= 덱 초기화

is\_empty(dq) ::= 덱이 공백상태인지 검사

is\_full(dq) ::= 덱이 포화상태인지 검사

add\_front(dq, e) ::= 덱의 앞에 요소를 추가

add\_rear(dq, e) :: = 덱의 뒤에 요소를 추가

delete\_front(dq) ::= 덱의 앞에 있는 요소를 반환한 다음 삭제

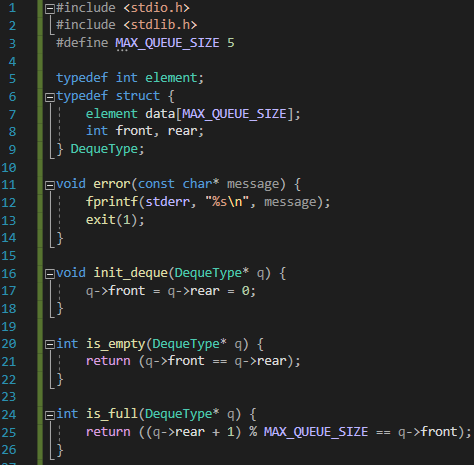
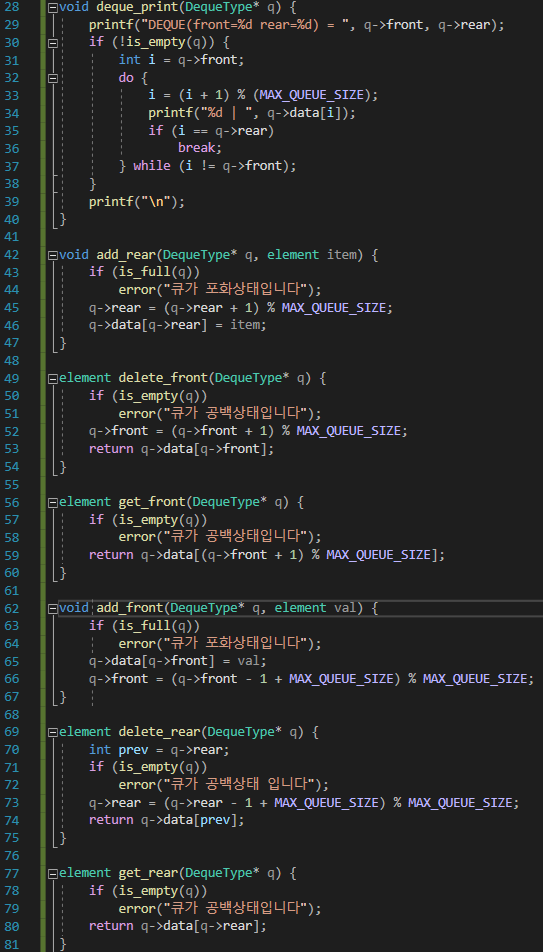
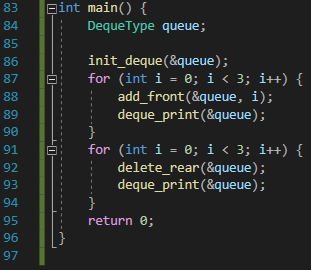
delete\_rear(dq) ::= 덱의 뒤에 있는 요소를 반환한 다음 삭제

get\_front(q) ::= 덱의 앞에서 삭제하지 않고 앞에 있는 요소 반환

get\_rear(q) ::= 덱의 뒤에서 삭제하지 않고 뒤에 있는 요소를 반환

배열을 이용한 덱의 구현 원형 큐의 확장

원형 큐의 배열, front, rear 그대로 사용

프로그램5.4   

연결된 덱의 구현 연결리스트로 구현 가능

하나의 노드에서 알아야 할 정보가 더 많음

Quiz 1. add\_rear, delete\_front

2. add\_front, delete\_front

5.6 큐의 응용: 시뮬레이션

시스템의 특성을 시뮬레이션해 분석하는데 큐 이용

큐잉 모델 서비스를 수행하는 서버와 서비스를 받는 고객들로 구성

은행 서비스 시뮬레이션 1. 현재 시각을 나타내는 clock 변수 하나 증가

2. [0,10]사이의 난수를 생성해 3보다 작으면 새로운 고객이 들어왔다고 판단, 새로운 고객이 들어오면 구조체를 생성하고 고객의 아이디, 도착시간, 서비스시간등 정보 복사

