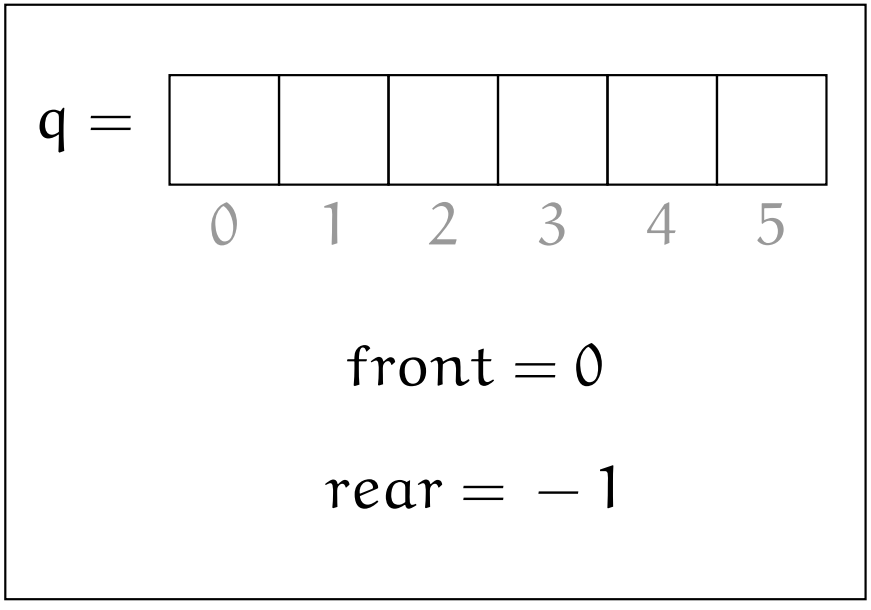
Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

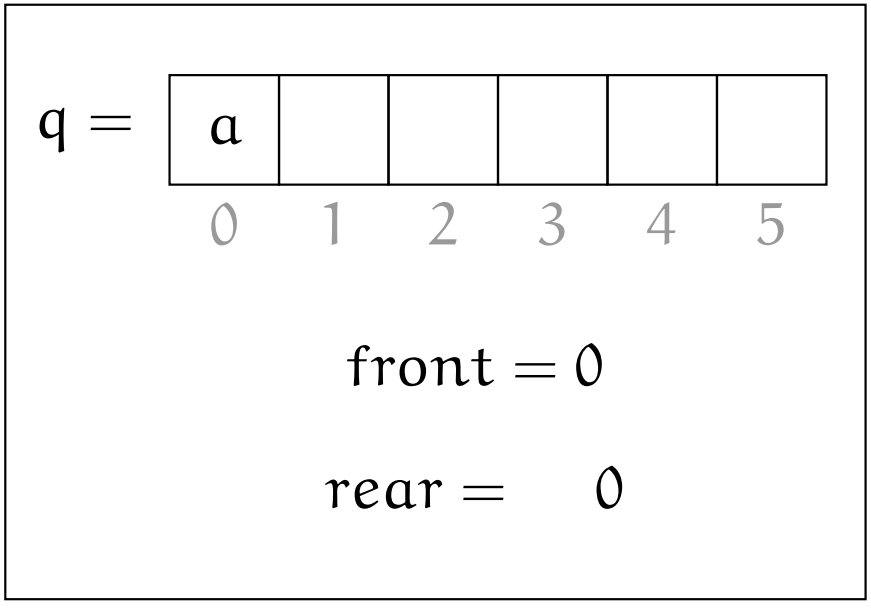
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

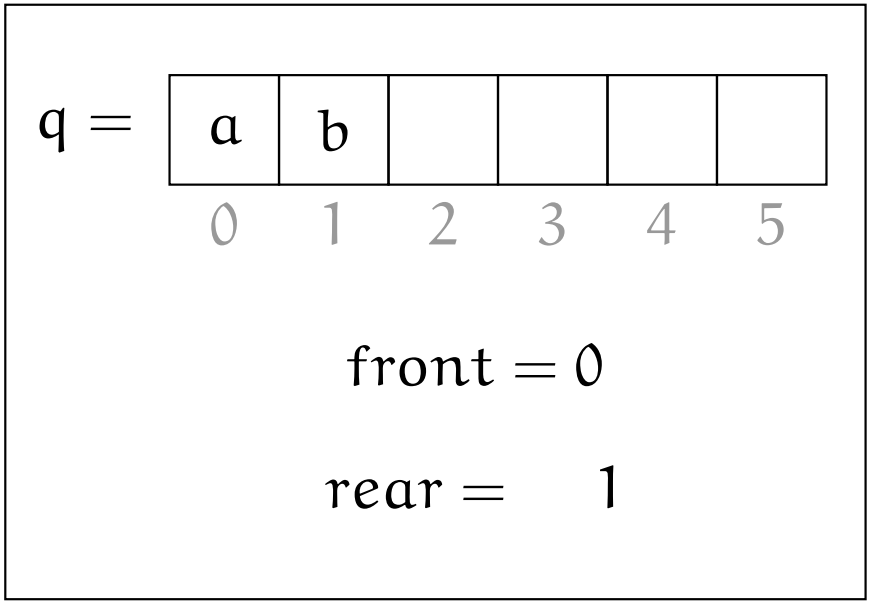
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

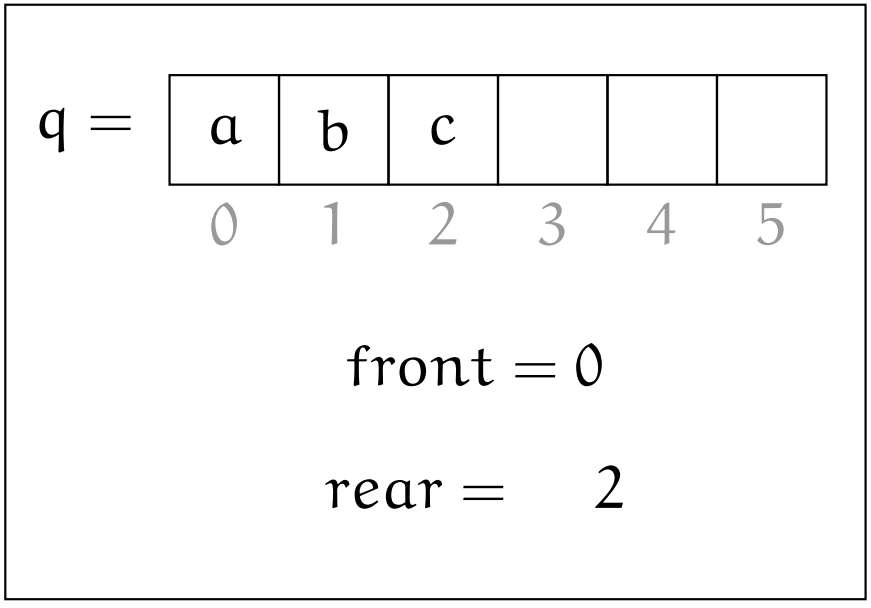
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

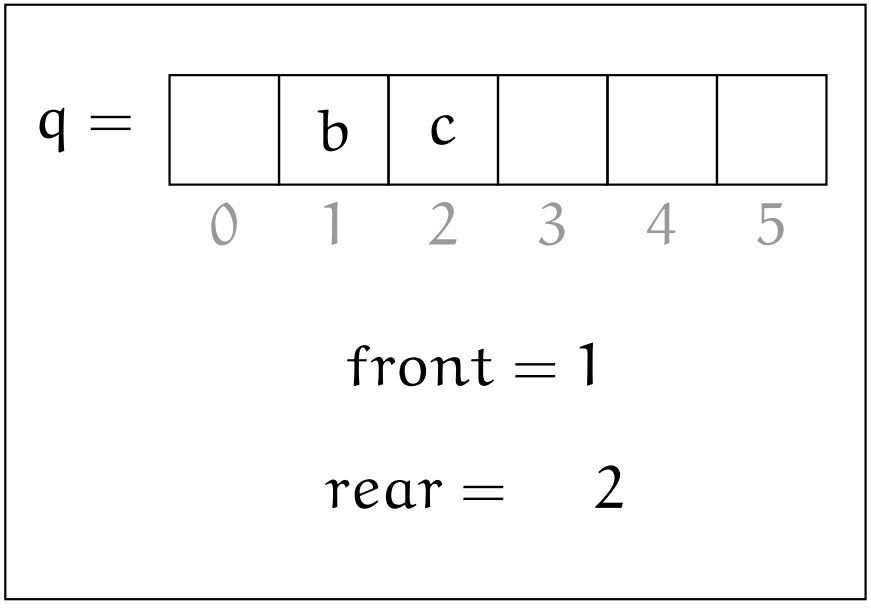
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

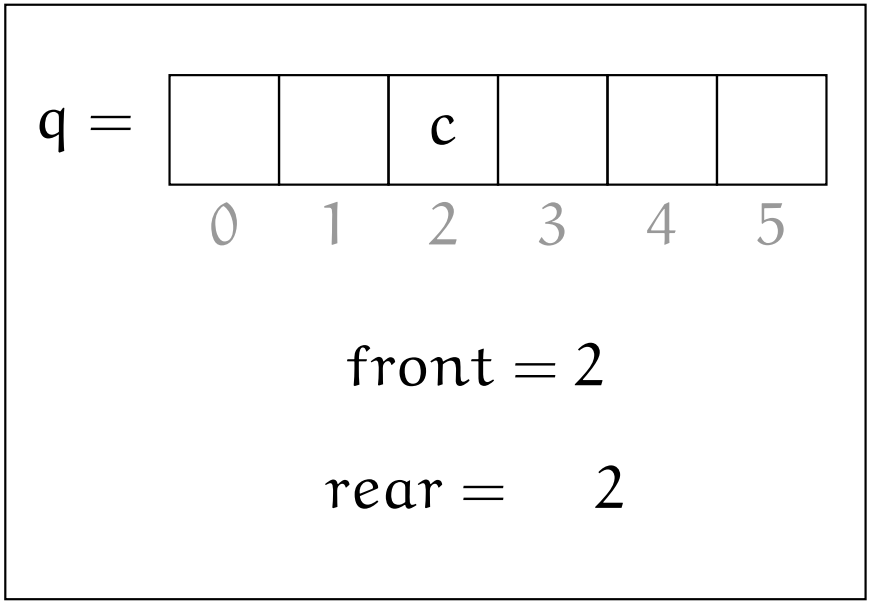
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : a

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

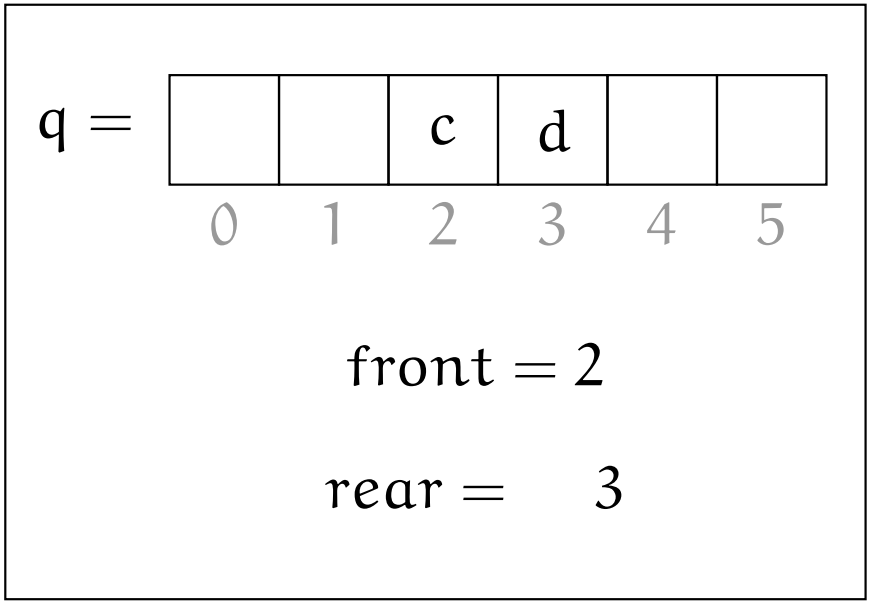
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

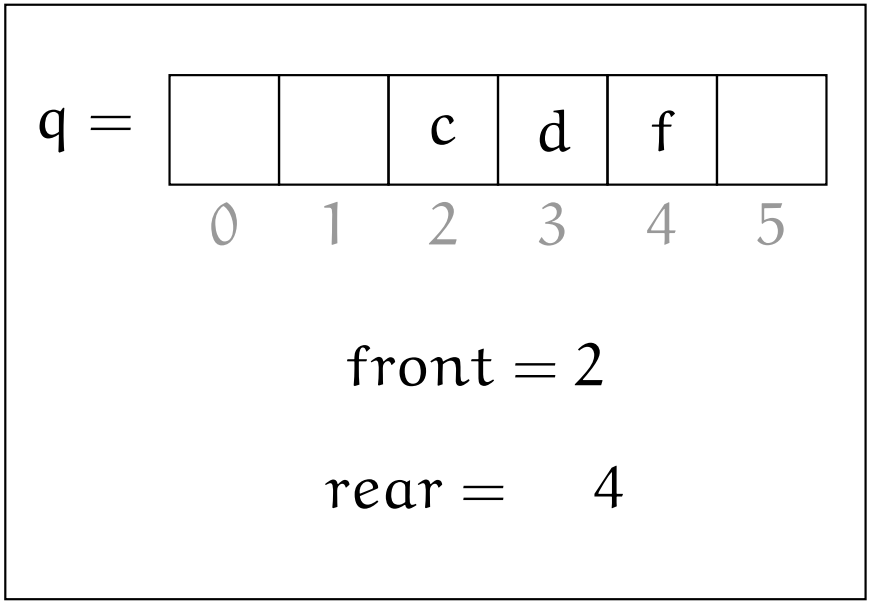
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

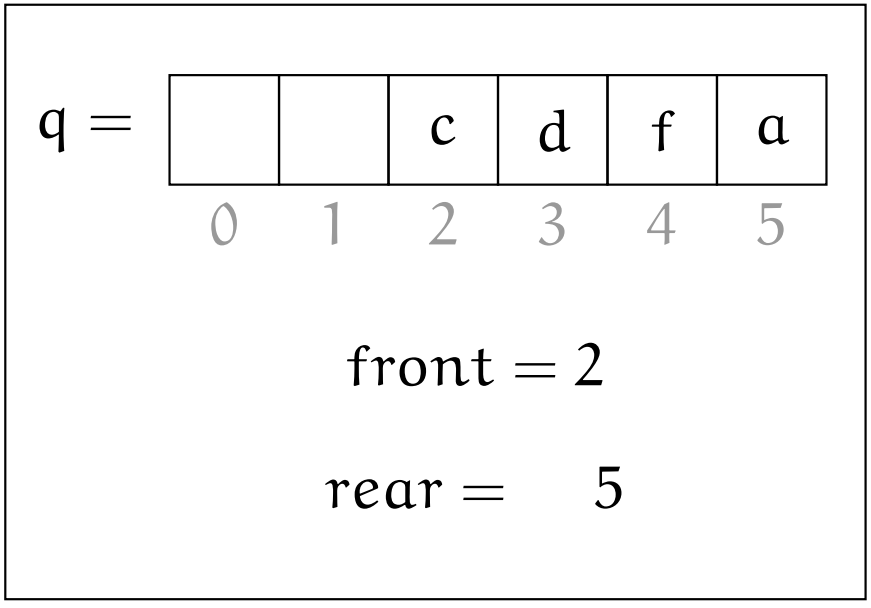
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

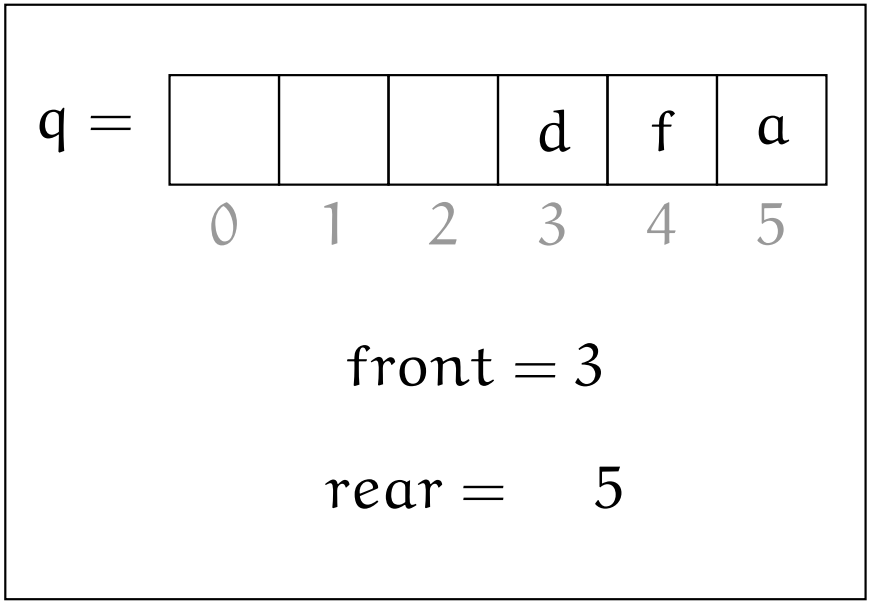
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

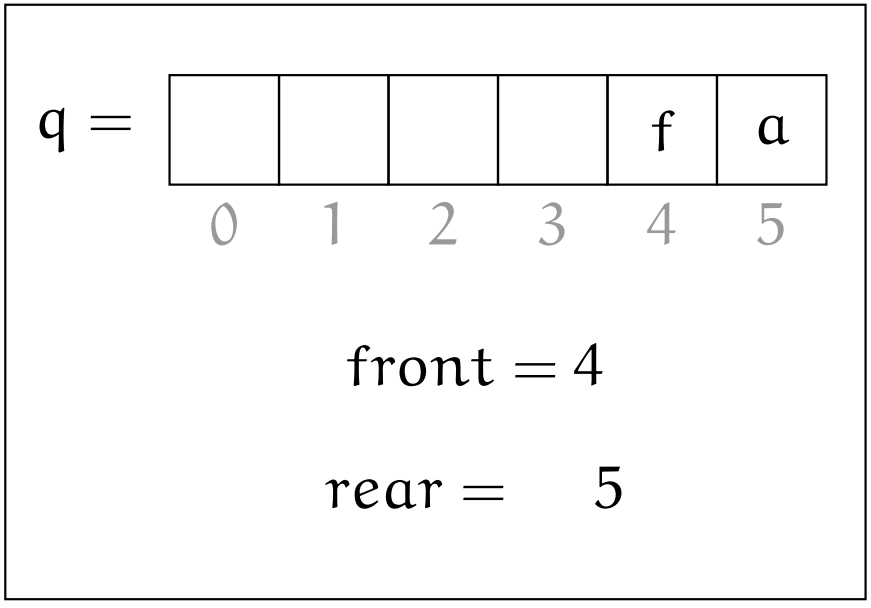
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : abc

