Data Structures

Circular Queue – Array Implementation

Team Emertxe

Circular Queue – Array Implementation

Data Structure –Array Implementation

Circular Queue

```
typedef struct
{
        unsigned int capacity;
        int front,rear,count;
        int *item;
} queue_t;
```

Data Structure –Array Implementation

Circular Queue

```
typedef struct
{
        unsigned int capacity;
        int front,rear,count;
        int *item;
} queue_t;
```

Max Queue Capacity

Data Structure –Array Implementation

Circular Queue

```
typedef struct
{
        unsigned int capacity;
        int front,rear,count;
        int *item;
} queue_t;
```

Max Queue Capacity

Variables

Data Structure –Array Implementation

Circular Queue

```
typedef struct
{
        unsigned int capacity;
        int front,rear,count;
        int *item;
} queue_t;
```

Max Queue Capacity

Variables

Array holding the queue elements

Data Structure –Array Implementation

enqueue(queue,element)

Input Specification:

queue : Pointer that contains address of structure variable (queue_t)

element : Item to be added

Output Specification:

Status : e_true / e_false

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue ⟶ capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue       count)
return e_true
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue → front  = -1)
    queue → front = 0
queue  → rear = (queue → rear + 1 ) % (queue → capacity )
queue  → items[queue → rear ] = element
++(queue       count)
return e_true
```

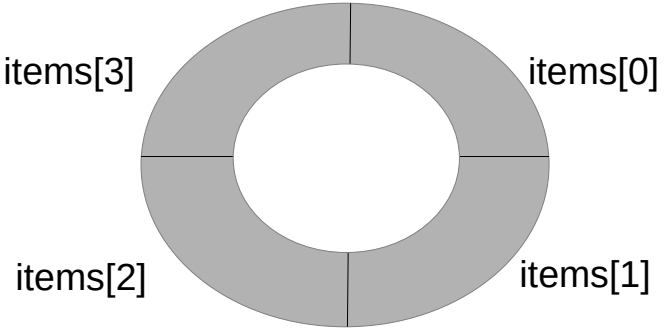
capacity = 4

count = 0

front = -1

rear = -1

element = 10



Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

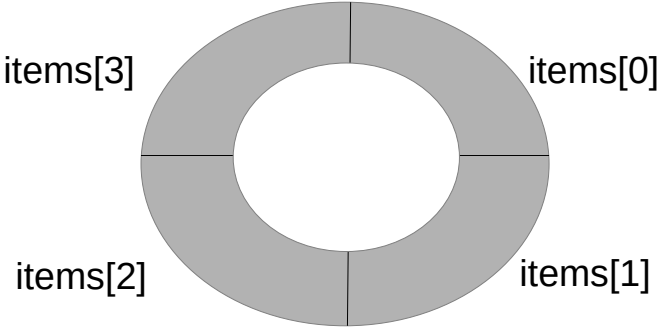
capacity = 4

count = 0

front = -1

rear = -1

element = 10



Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

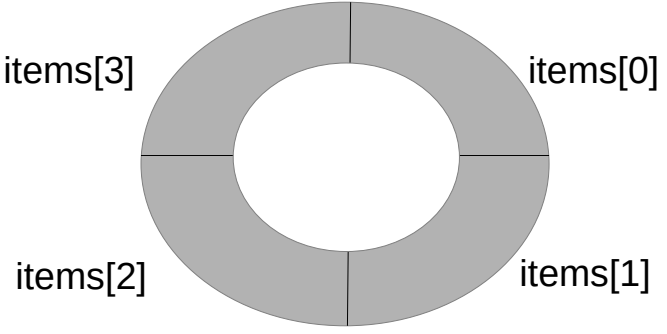
capacity = 4

count = 0

front = -1

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

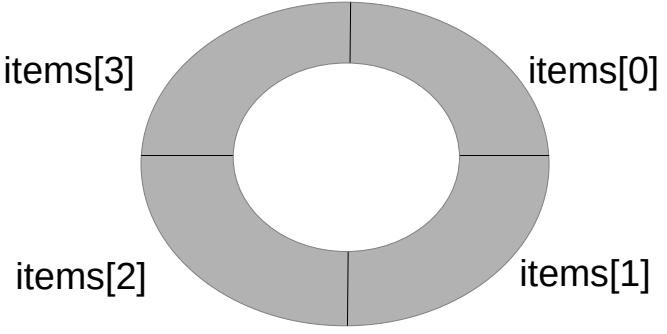
capacity = 4

count = 0

front = -1

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
 else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