enqueue()를 호출해 큐에 추가

전역변수인 sevice\_time에 현재 처리 중인 고객의 서비스 시간 저장

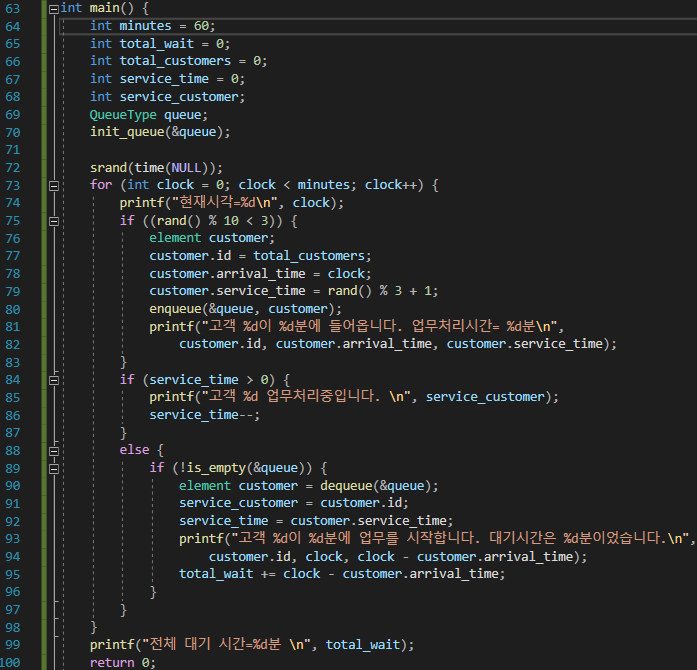
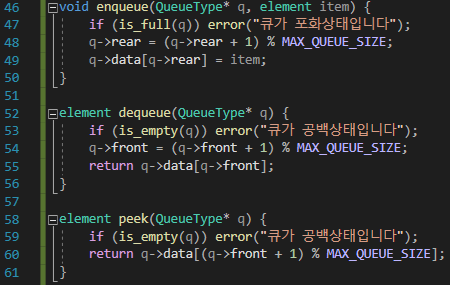
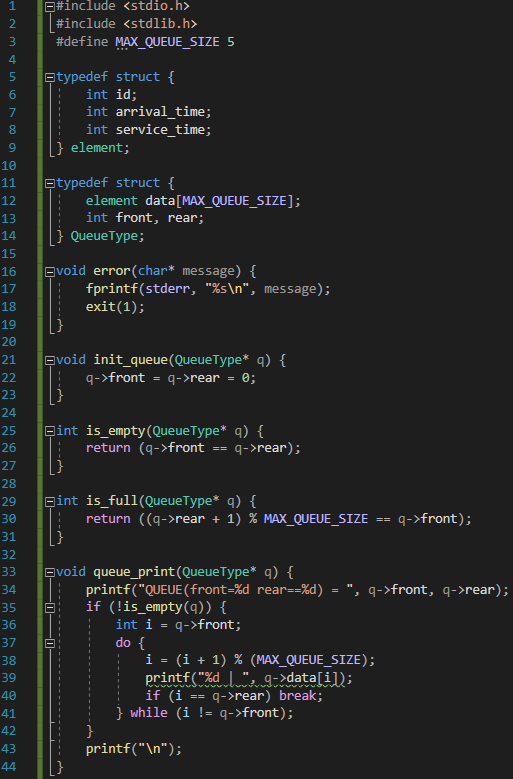
3. sevice\_time이 0이 아닌지 살펴봄