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

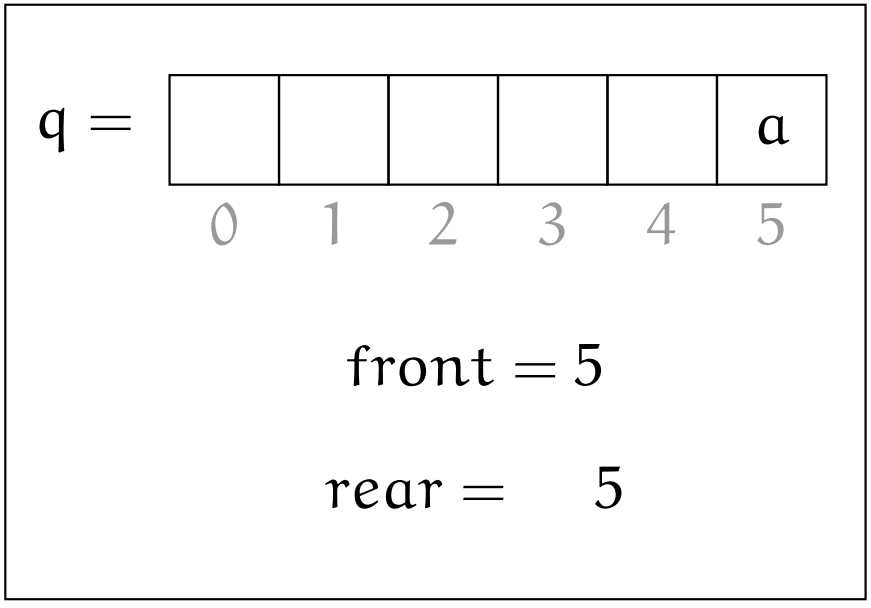
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : abcd

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

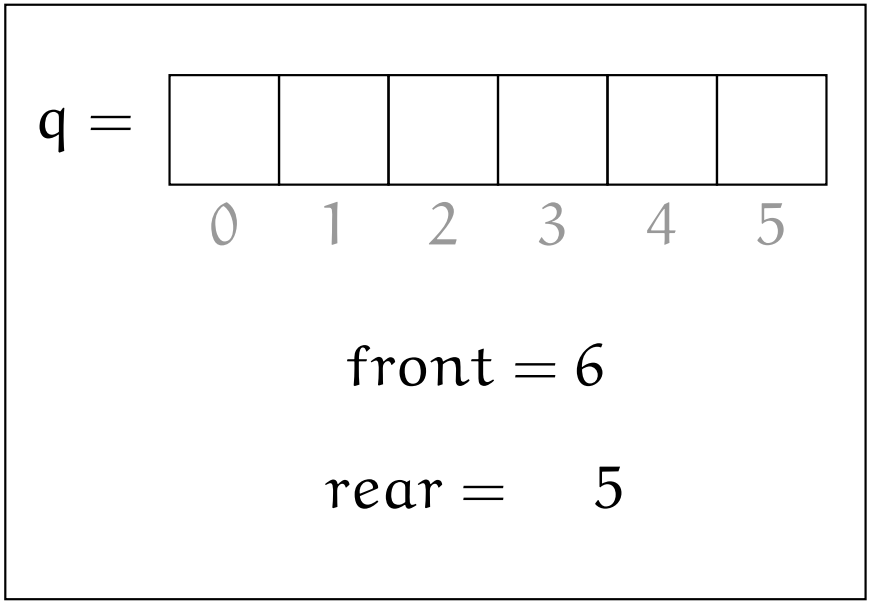
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : abcdf

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

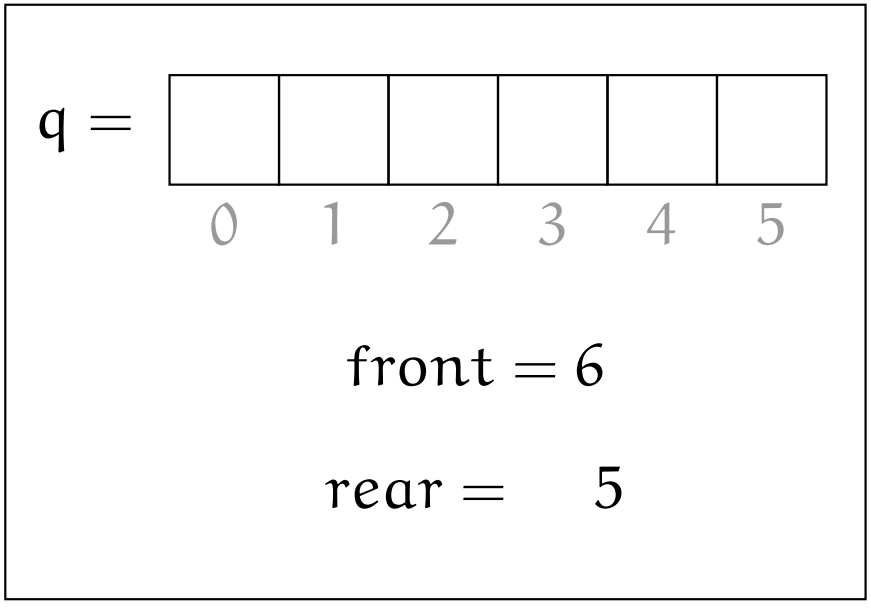
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : abcdfa

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

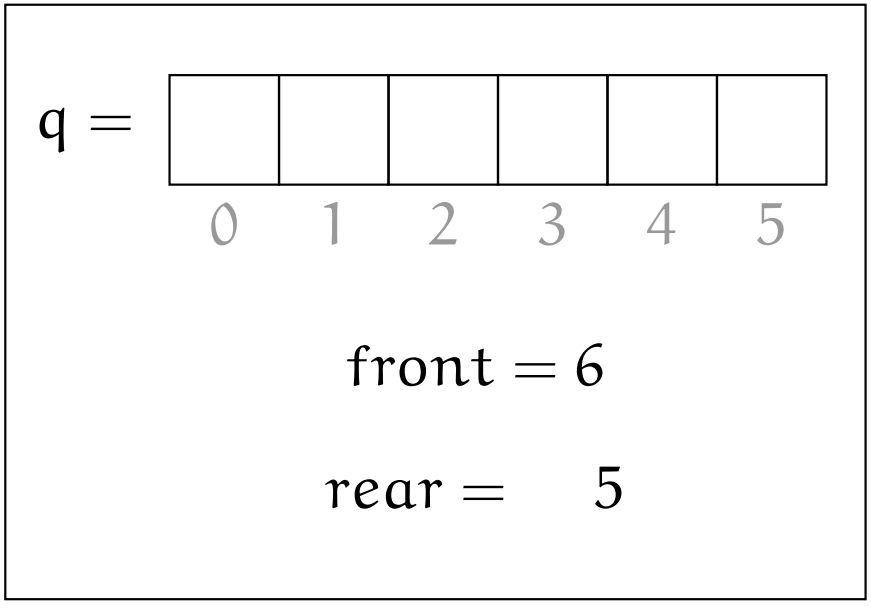
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPP