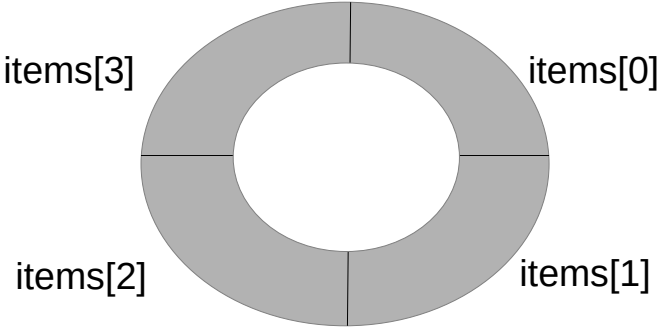
capacity = 4

count = 0

front = -1

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

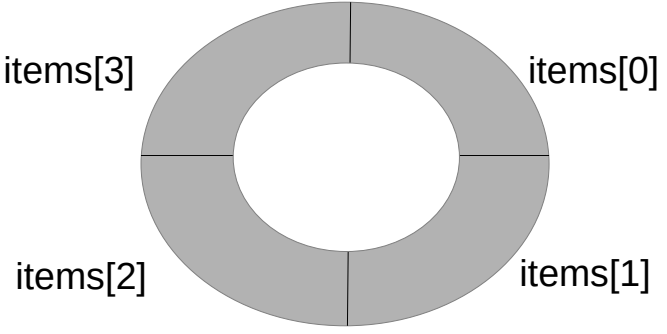
capacity = 4

count = 0

front = -1

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e  false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

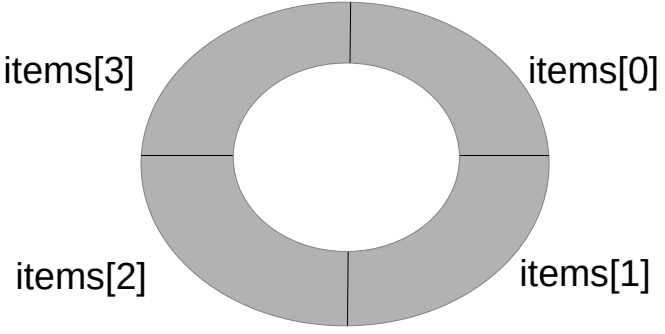
capacity = 4

count = 0

front = -1

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

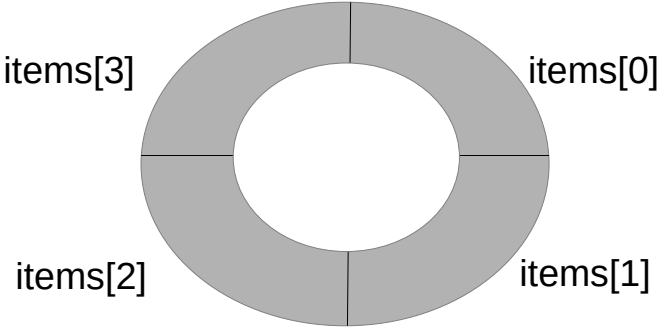
capacity = 4

count = 0

front = -1

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

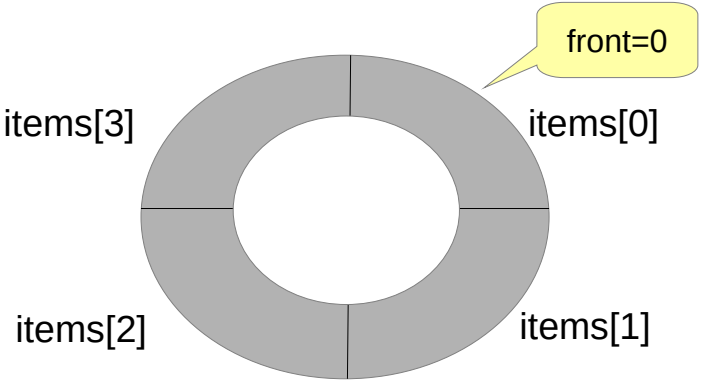
capacity = 4

count = 0

front = 0

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

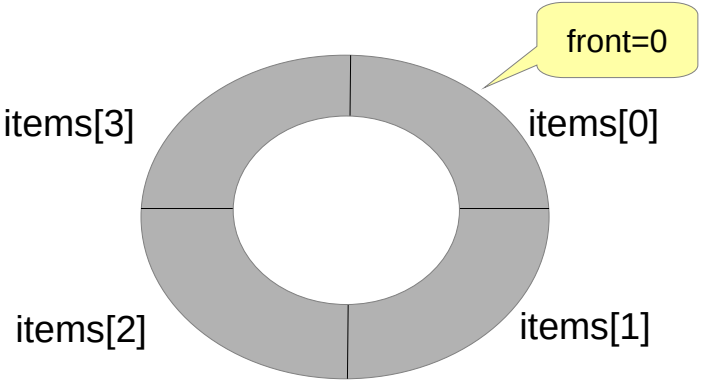
capacity = 4

count = 0

front = 0

rear = -1

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

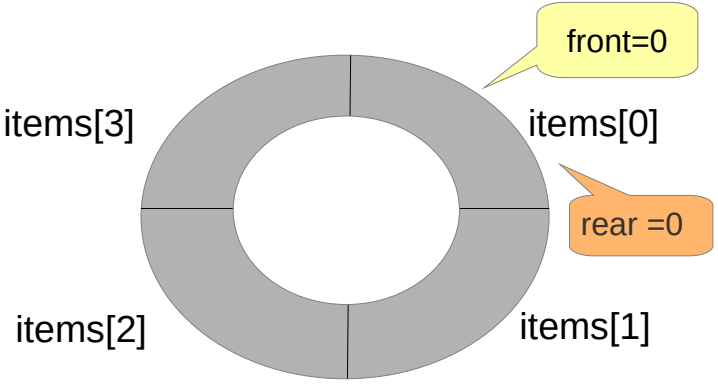
capacity = 4

count = 0

front = 0

rear = 0

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

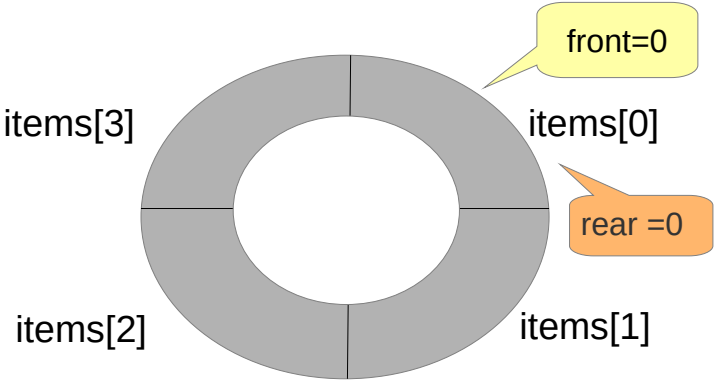
capacity = 4