만약 service\_time이 0이 아니면 어떤 고객이 지금 서비스를 받고 있는 중

clock이 하나 증가했으므로, service\_time을 하나 감소시킴

service\_time이 0이면 현재 서비스 받는 고객이 없다는 것 🡪 큐에서 고객 구조체를 하나 꺼내어 서비스 시작 = service\_time에 고객의 서비스 시간 저장

4. 60분의 시간이 지나면 고객들이 기다린 시간을 전부 합하여 화면에 출력

프로그램5-5 

도전문제 큐를 두개를 만들고, 쓰레드를 하나 더 만들어 처리

연습문제

1. a

2. b

3. 40 50

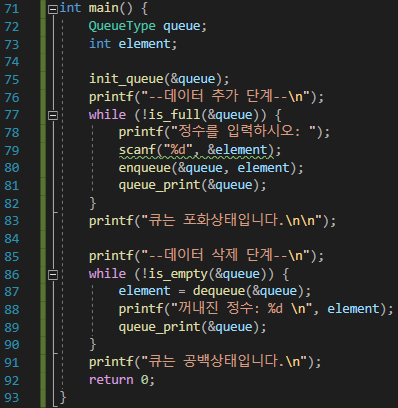
4. c

5. 2

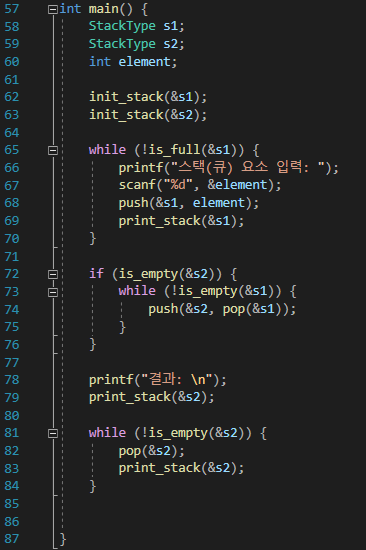
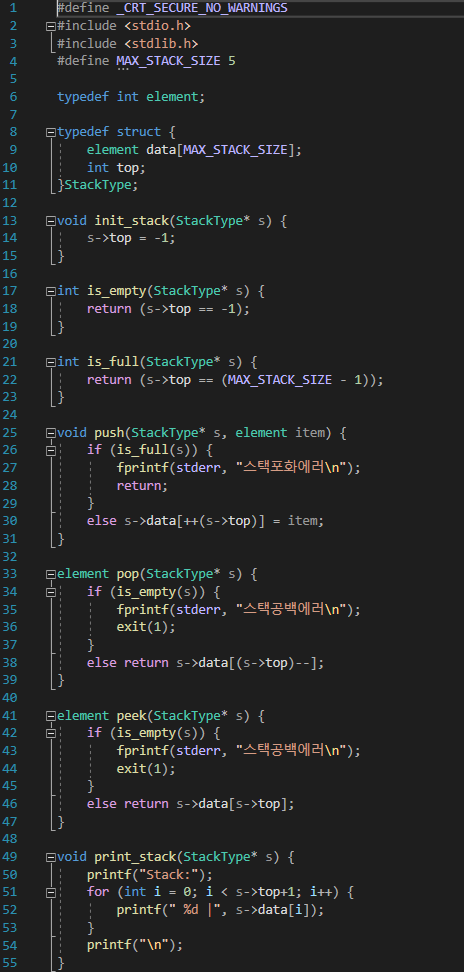
6. cad