Output : abcdfaE

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Empty

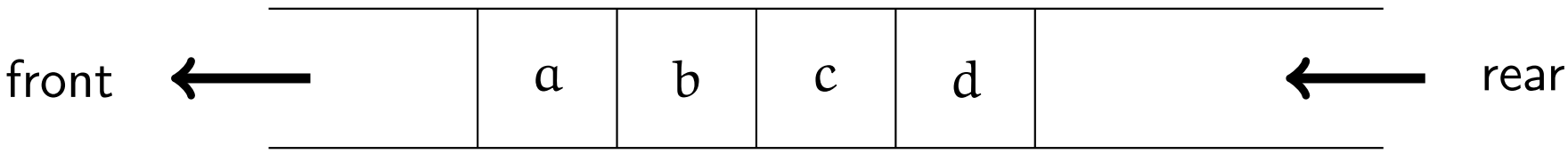
```
front > rear
```

Input : abcPPdfaPPPPP

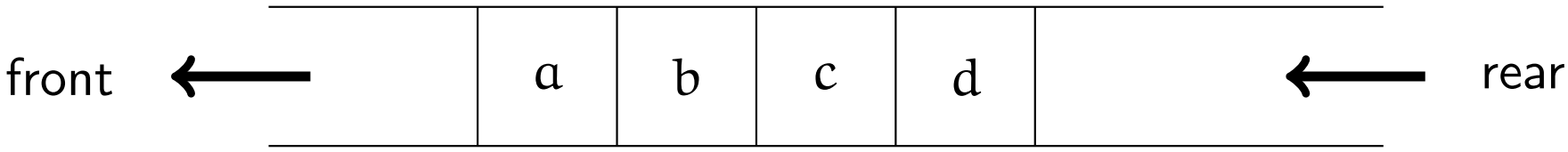
Output : abcdfaE

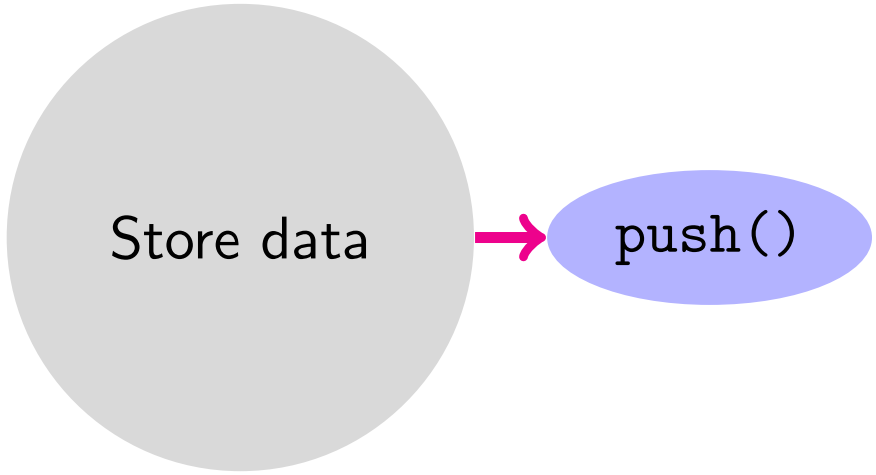
Data Structures
- Queues

Data Structures
- Queues

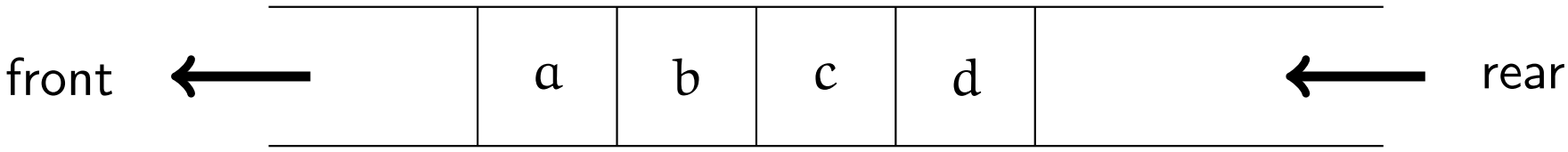


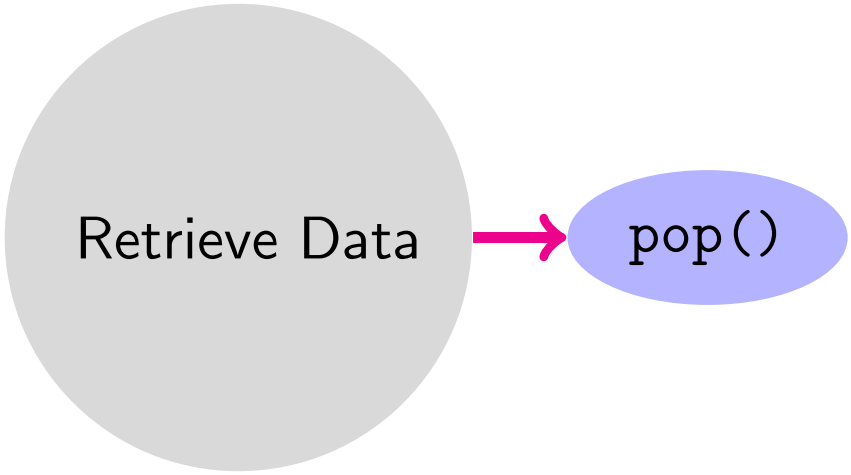
Data Structures - Queues





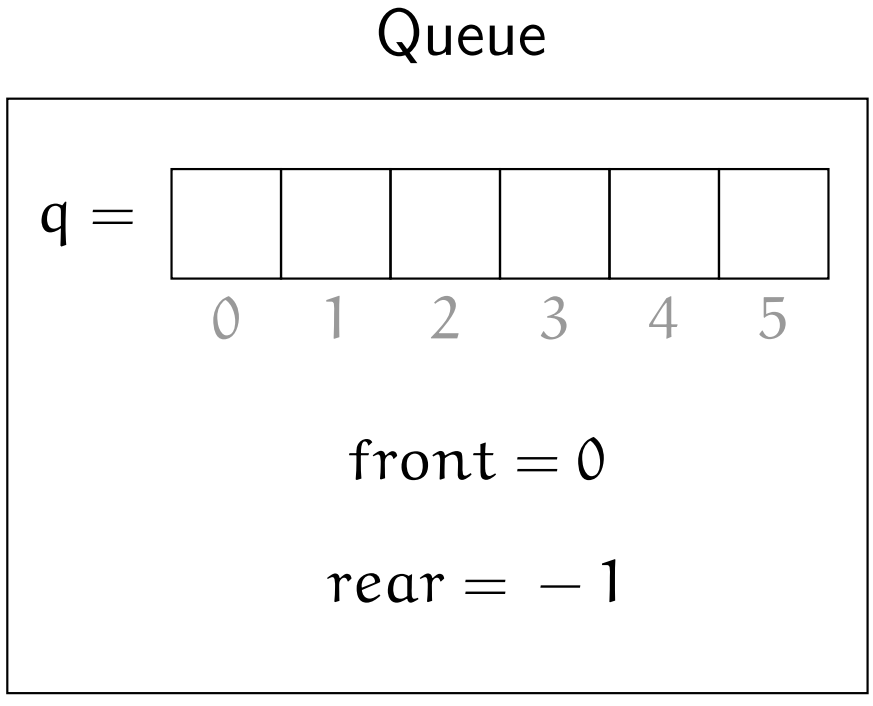
Data Structures - Queues





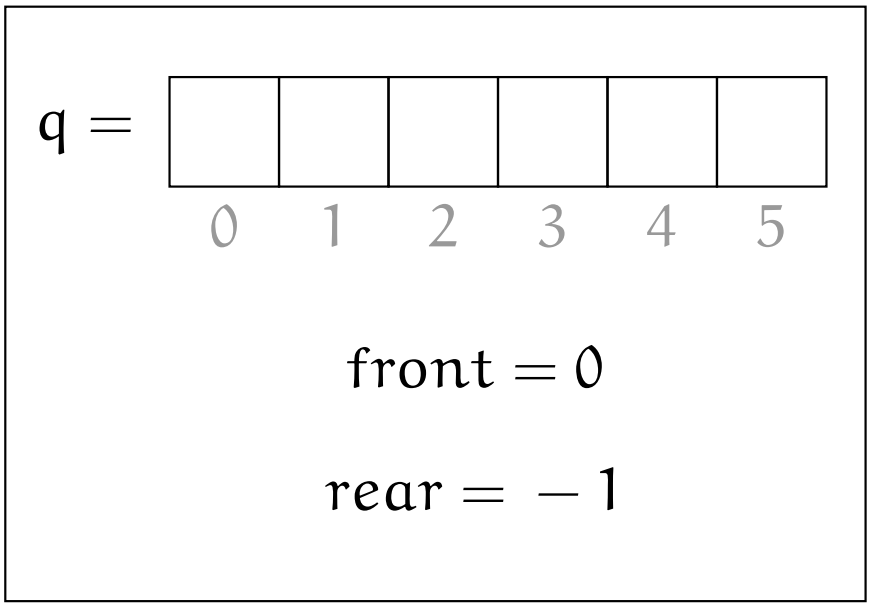
Queues
- Implementation

Queues
- Implementation



Queues
- Implementation

Queue

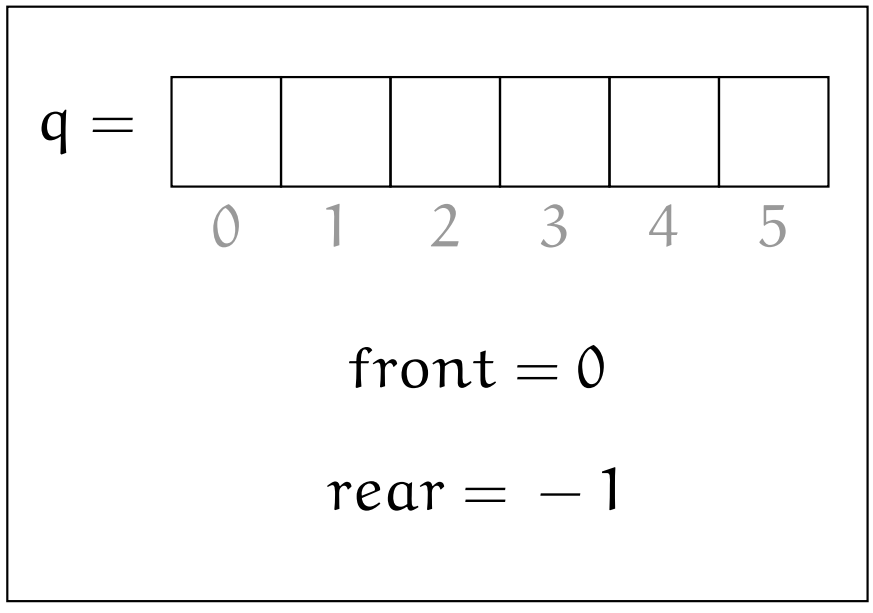


Push

```
rear++;
q[rear] = a;
```

Queues
- Implementation

Queue



Push

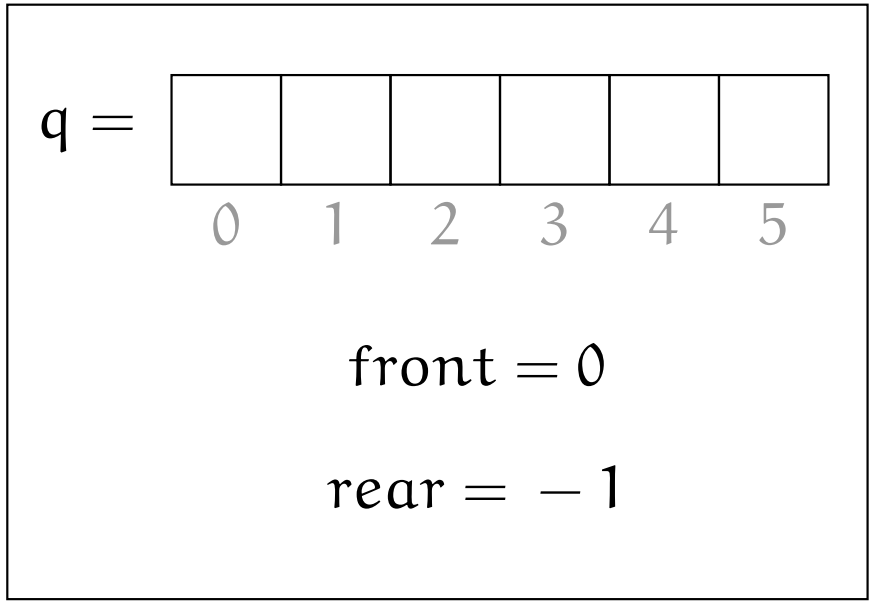
```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

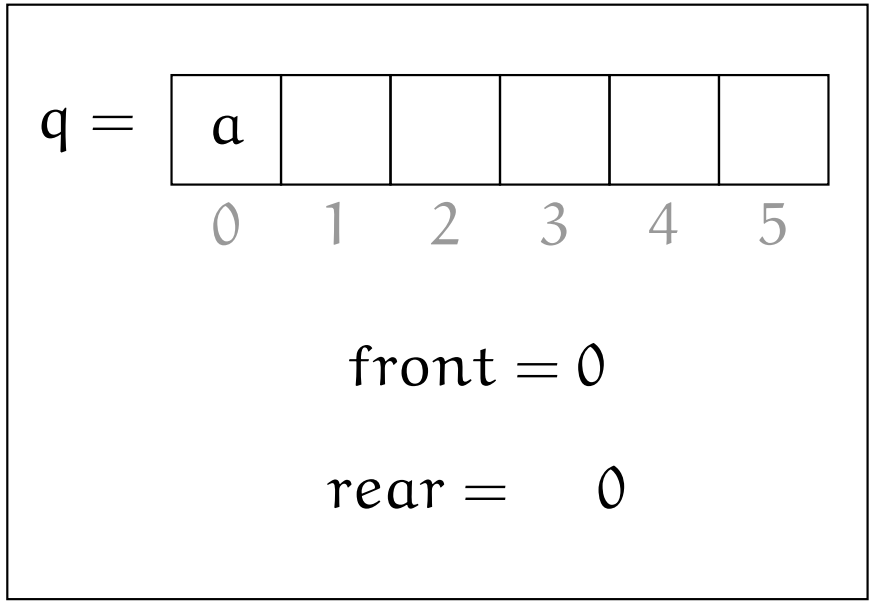
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

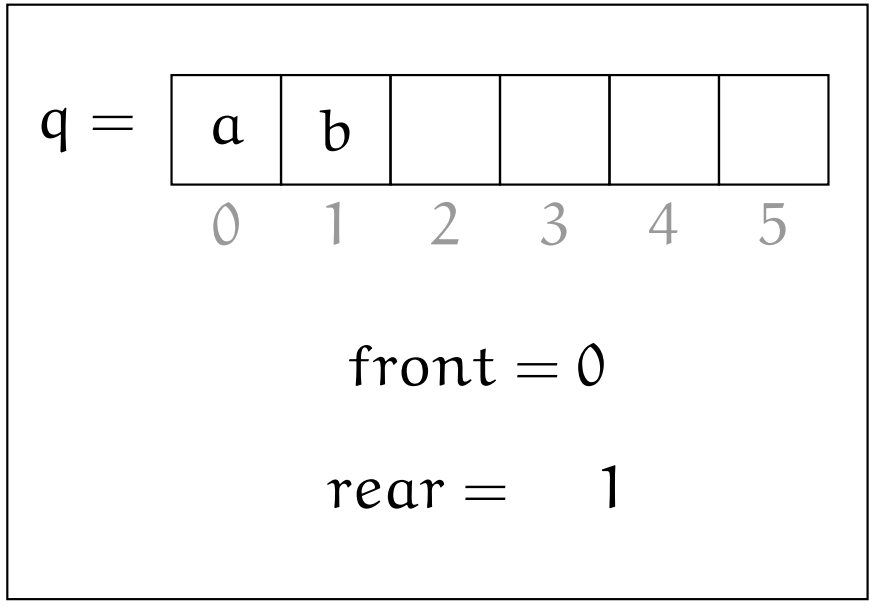
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

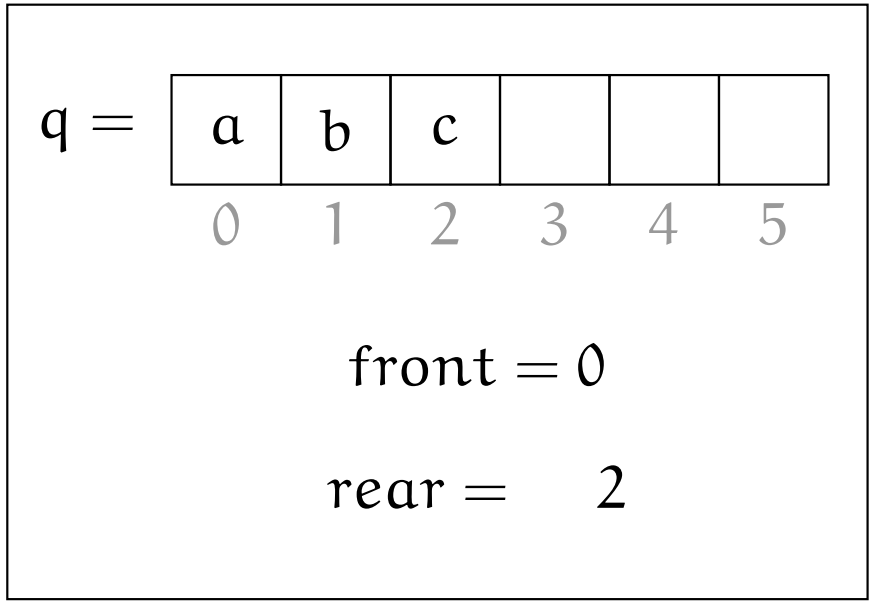
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

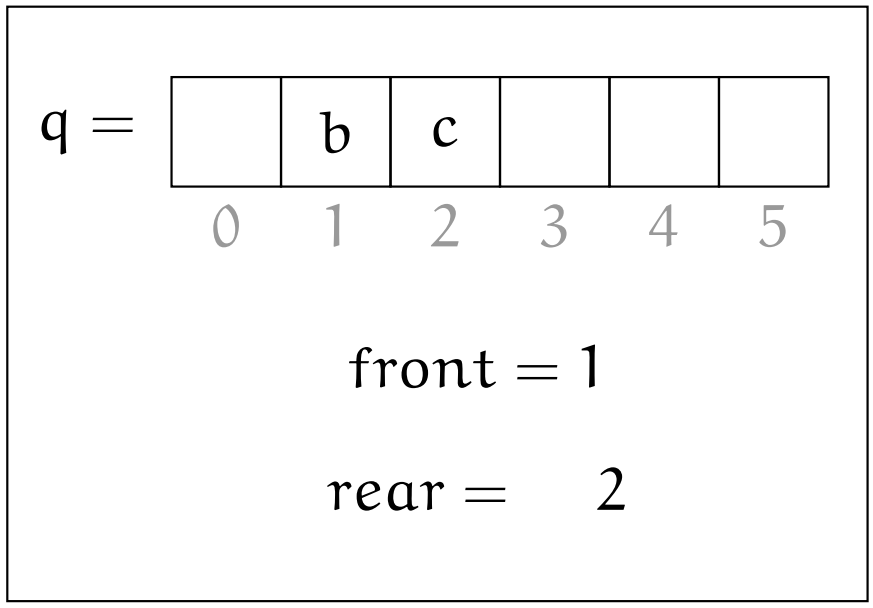
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output :

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

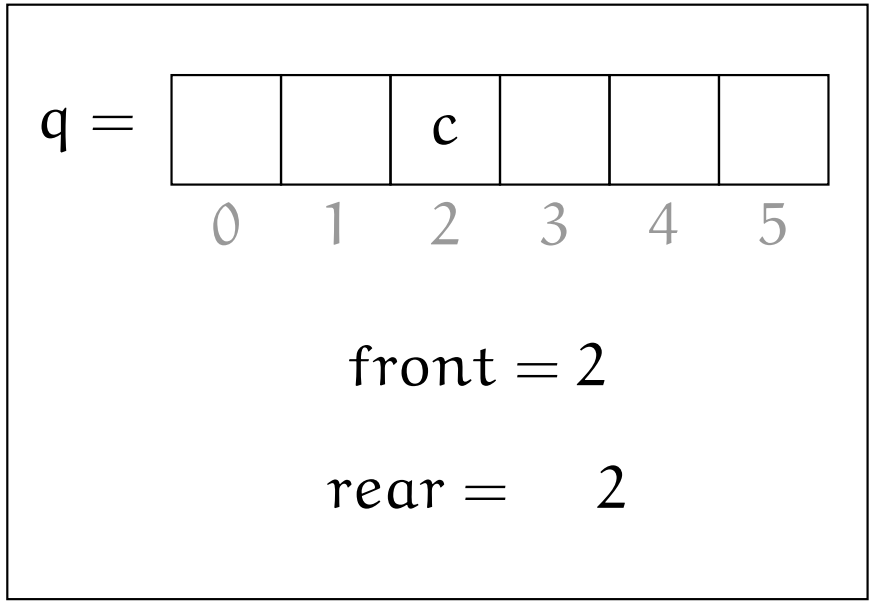
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : a

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

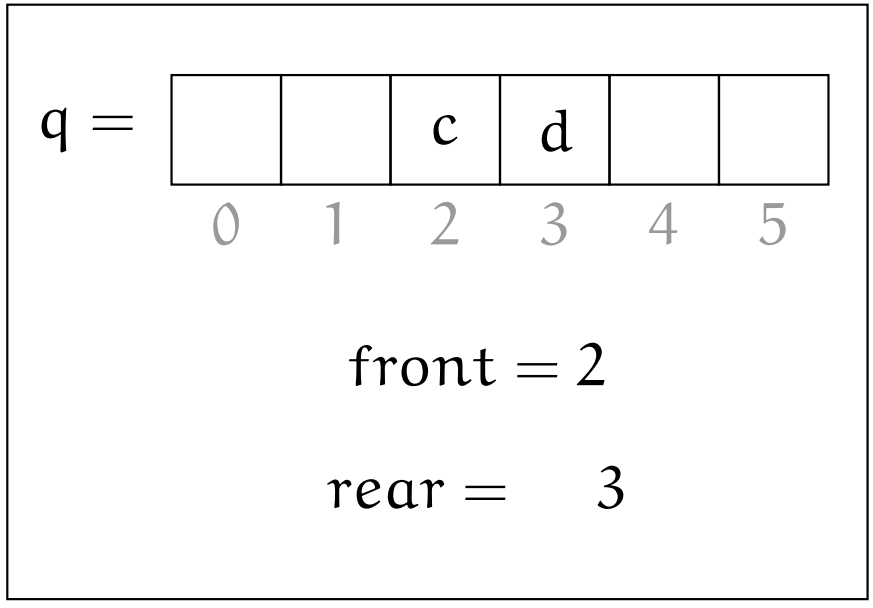
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

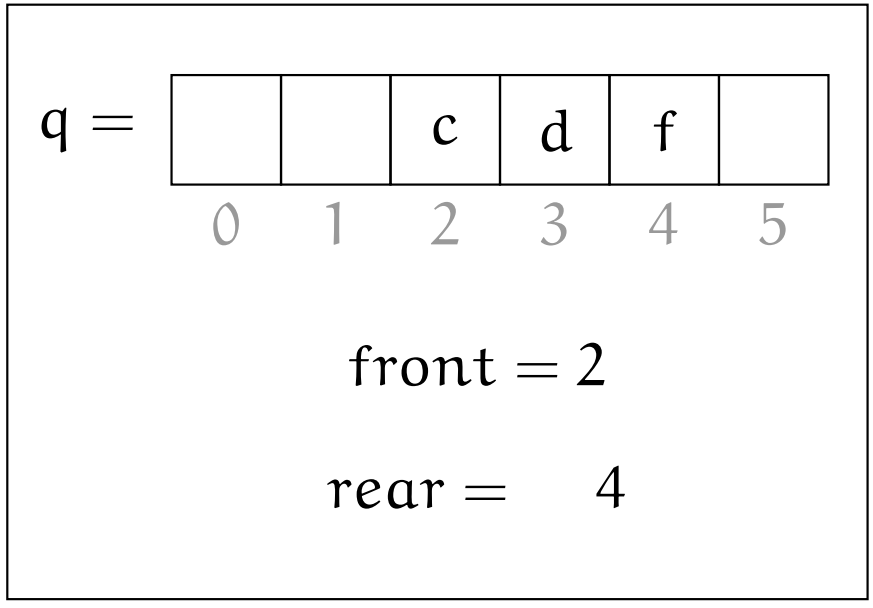
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

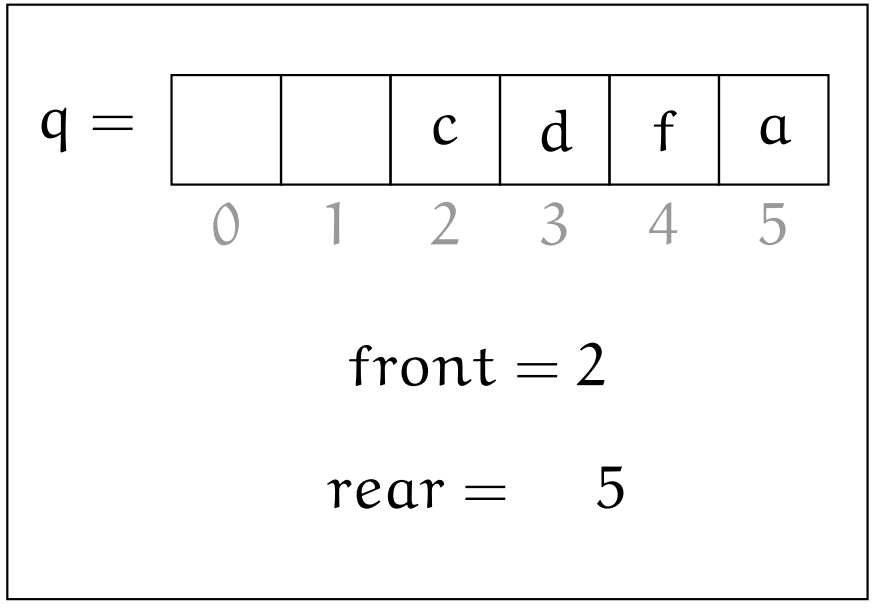
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

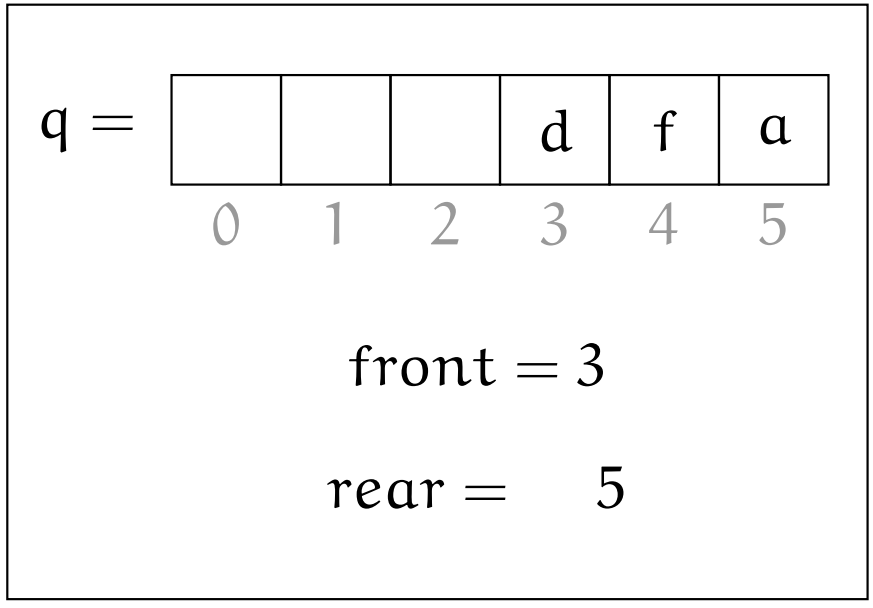
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : ab

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

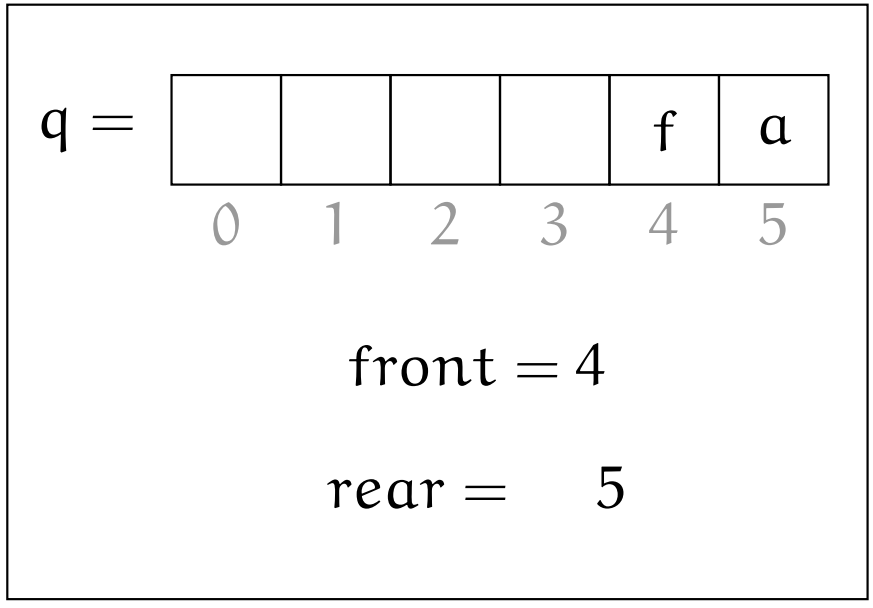
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abc

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

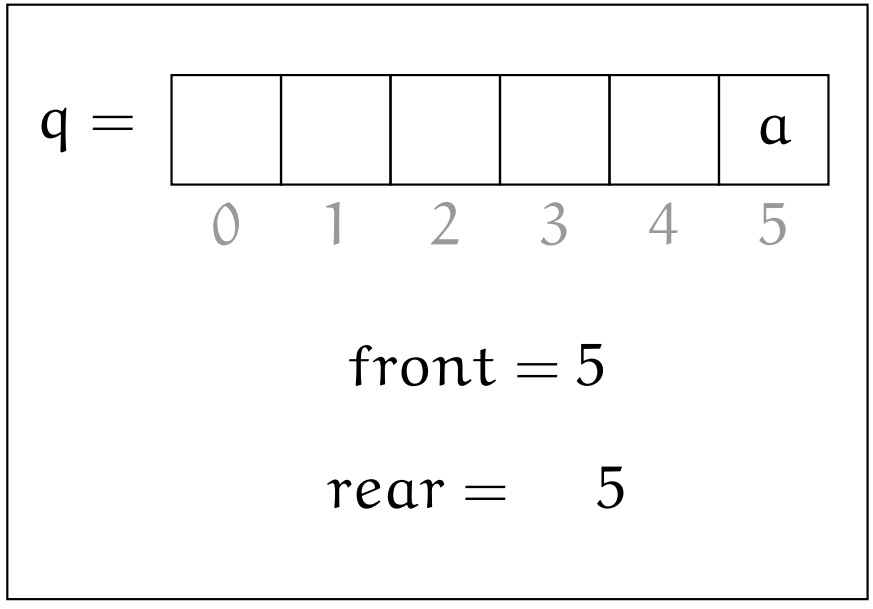
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcd

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

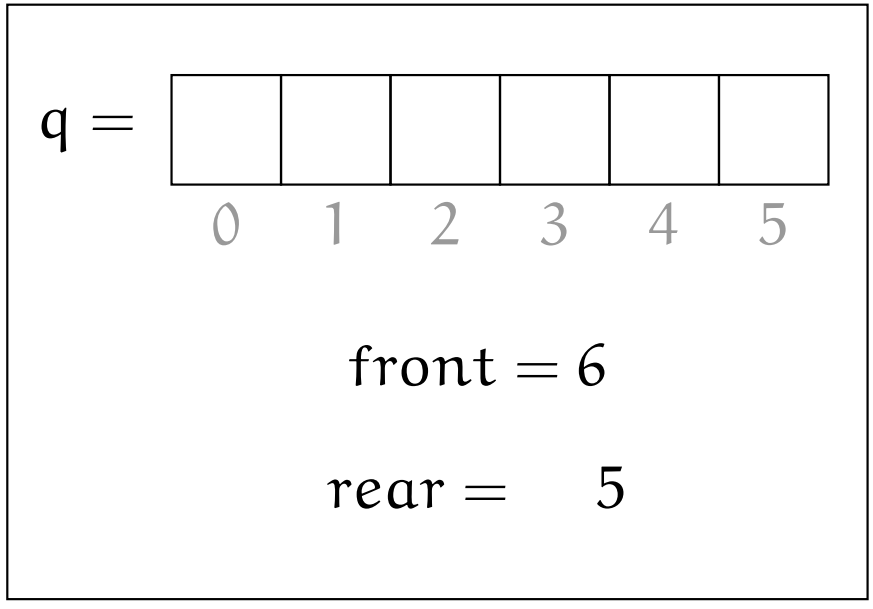
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdf

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

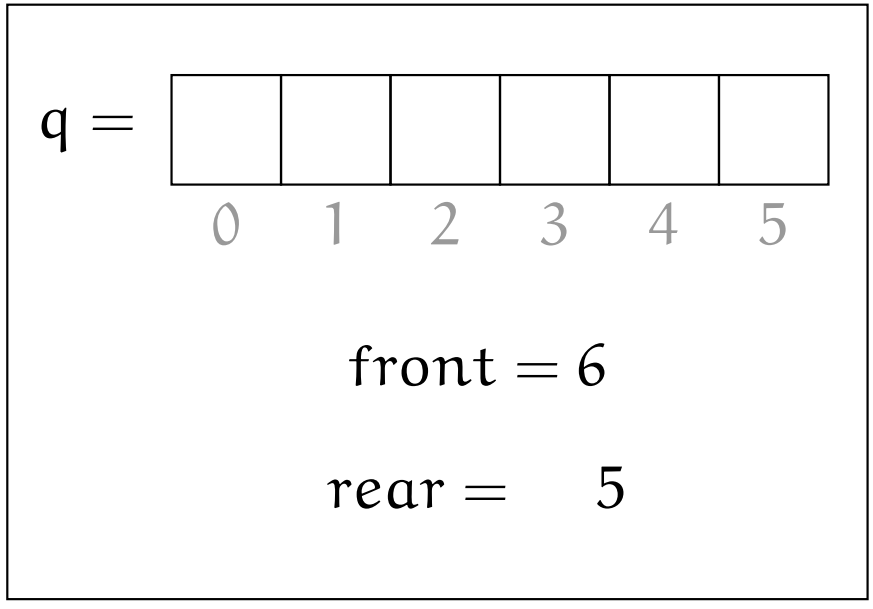
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfa

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

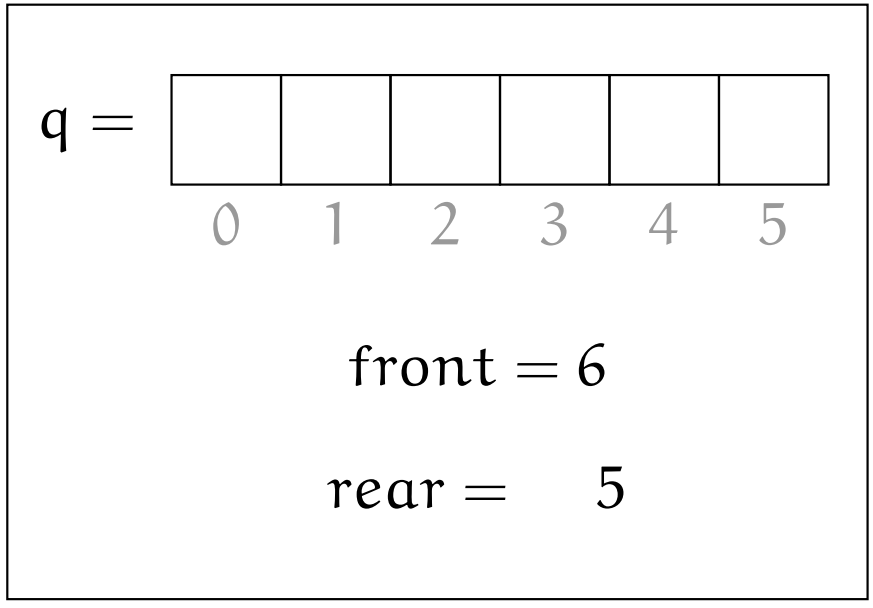
```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Empty