count = 0

front = 0

rear = 0

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

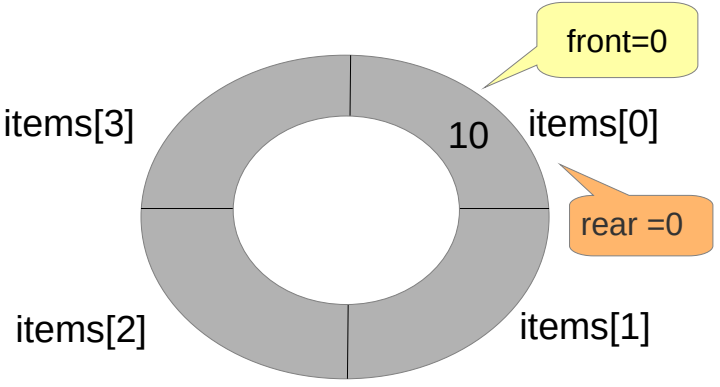
capacity = 4

count = 0

front = 0

rear = 0

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

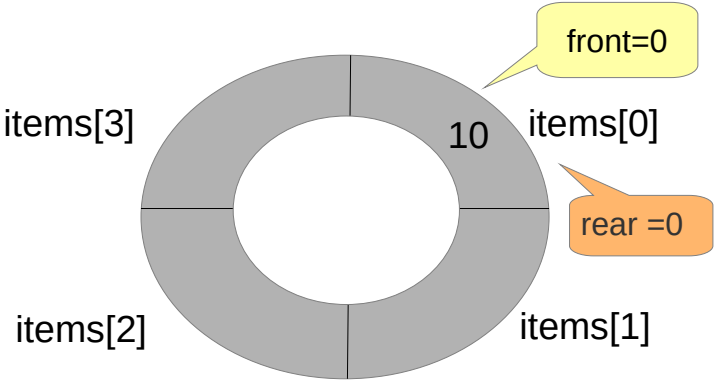
capacity = 4

count = 0

front = 0

rear = 0

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

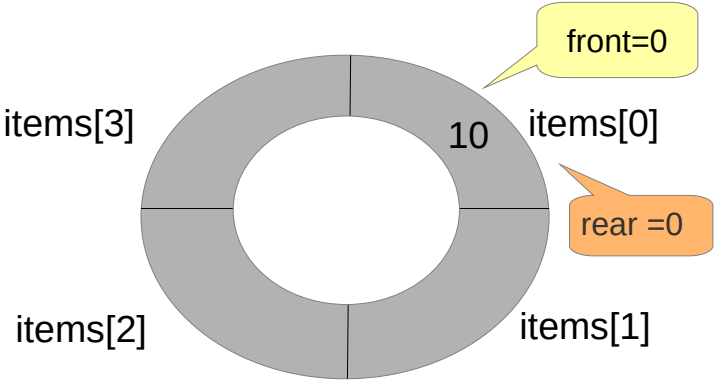
capacity = 4

count = 1

front = 0

rear = 0

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

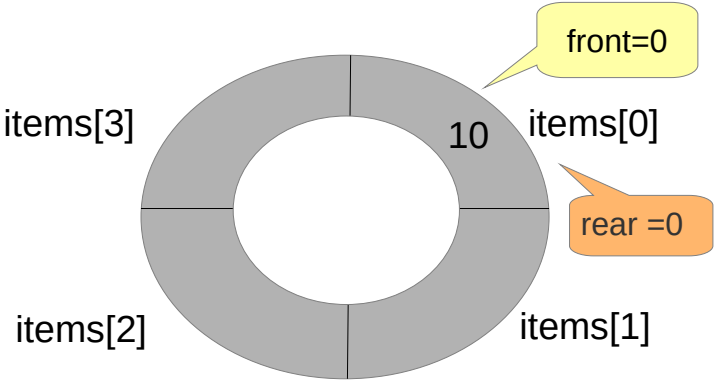
capacity = 4

count = 1

front = 0

rear = 0

element = 10



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

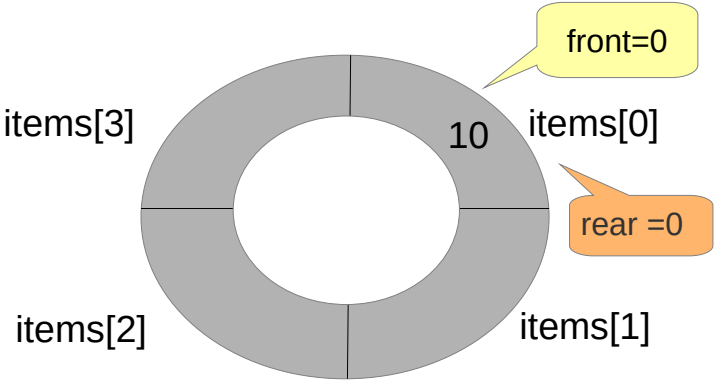
capacity = 4

count = 1

front = 0

rear = 0

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

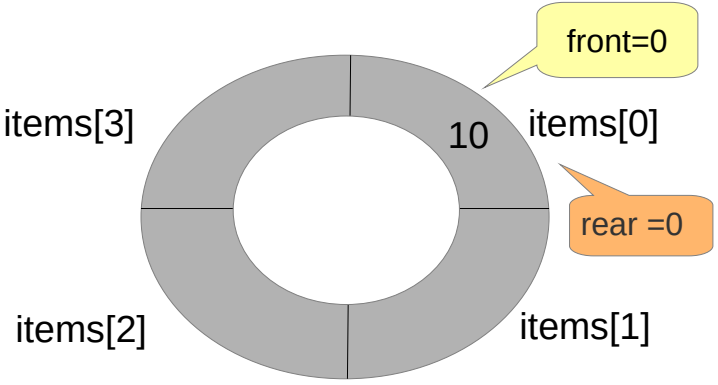
capacity = 4

count = 1

front = 0

rear = 0

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

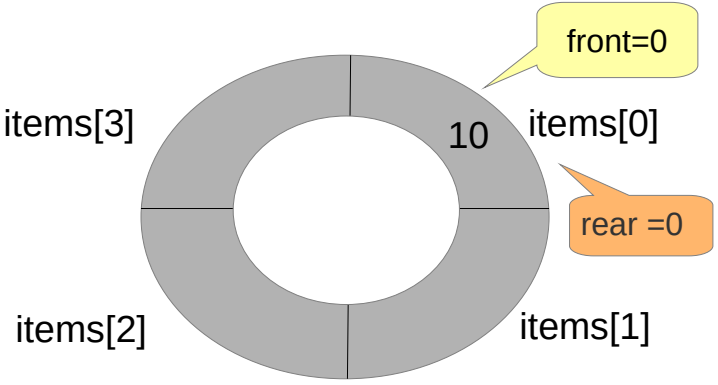
capacity = 4

count = 1

front = 0

rear = 0

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