7. a

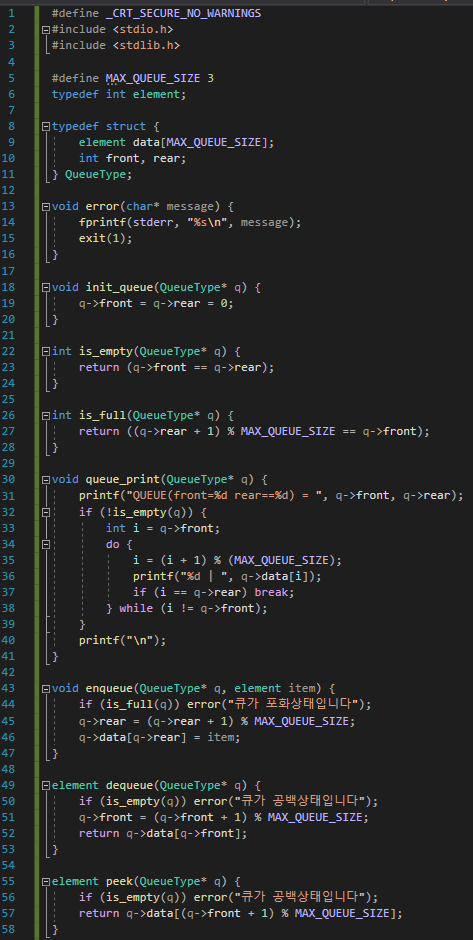
8.



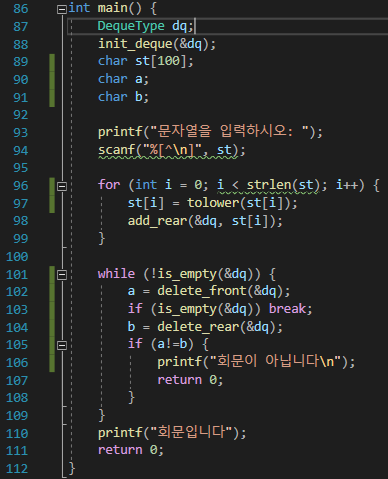
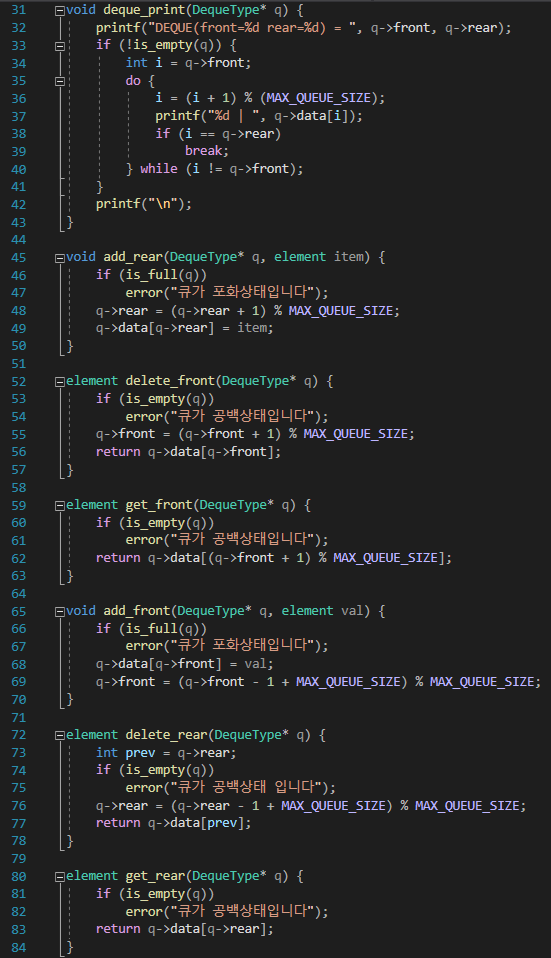
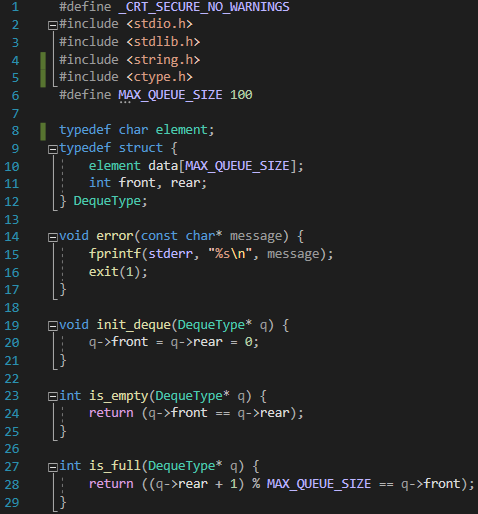
9.



10.



11.



12.