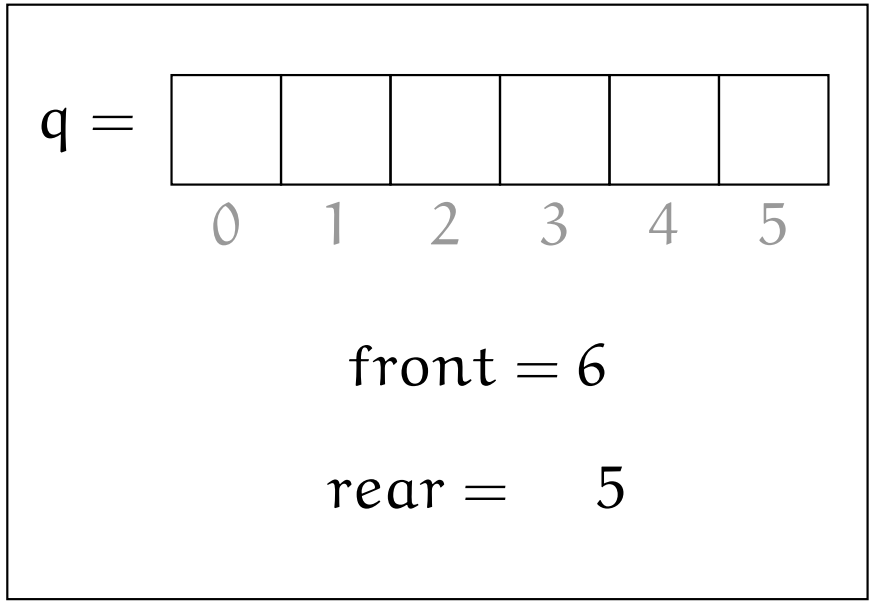
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Full

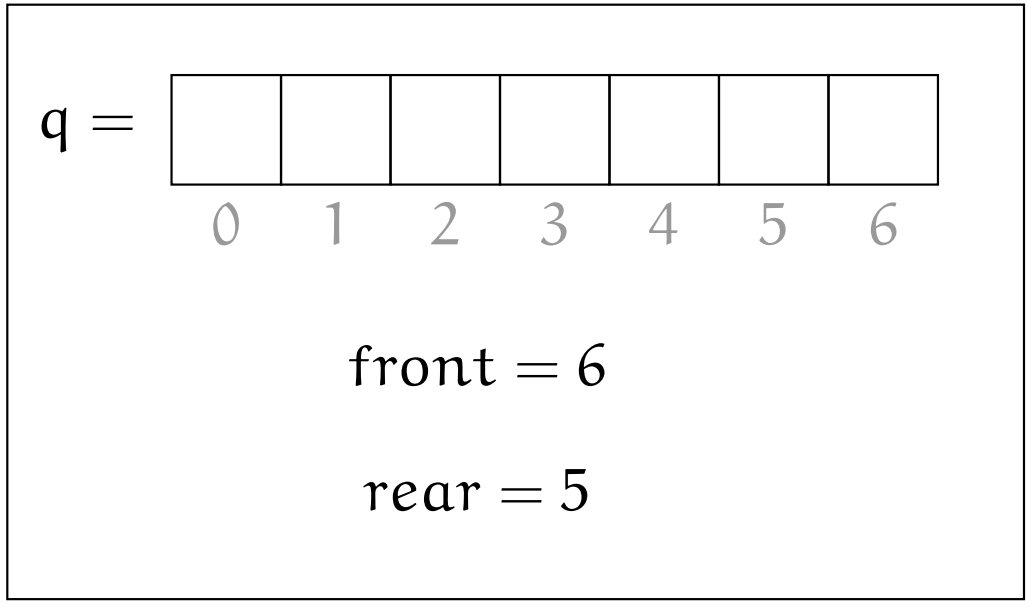
```
rear == n - 1
```

Empty

```
front > rear
```

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Full

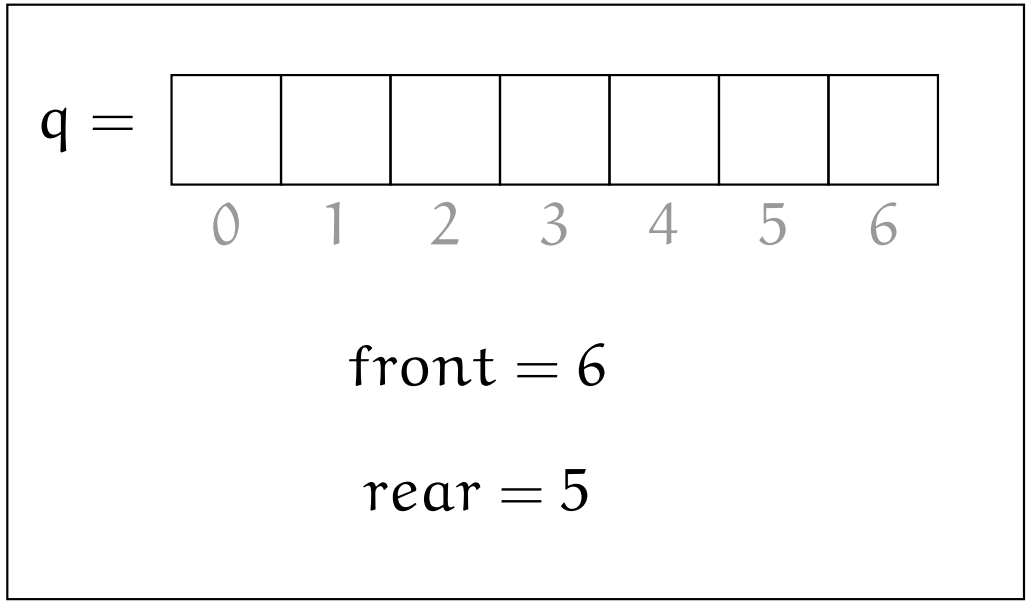
```
rear == n - 1
```

Empty

```
front > rear
```

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Full

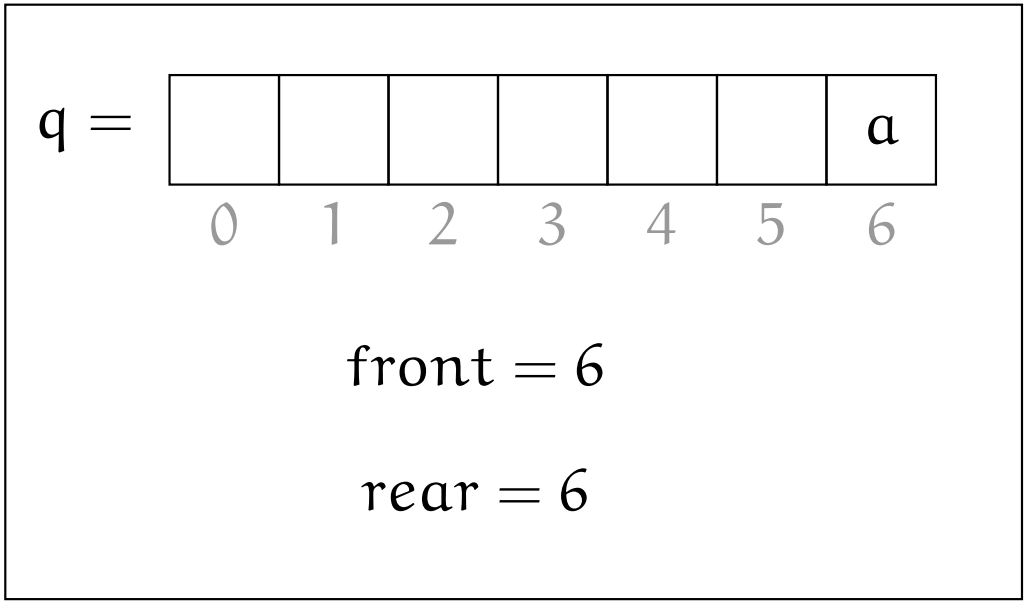
```
rear == n - 1
```

Empty

```
front > rear
```

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Full

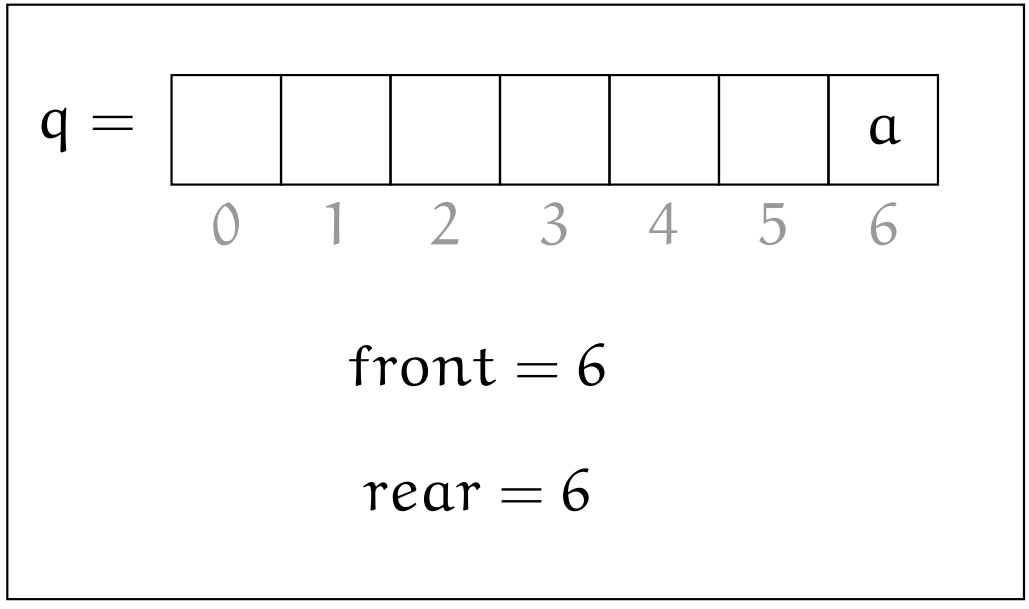
```
rear == n - 1
```

Empty

```
front > rear
```

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Full

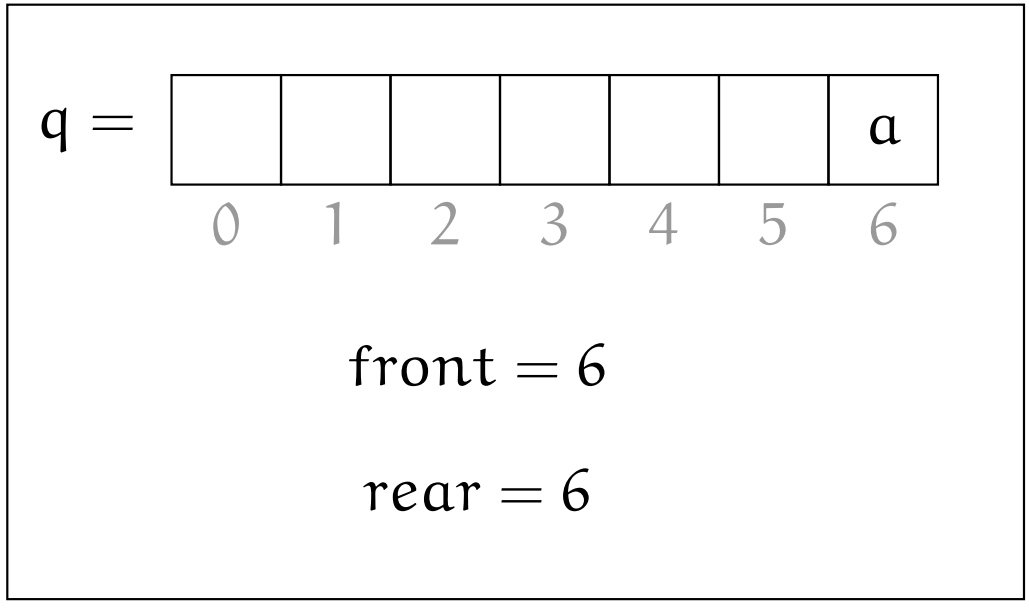
```
rear == n - 1
```

Empty

```
front > rear
```

Queues
- Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

```
t = q[front];
front++;
```