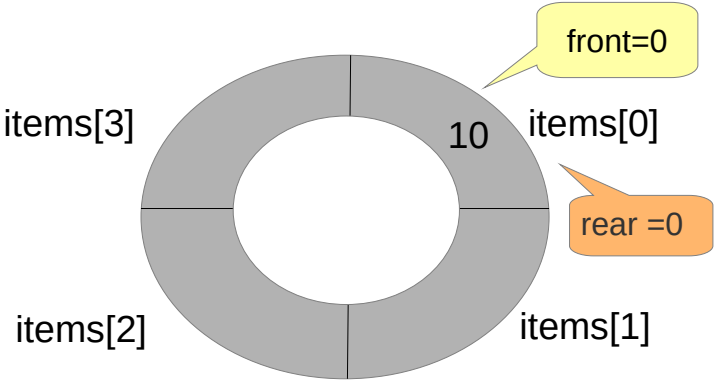
capacity = 4

count = 1

front = 0

rear = 0

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

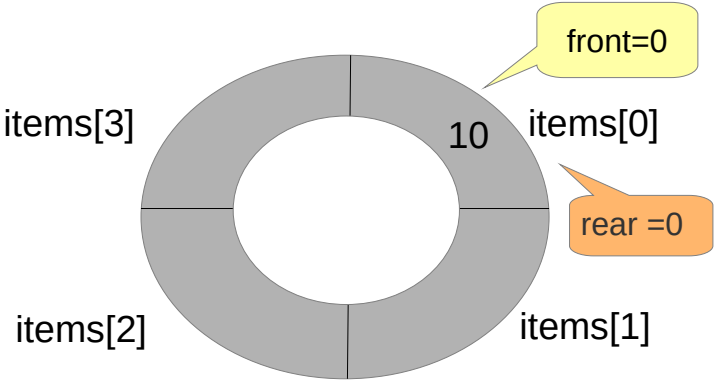
capacity = 4

count = 1

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

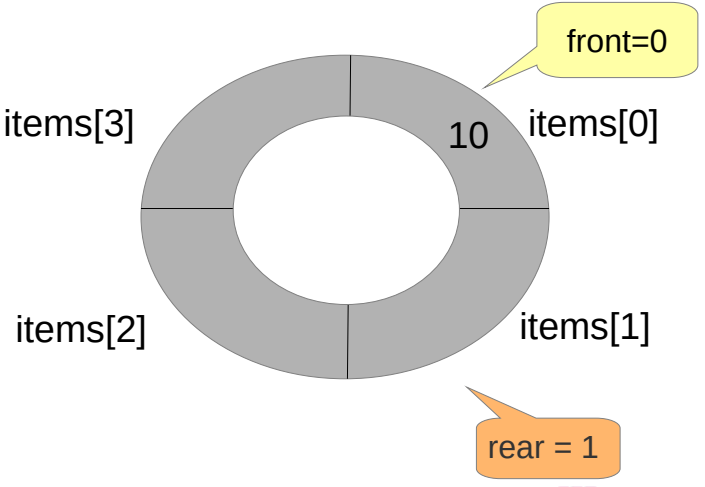
capacity = 4

count = 1

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

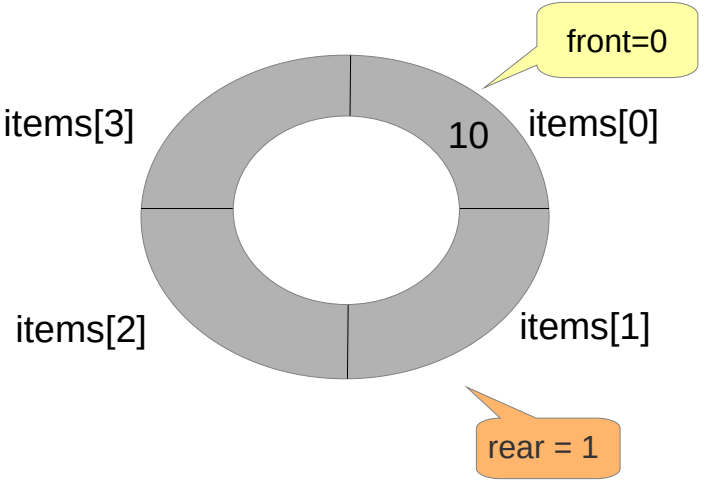
capacity = 4

count = 1

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

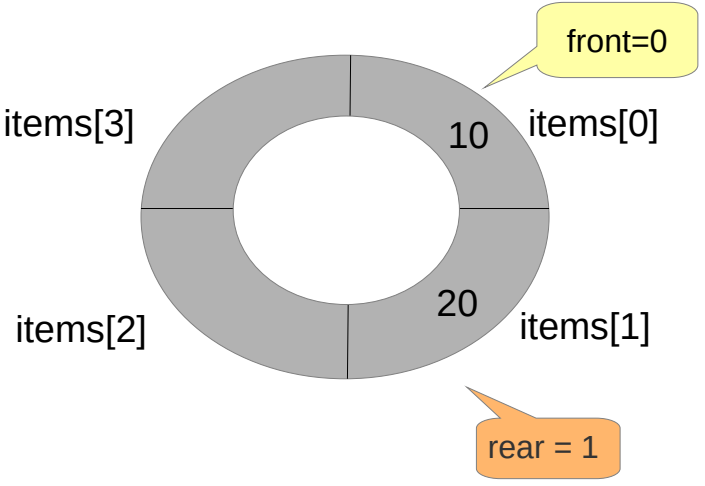
capacity = 4

count = 1

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

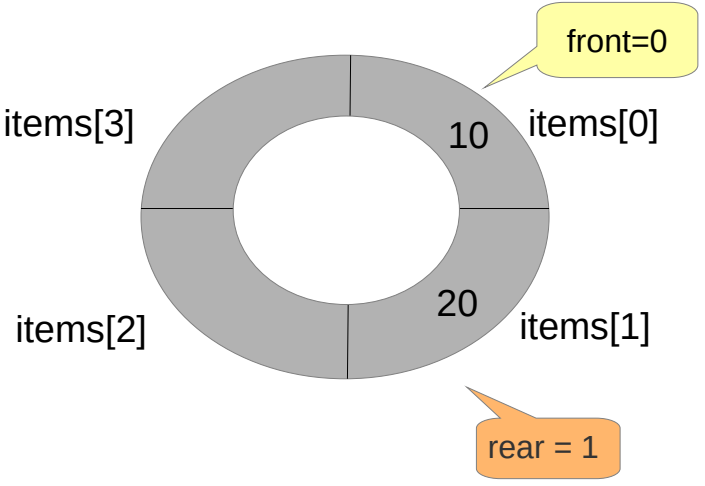
capacity = 4

count = 1

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

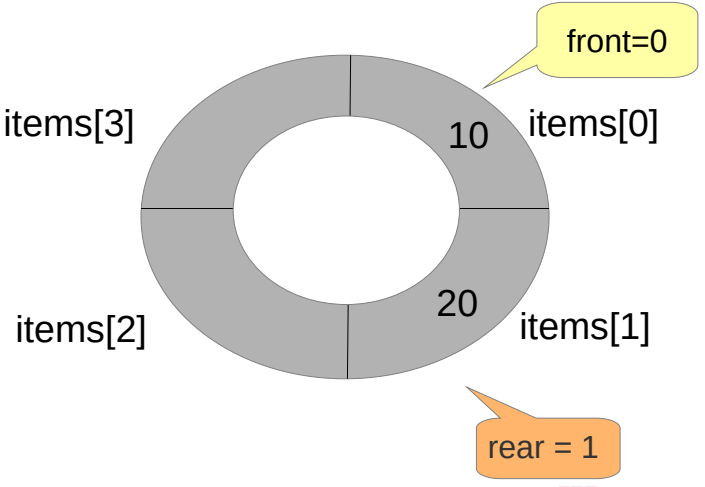
capacity = 4

count = 2

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

ƩMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

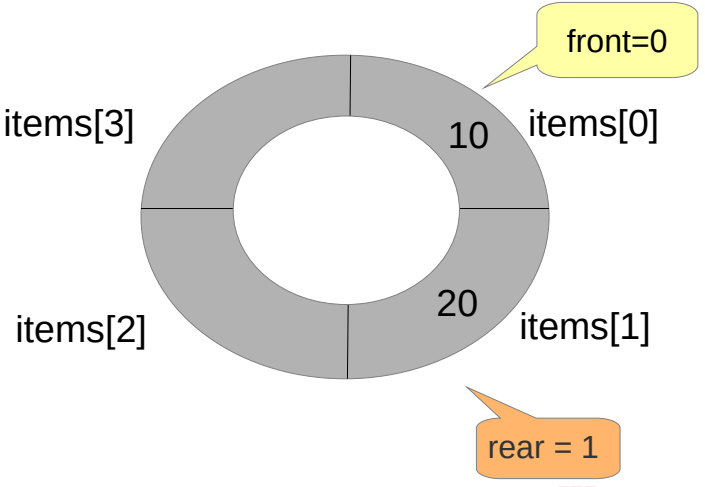
capacity = 4

count = 2

front = 0

rear = 1

element = 20



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