Input : abcPPdfaPPPPPabcPabcd

Output : abcdfaE

Full

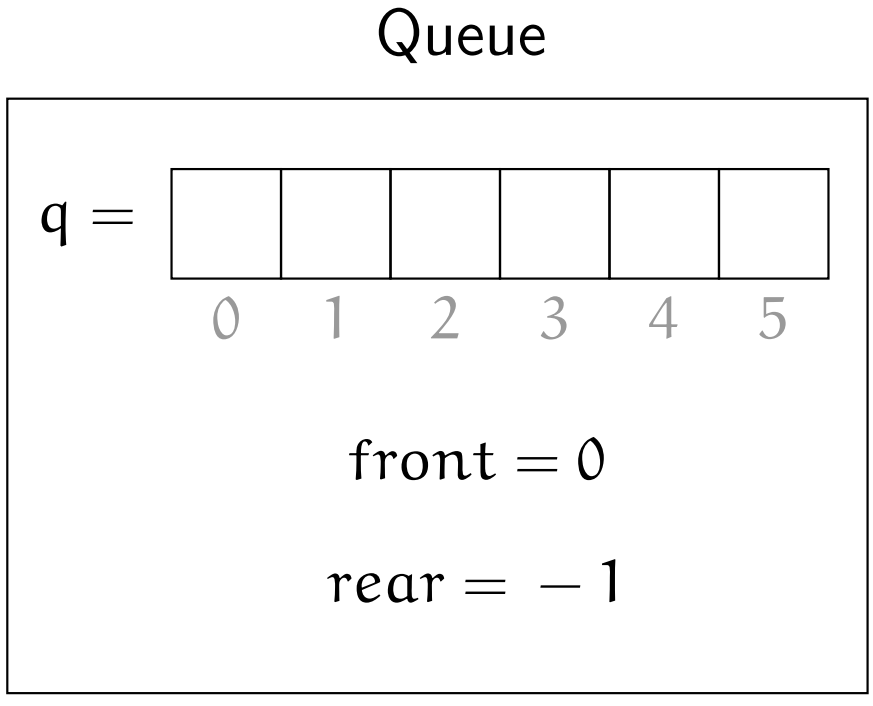
```
rear == n - 1
```

Empty

```
front > rear
```

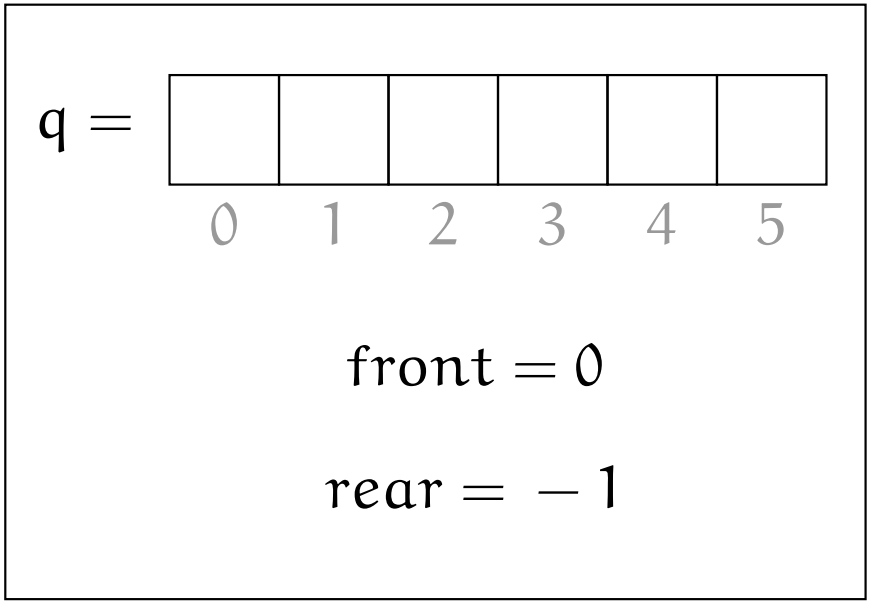
Circular Queues
- Implementation

Circular Queues - Implementation



Circular Queues - Implementation

Queue



Push

```
rear++;
q[rear] = a;
```

Pop

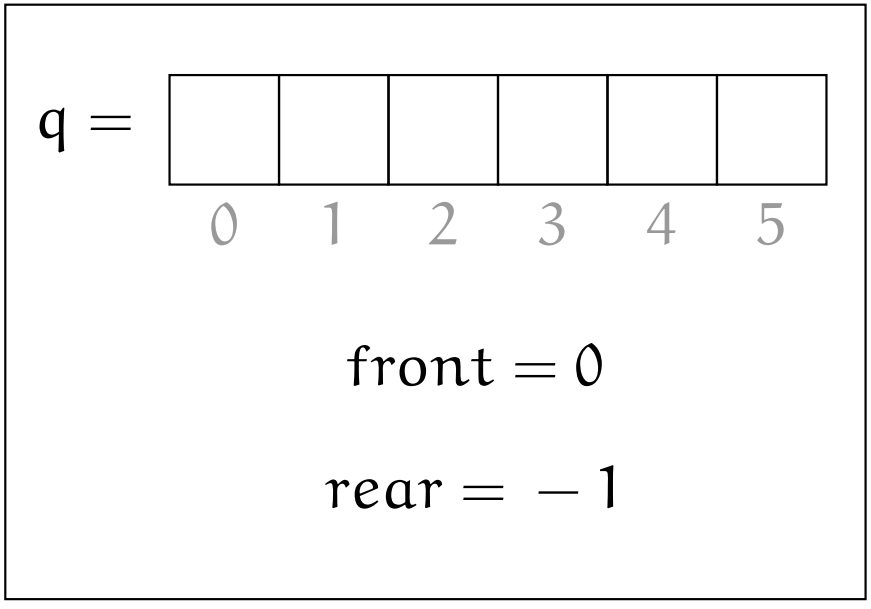
```
t = q[front];
front++;
```

Empty

```
front > rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

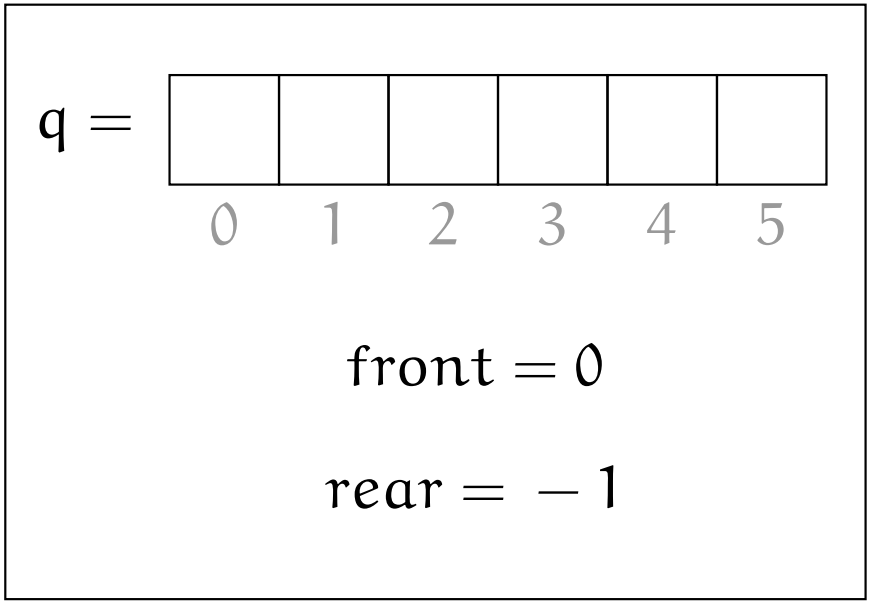
```
t = q[front];
front++;
```

Empty

```
front > rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

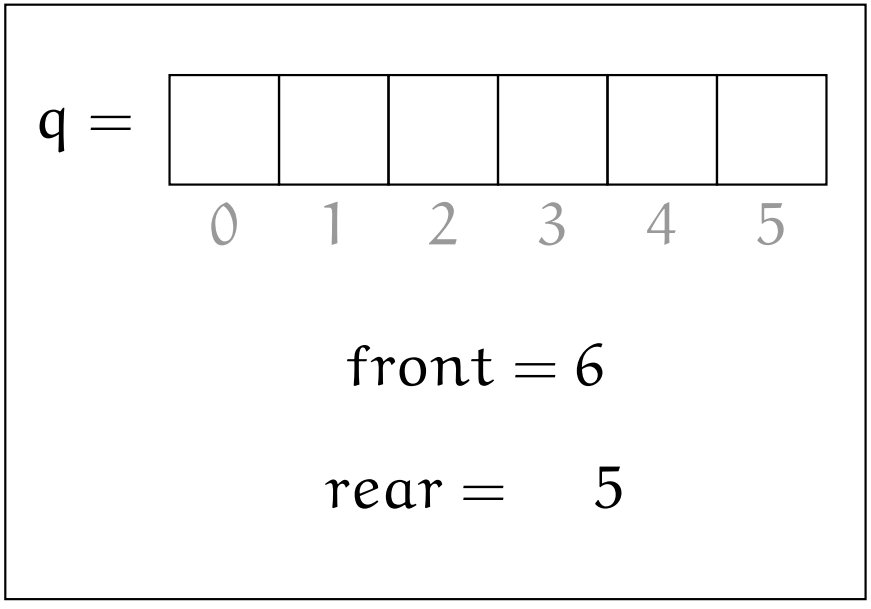
```
t = q[front];
front = (front + 1)%6
```

Empty

```
front > rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

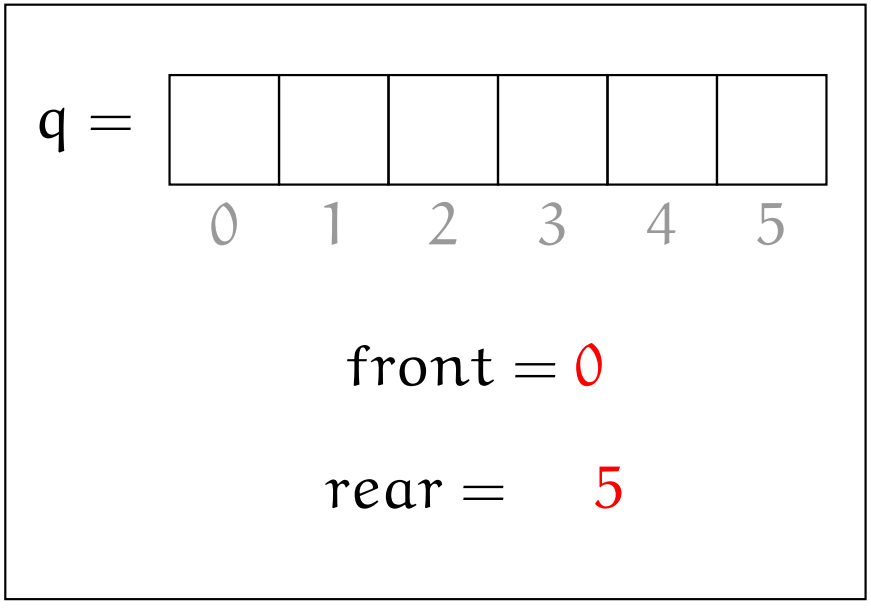
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaE

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

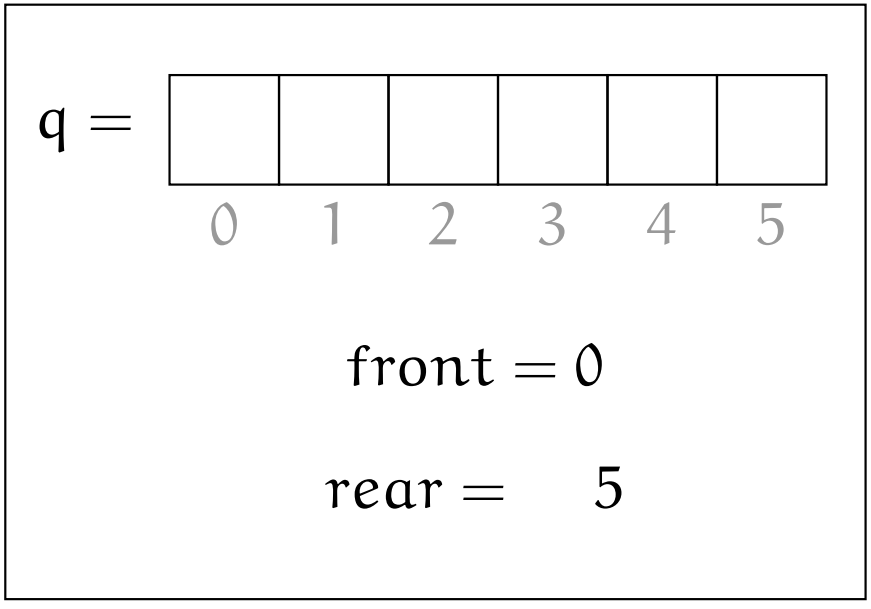
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaE

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

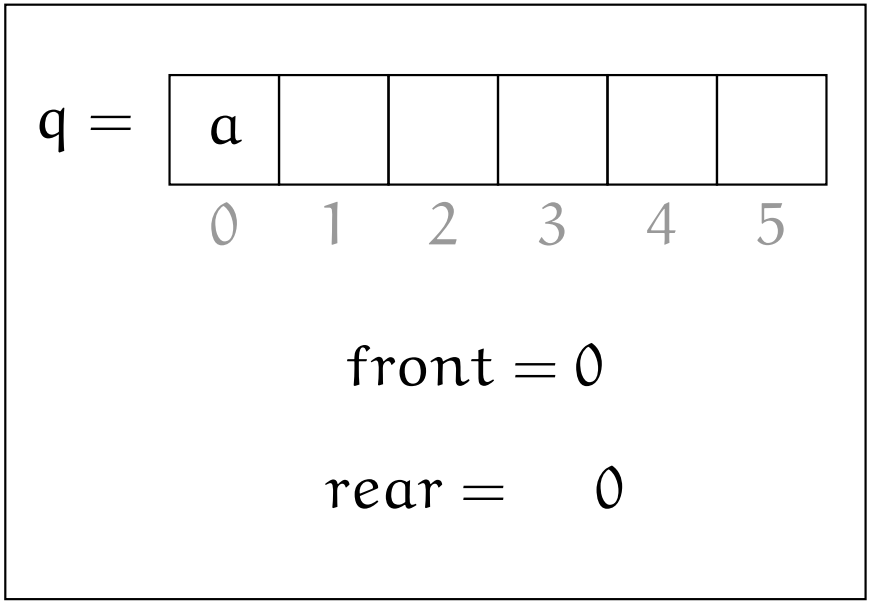
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaE

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

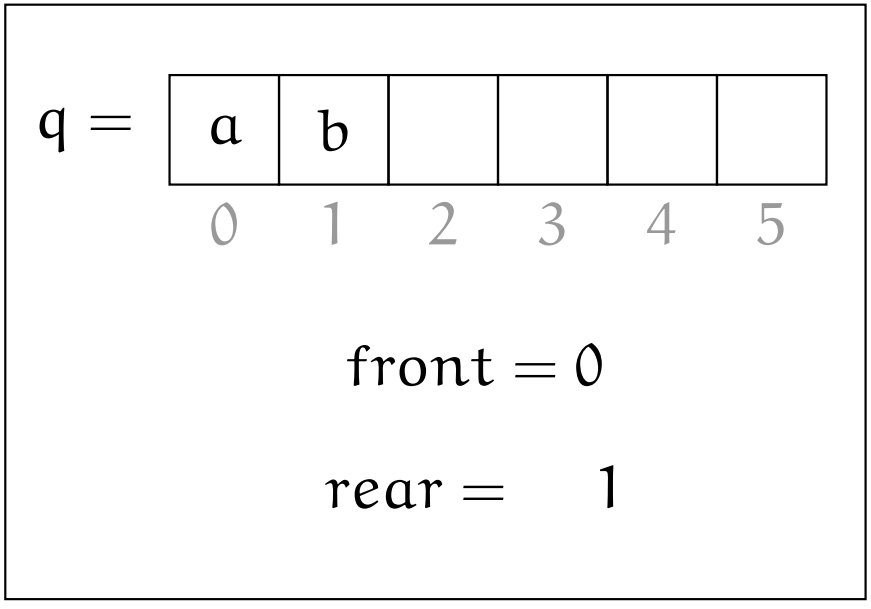
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaE

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

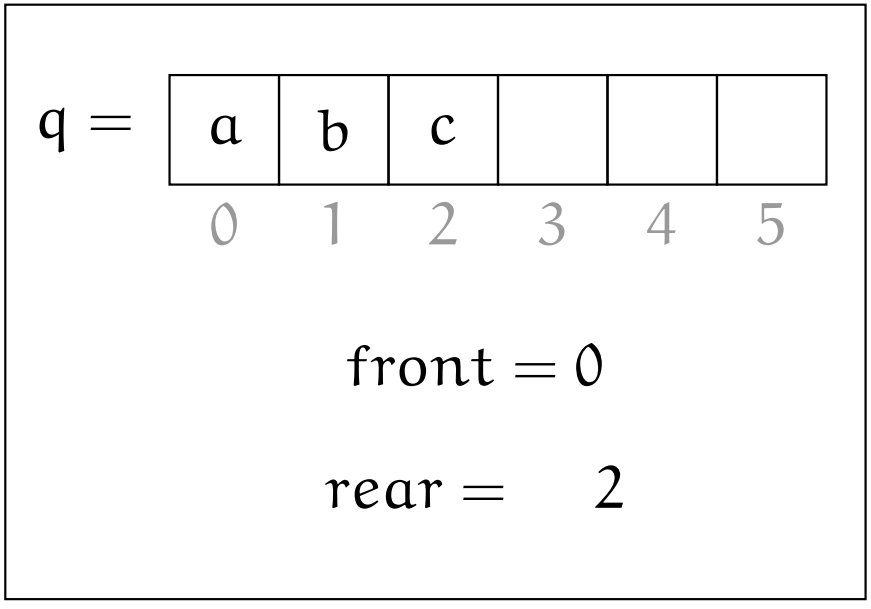
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaE

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Input : abcPPdfaPPPPPabcPabcde

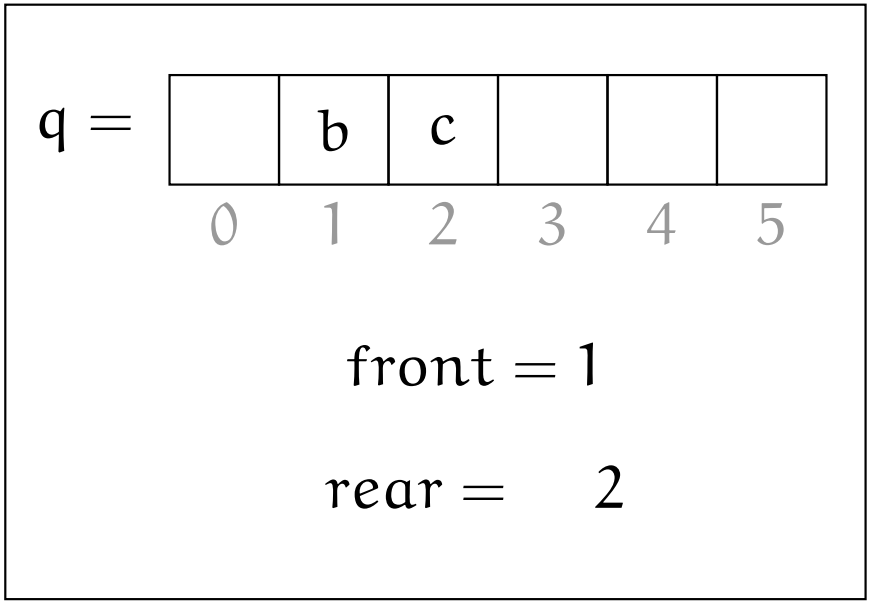
Output : abcdfaE

Empty

```
front > rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

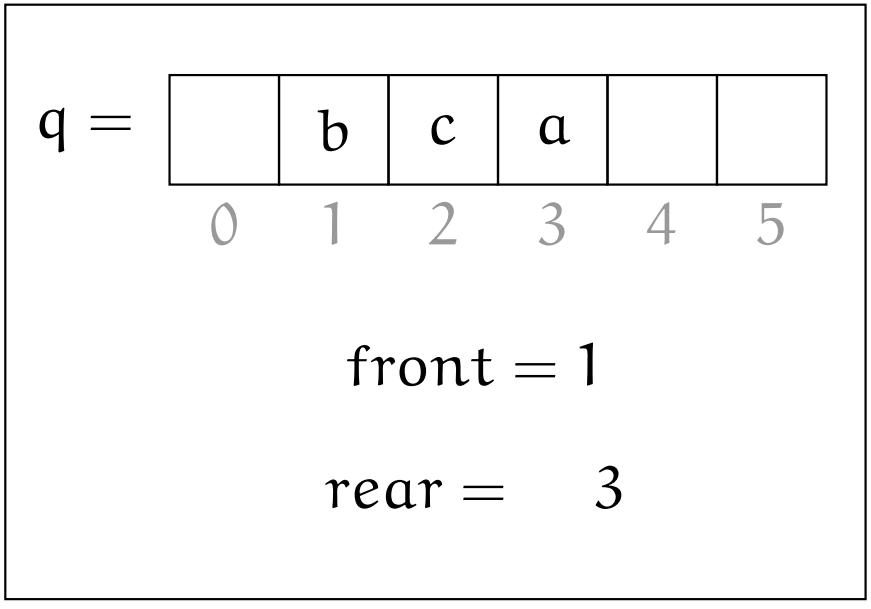
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaEa

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

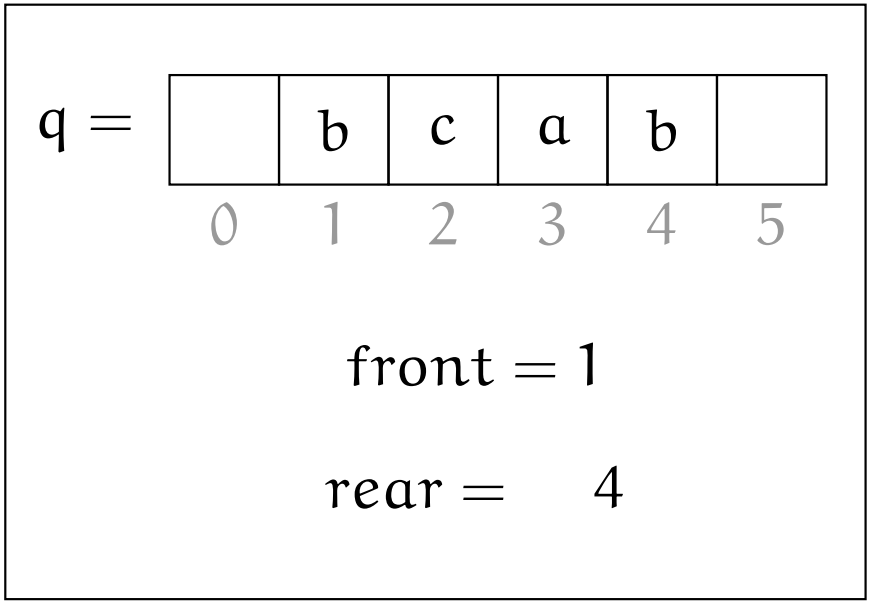
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaEa

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

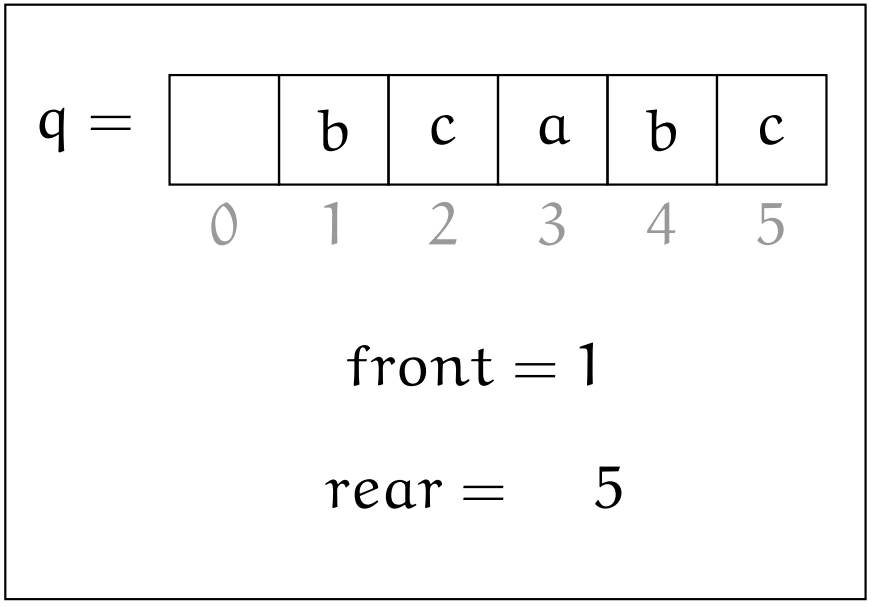
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaEa

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

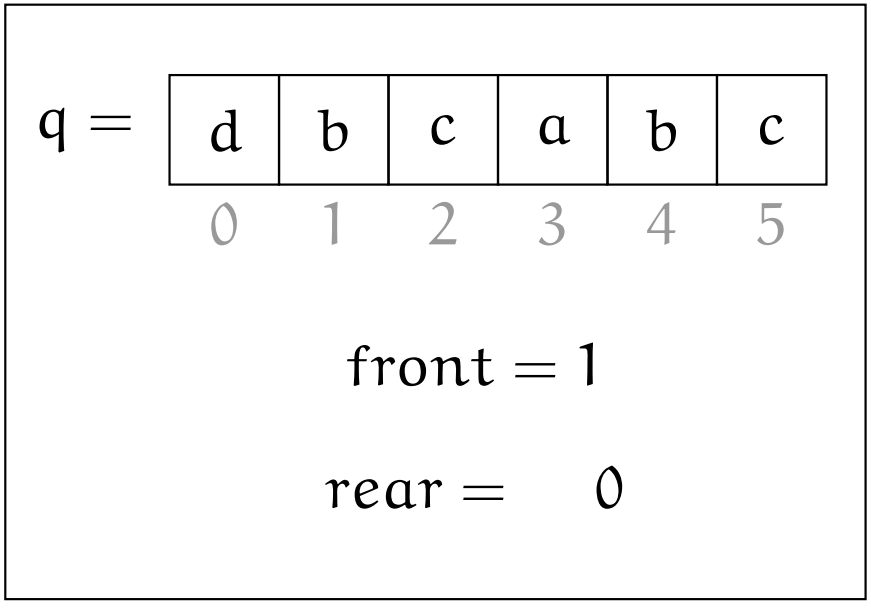
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaEa

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

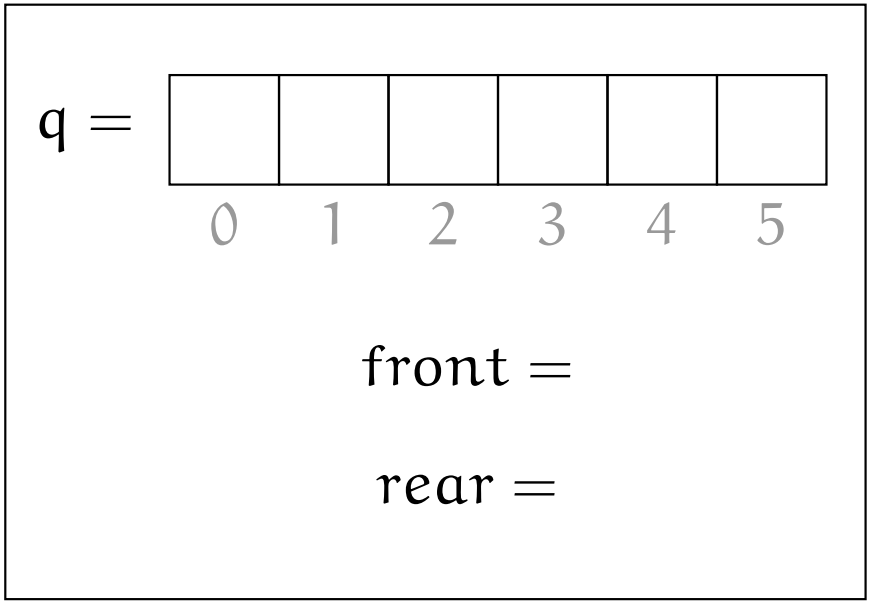
```
front > rear
```

Input : abcPPdfaPPPPPabcPabcde

Output : abcdfaEa

Circular Queues - Implementation

Queue



Push

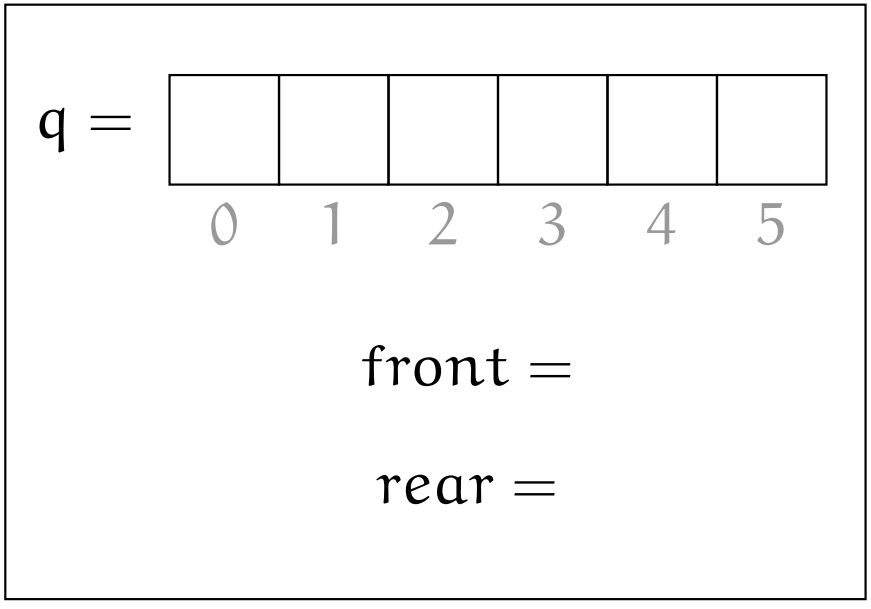
```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

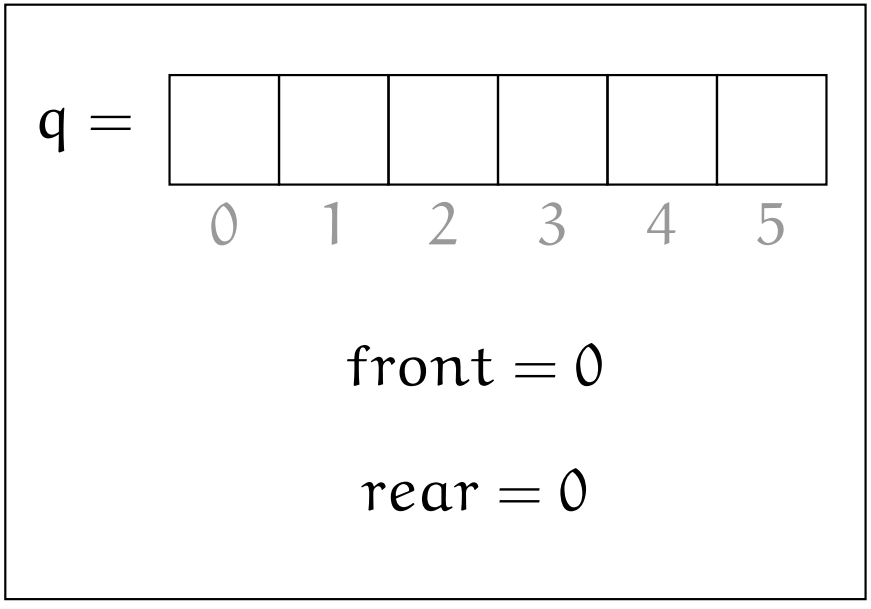
```
t = q[front];
front = (front + 1)%6
```

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

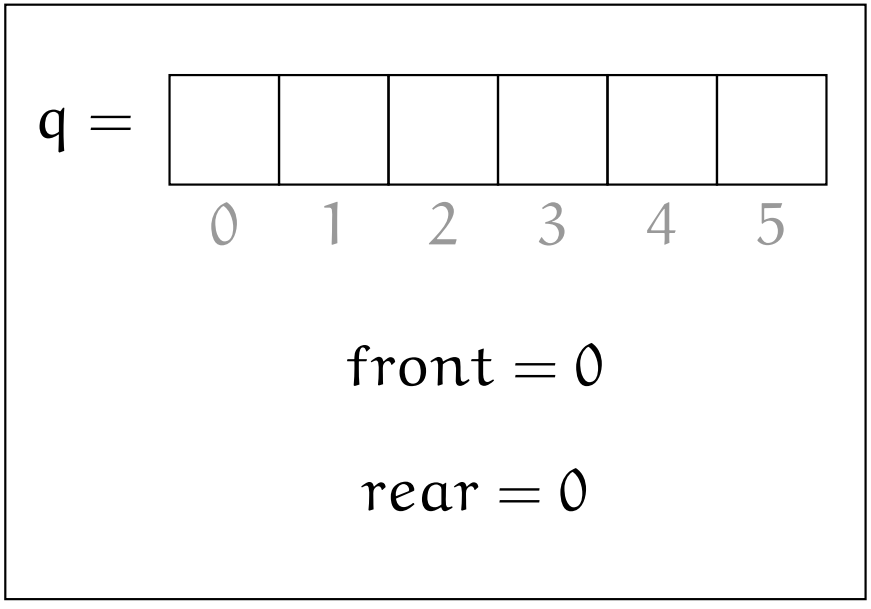
```
t = q[front];
front = (front + 1)%6
```

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

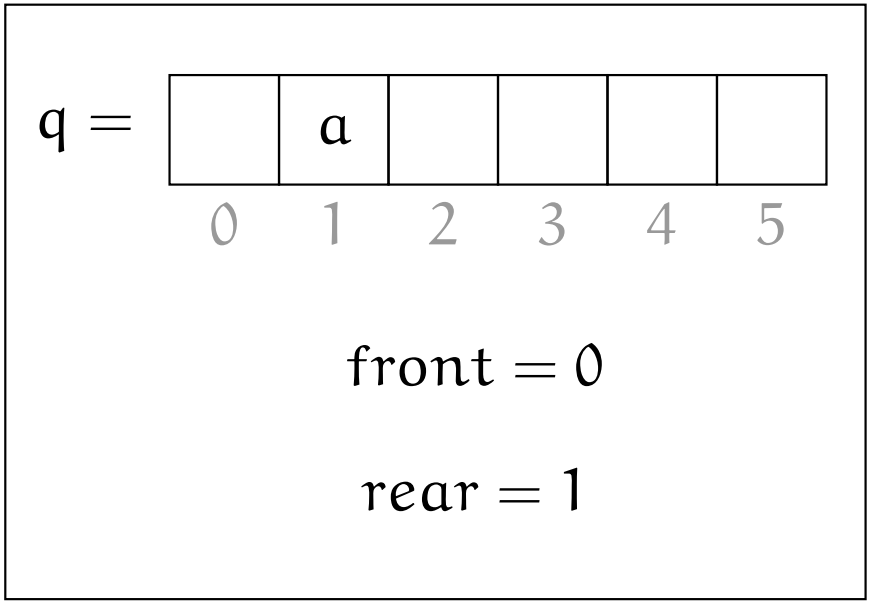
```
front == rear
```

Input : abcPPdfaPPPPP

Output :

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

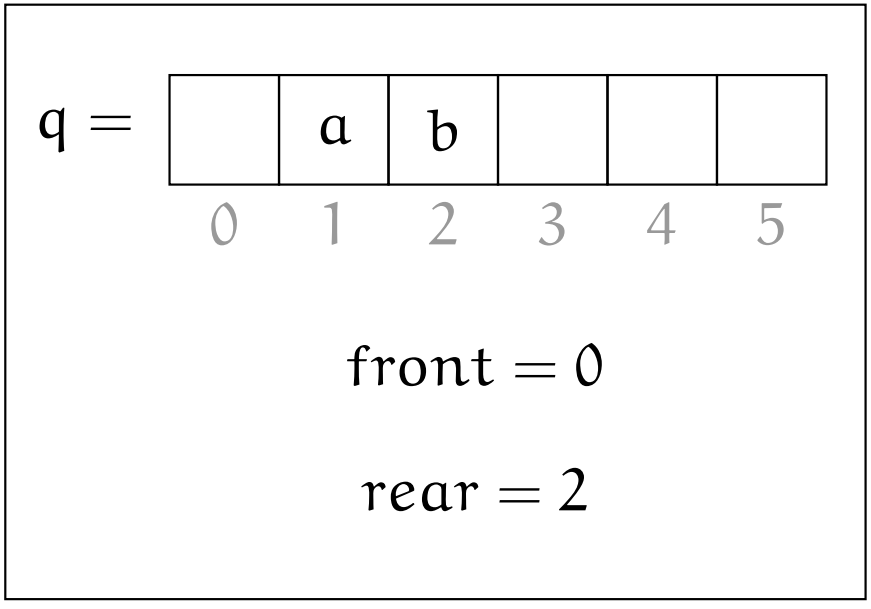
```
front == rear
```

Input : abcPPdfaPPPPP

Output :

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Empty

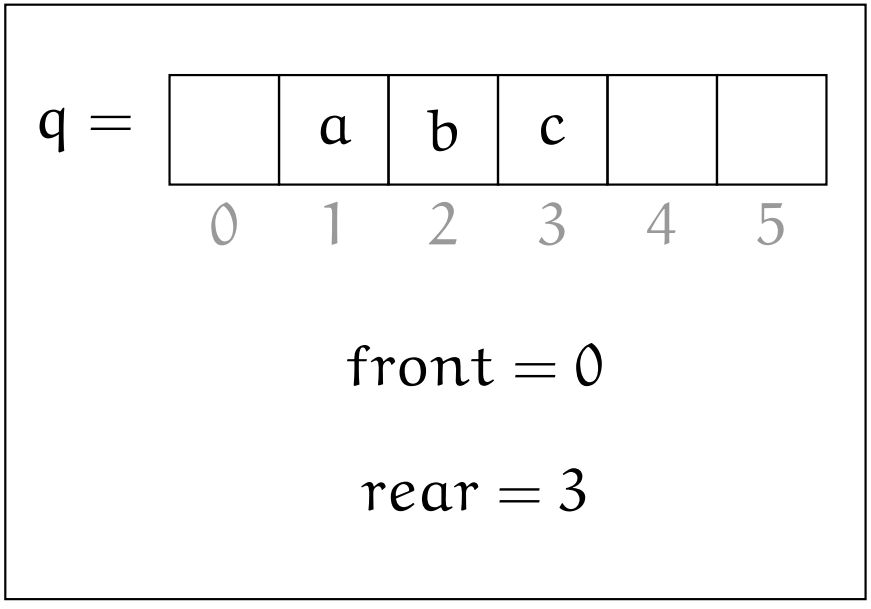
```
front == rear
```

Input : abcPPdfaPPPPP

Output :

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
t = q[front];
front = (front + 1)%6
```