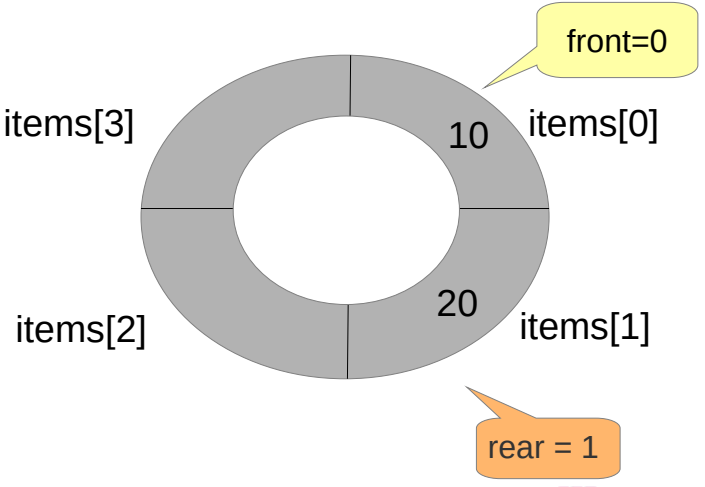
capacity = 4

count = 2

front = 0

rear = 1

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

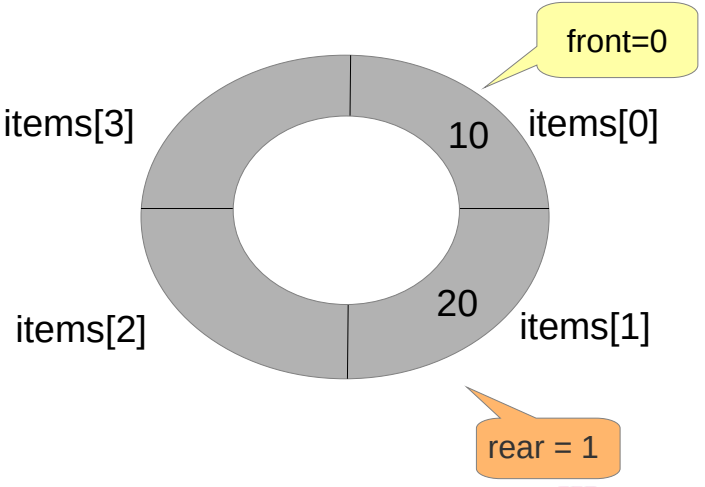
capacity = 4

count = 2

front = 0

rear = 1

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

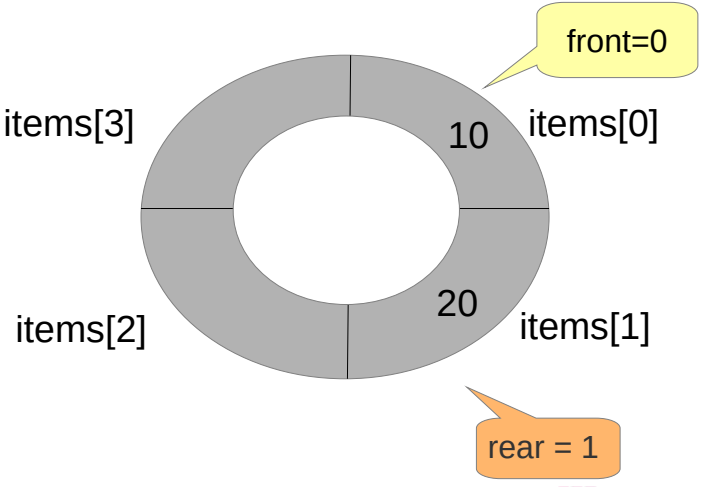
capacity = 4

count = 2

front = 0

rear = 1

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

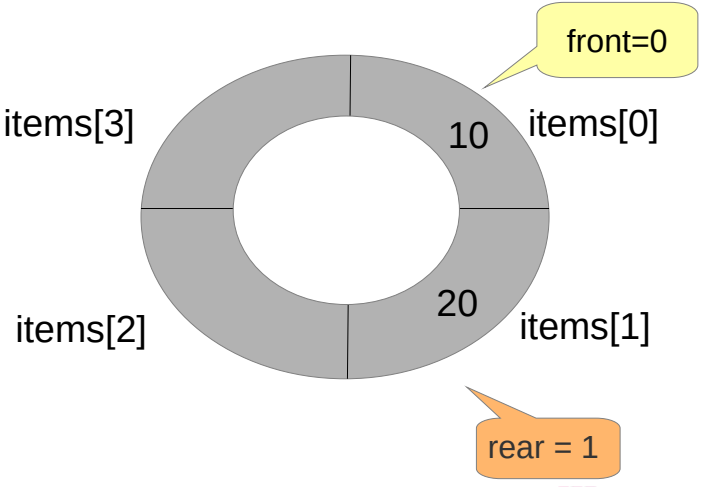
capacity = 4

count = 2

front = 0

rear = 1

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

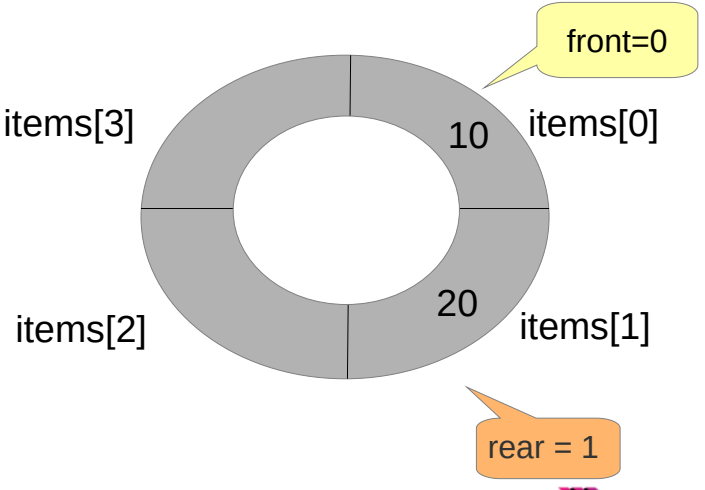
capacity = 4

count = 2

front = 0

rear = 2

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

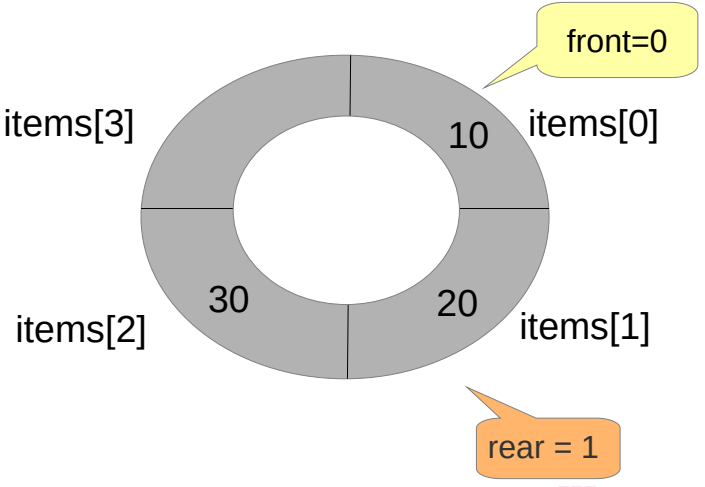
capacity = 4

count = 2

front = 0

rear = 2

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

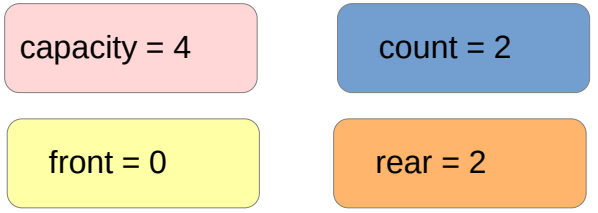
Data Structure –Array Implementation

enqueue(queue,element)

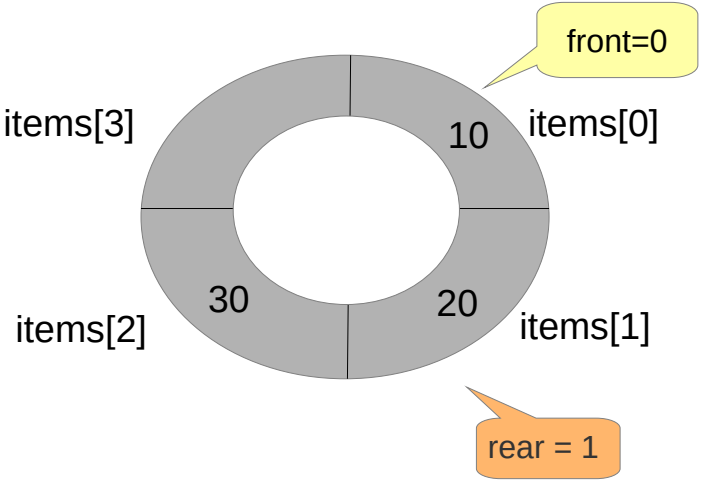
```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