Input : abcPPdfaPPPPP

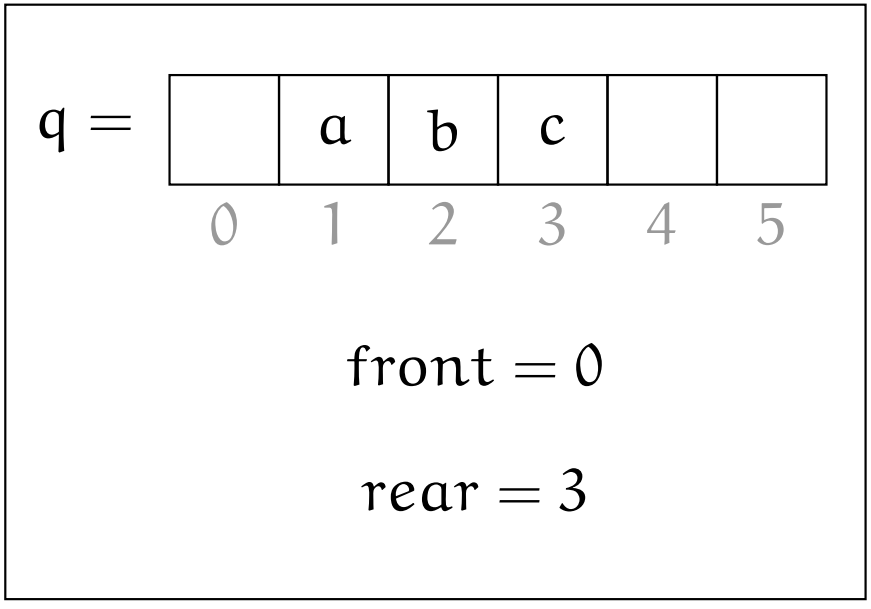
Output :

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

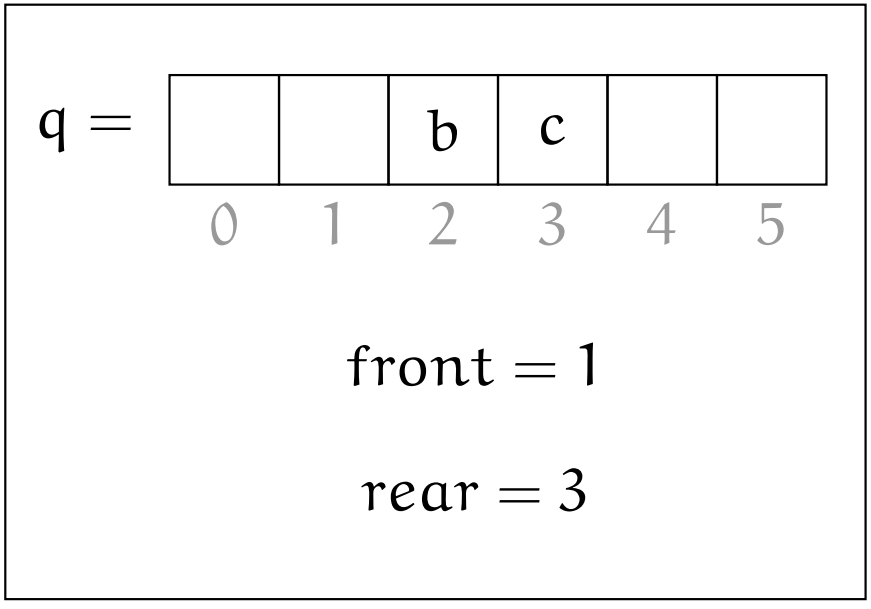
Output :

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

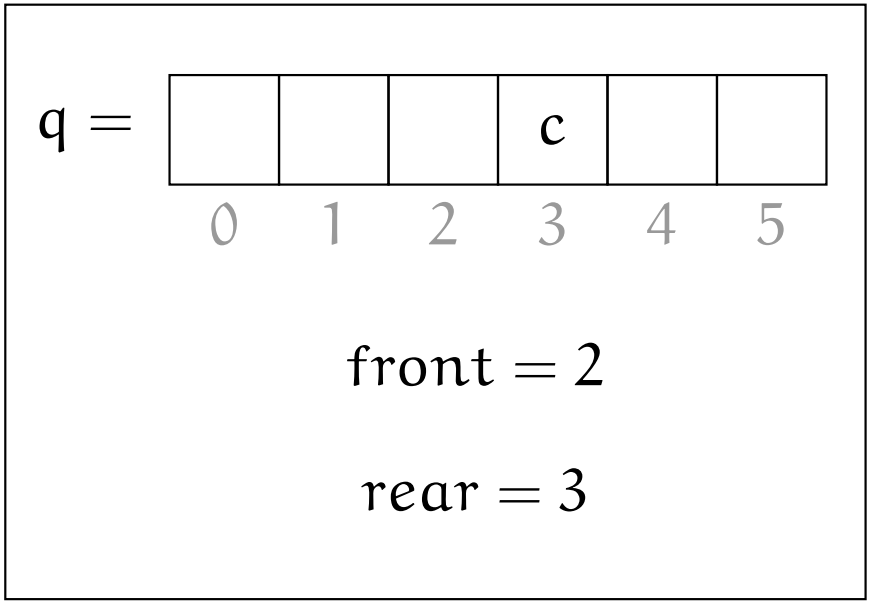
Output : a

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Empty

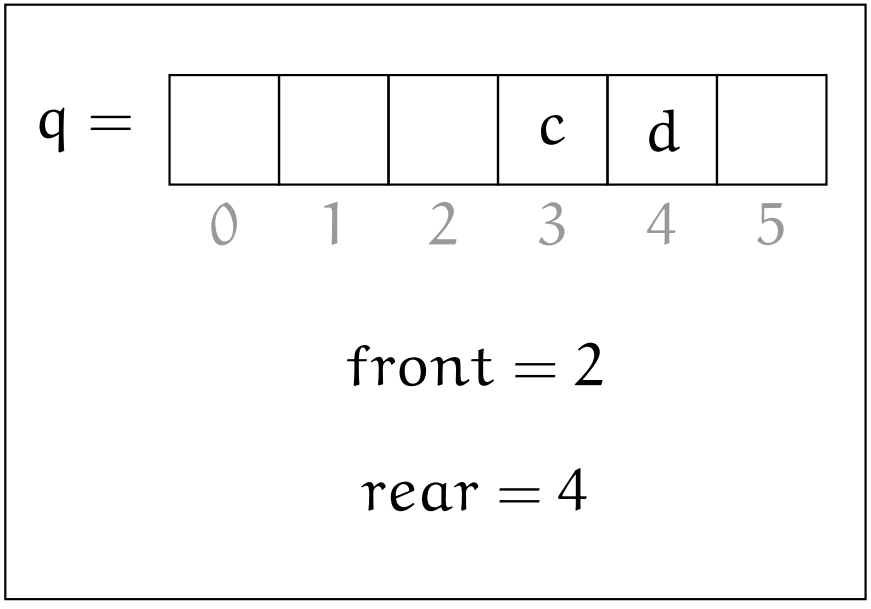
```
front == rear
```

Input : abcPPdfaPPPPP

Output : ab

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

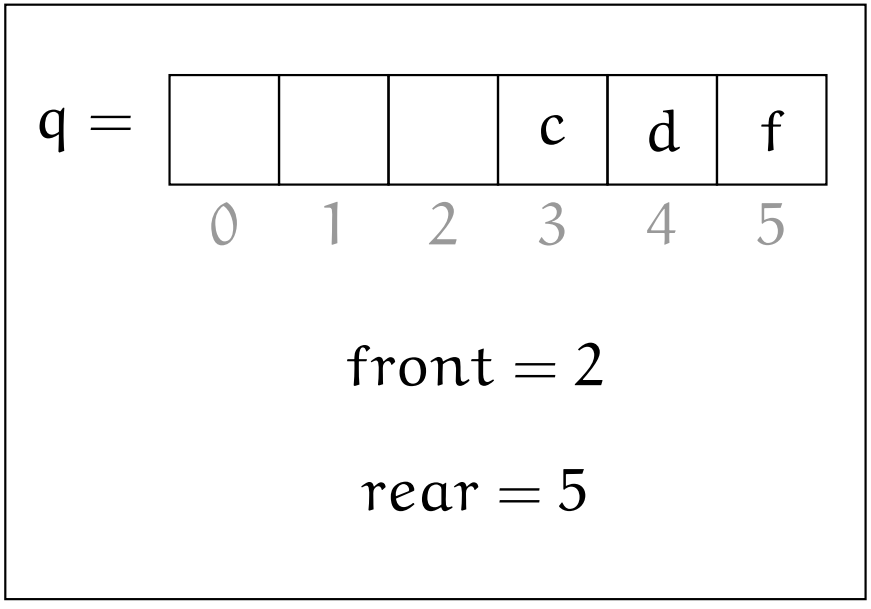
Output : ab

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

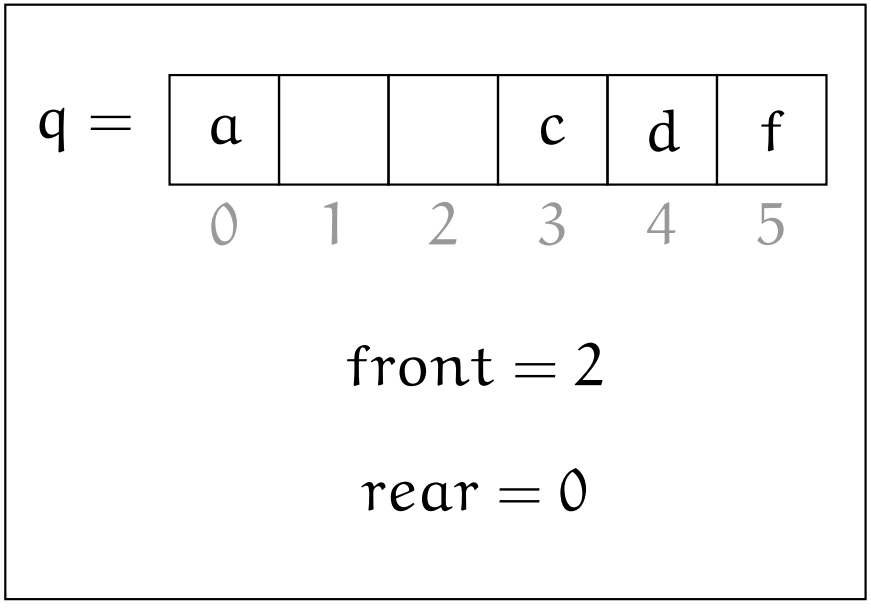
Output : ab

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

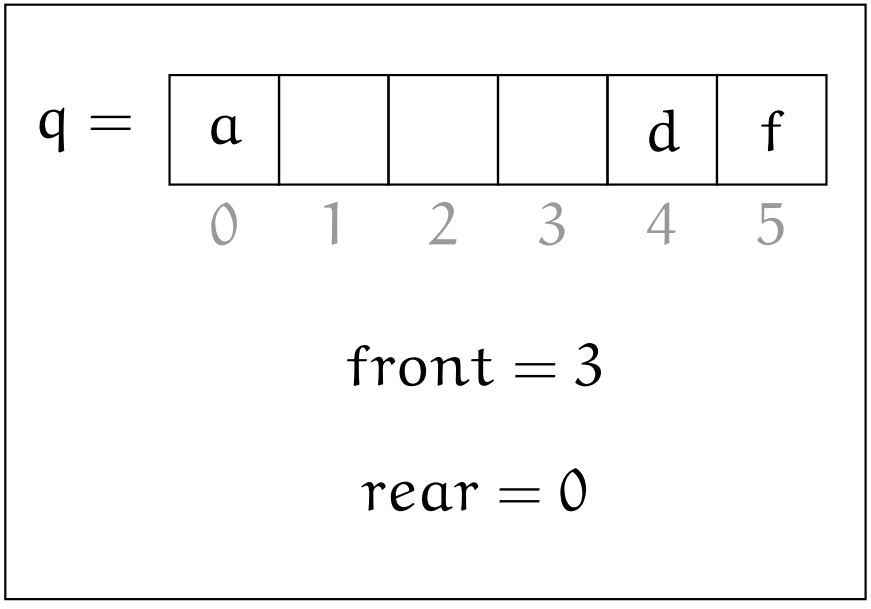
Output : ab

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

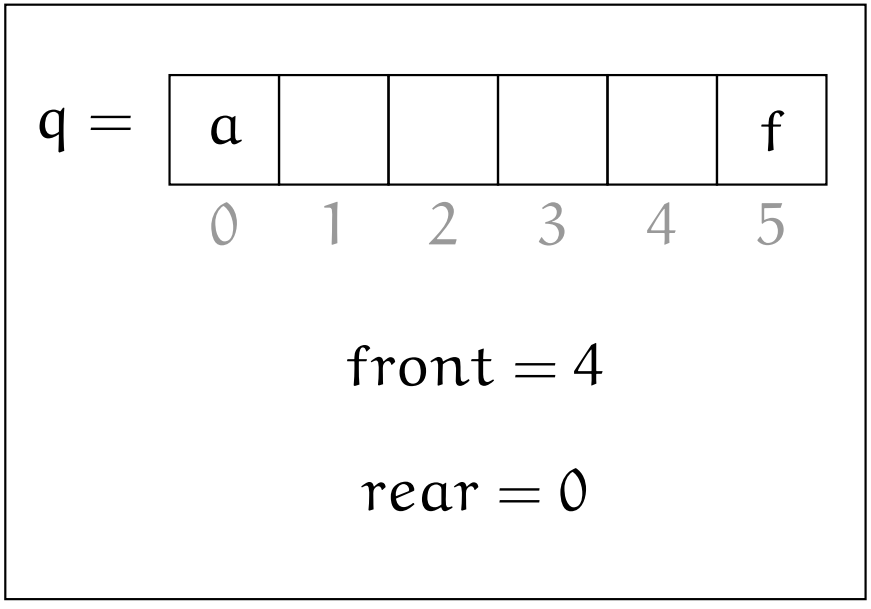
Output : abc

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Empty

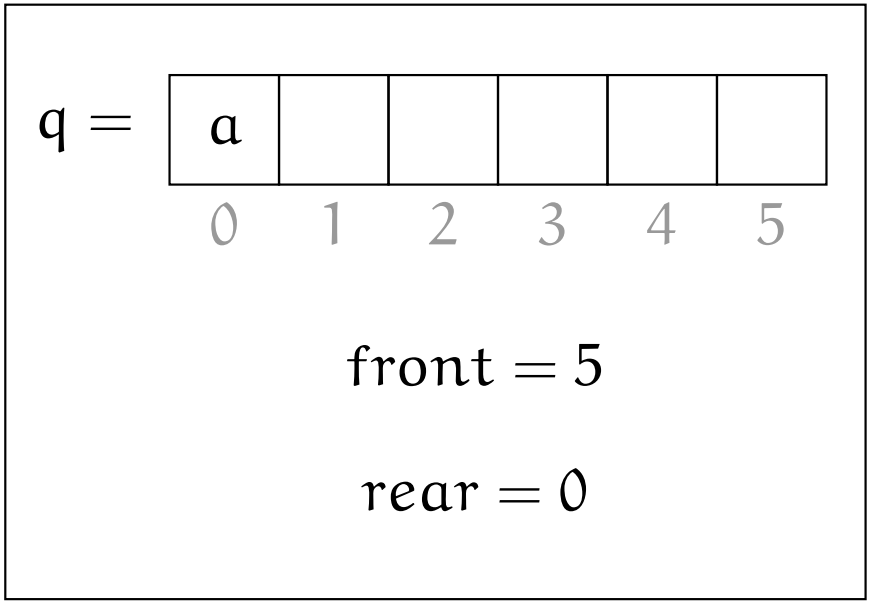
```
front == rear
```

Input : abcPPdfaPPPPP

Output : abcd

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Empty

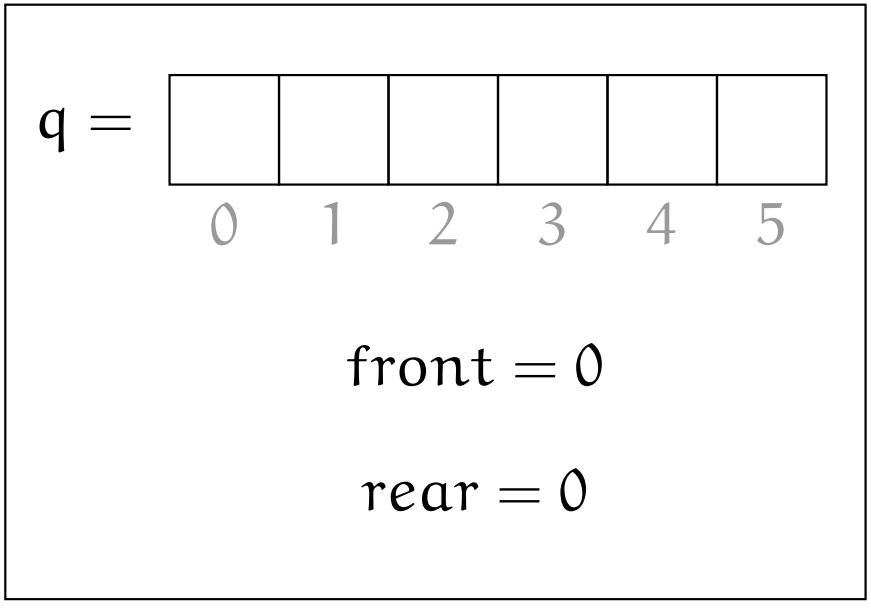
```
front == rear
```

Input : abcPPdfaPPPPP

Output : abcdf

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

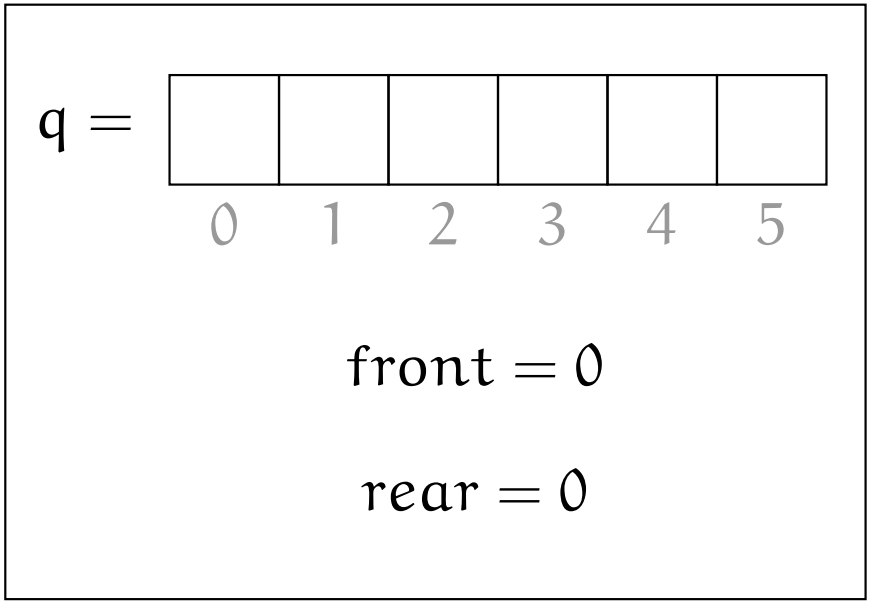
Output : abcdfa

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPP

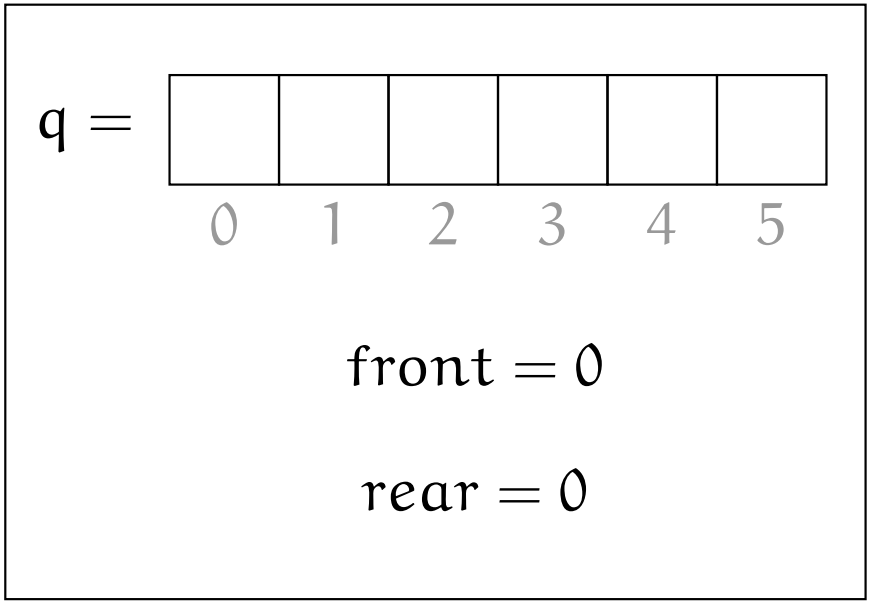
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

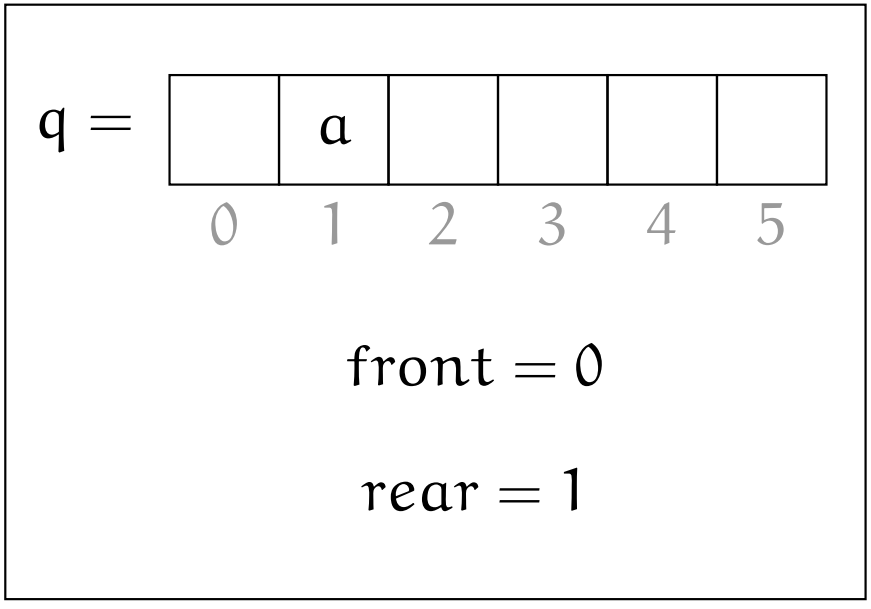
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

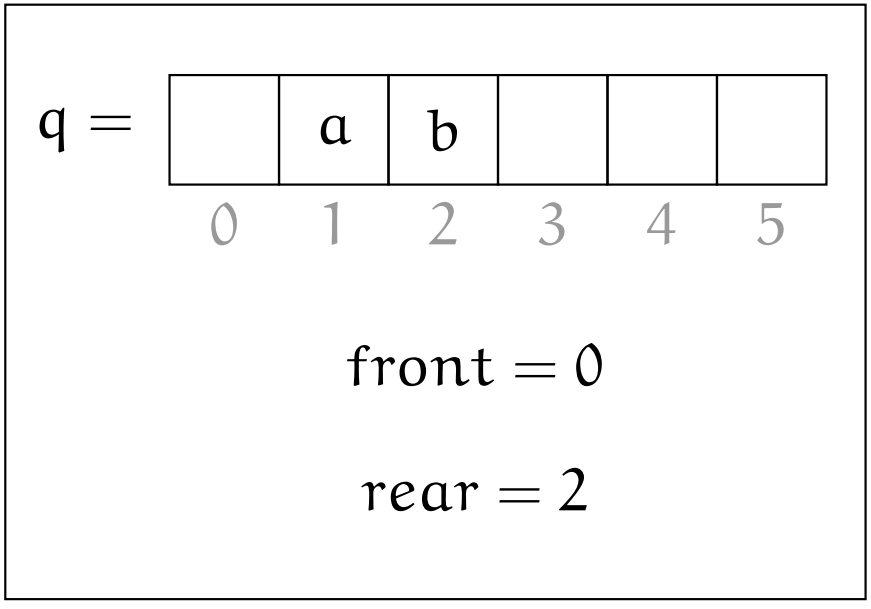
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

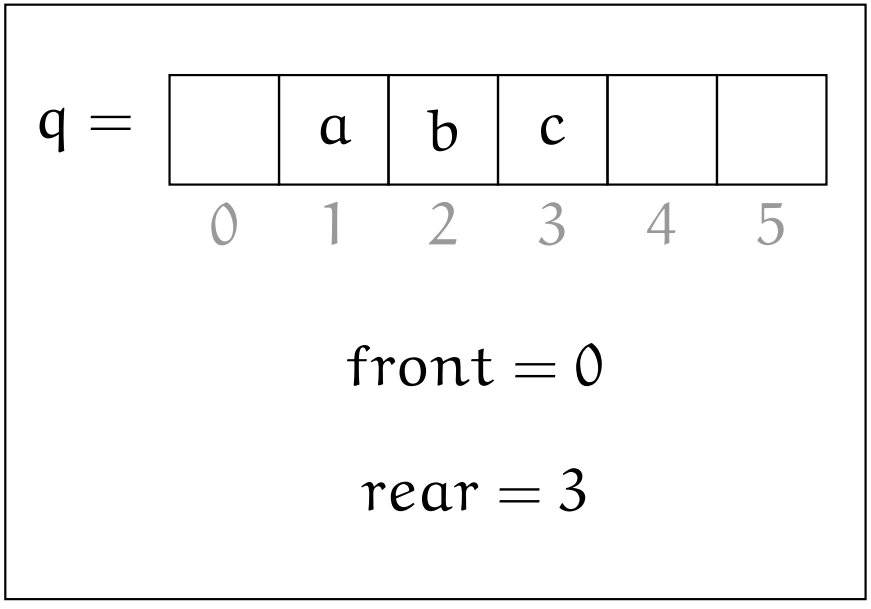
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

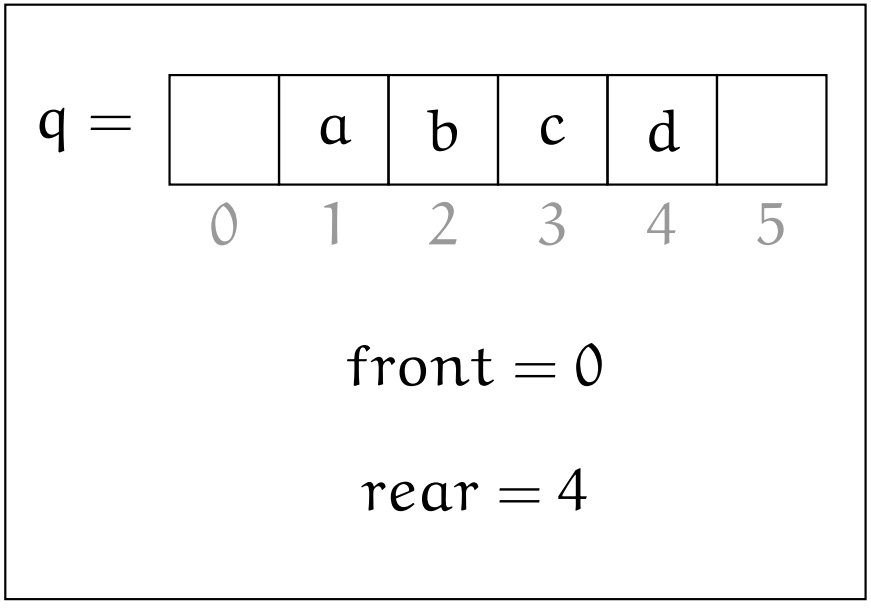
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

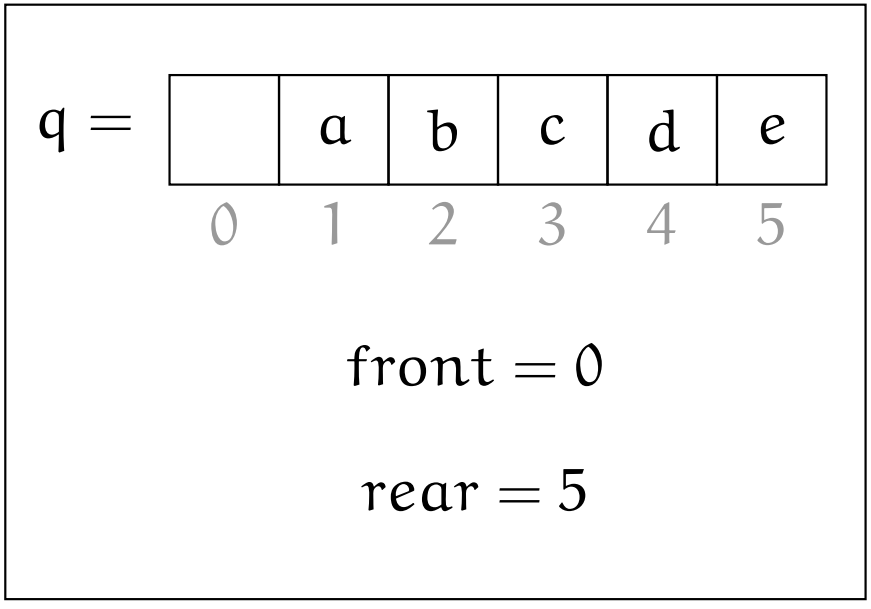
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

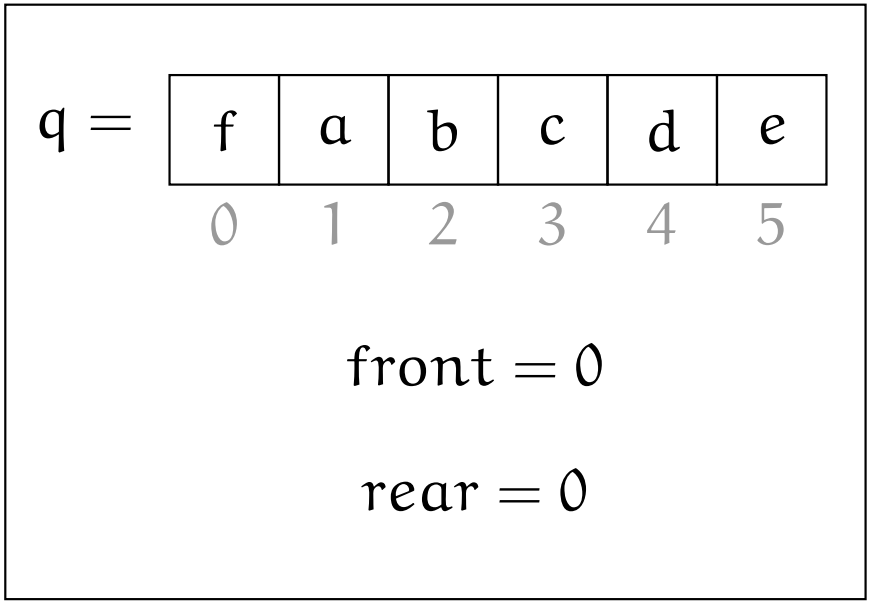
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

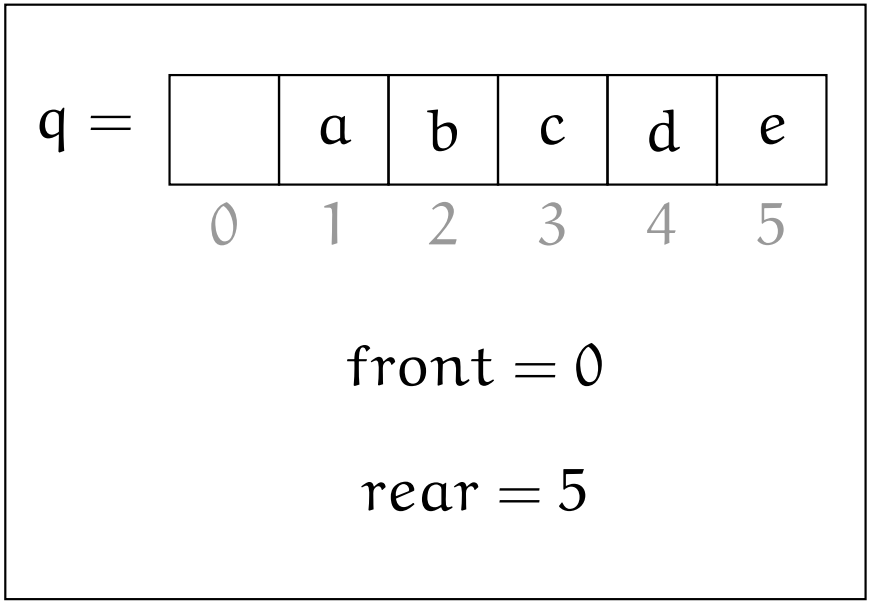
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Input : abcPPdfaPPPPPabcdef

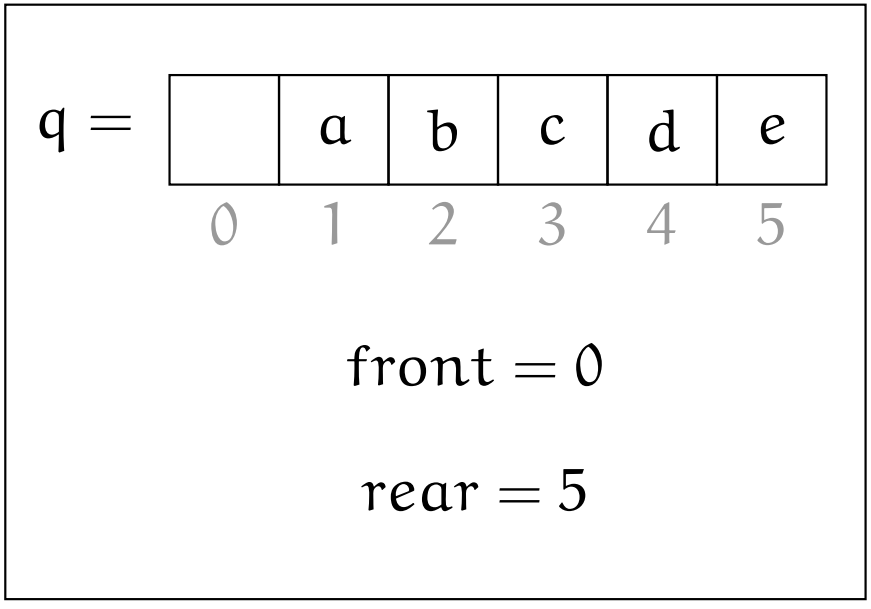
Output : abcdfaE

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Full

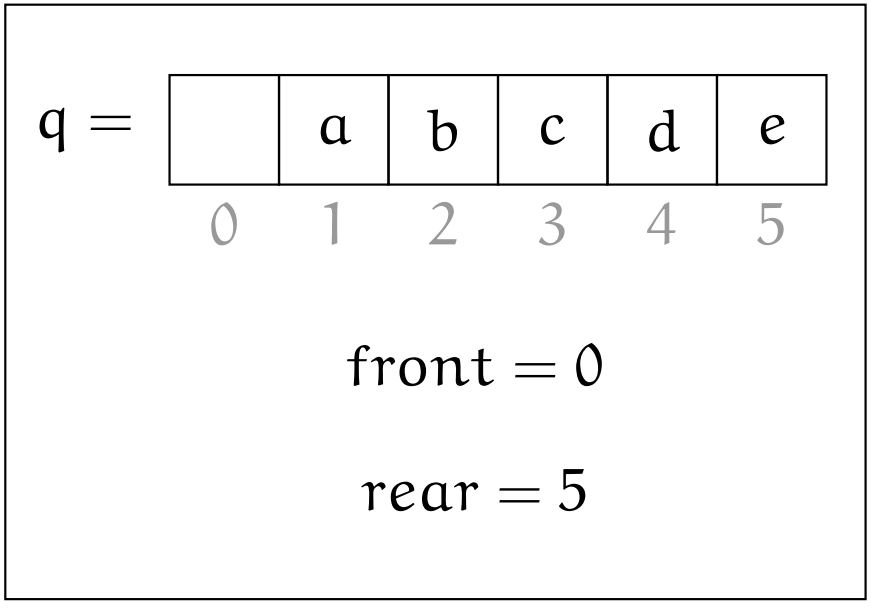
```
(rear + 1)%6 == front
```

Empty

```
front == rear
```

Circular Queues - Implementation

Queue



Push

```
rear = (rear + 1)%6;
q[rear] = a;
```

Pop

```
front = (front + 1)%6
t = q[front];
```

Capacity

$n - 1$

Full

$(rear + 1)\%6 == front$

Empty

$front == rear$