is_queue_full(queue)

```
   if(queue ⟶ count = queue ⟶ capacity)
                return e_true
   else
      return e_false
```



element = 30



EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

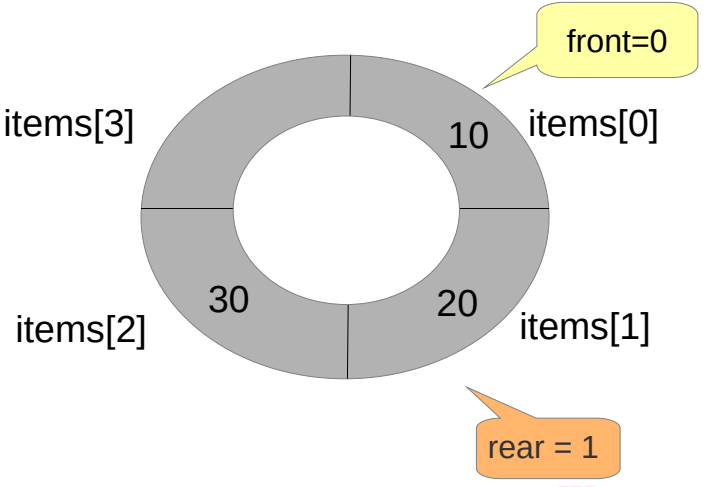
capacity = 4

count = 3

front = 0

rear = 2

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

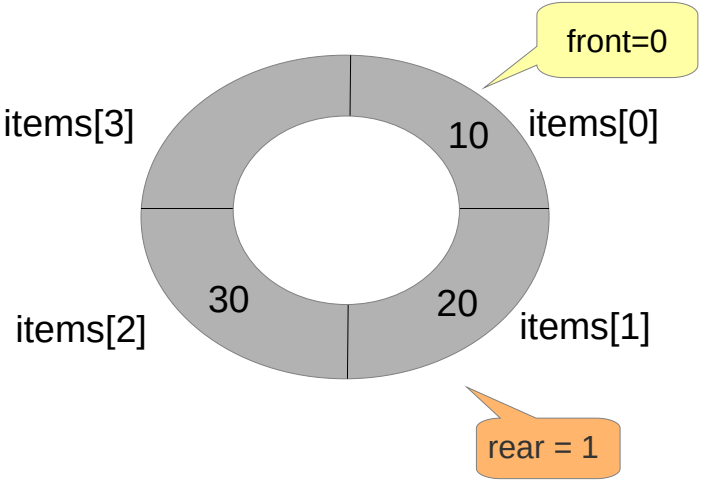
capacity = 4

count = 3

front = 0

rear = 2

element = 30



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

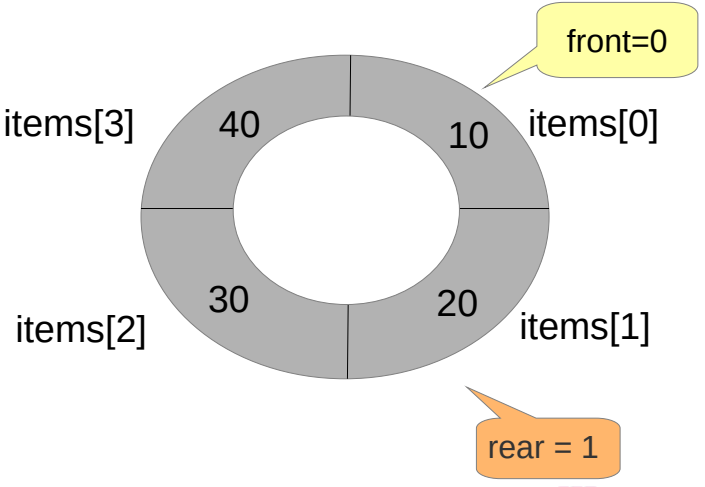
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

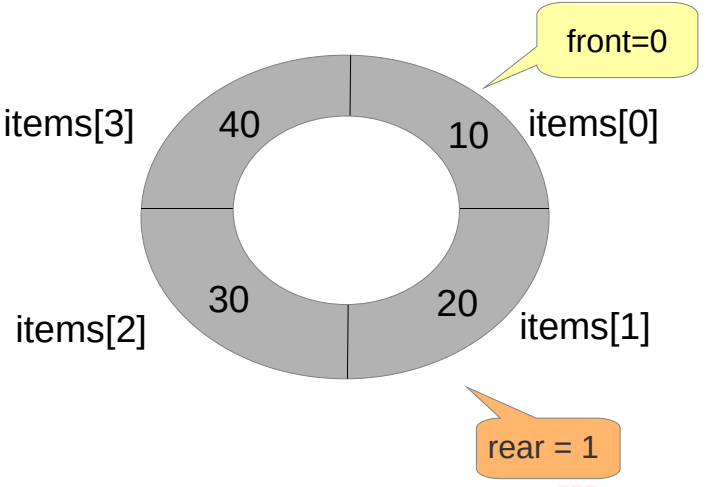
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

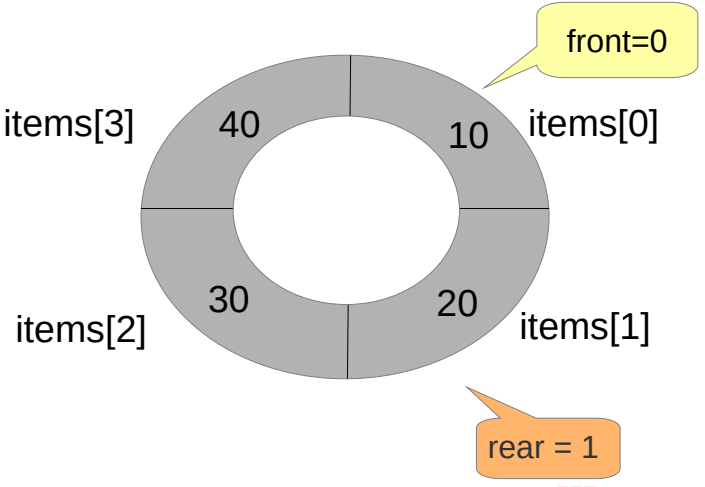
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
  else
     return e_false
```

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

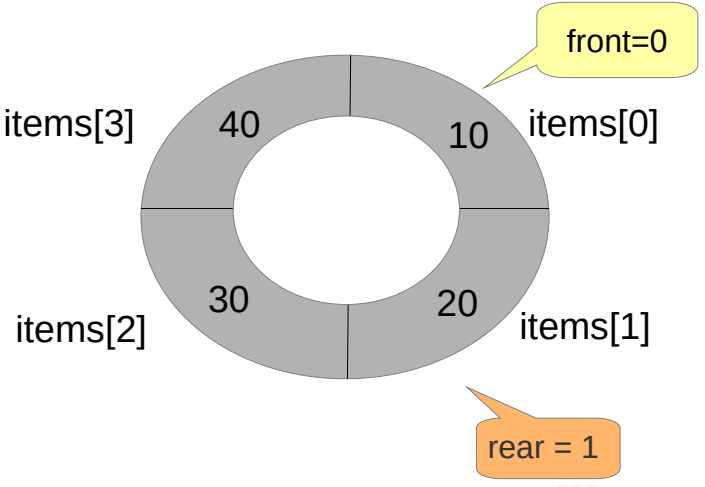
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
  else
     return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

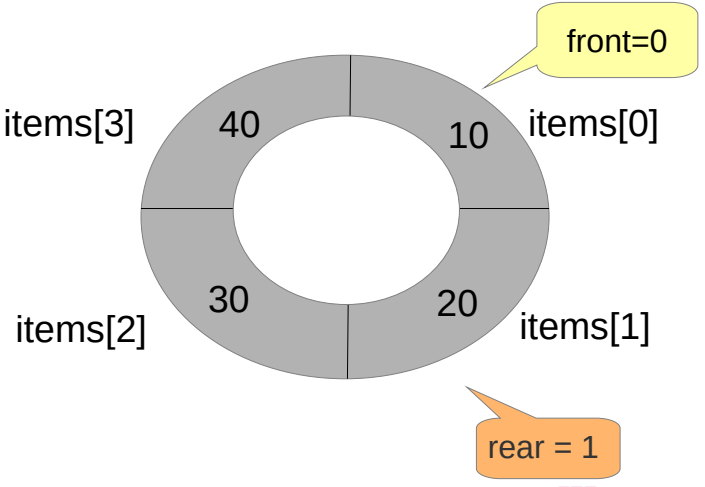
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
  else
     return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

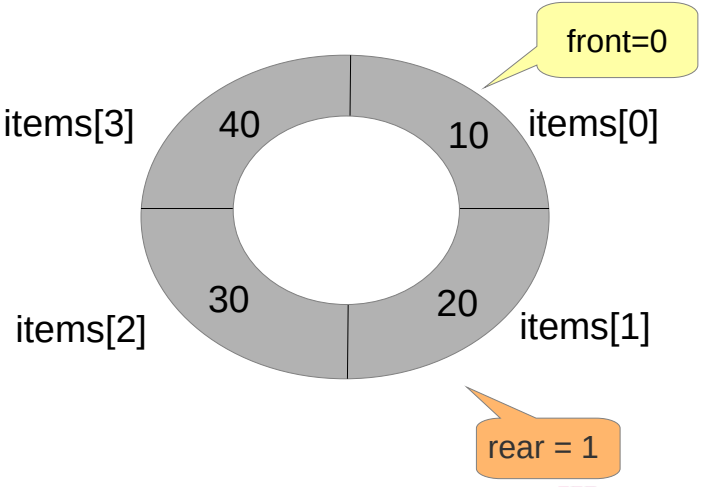
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
        return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

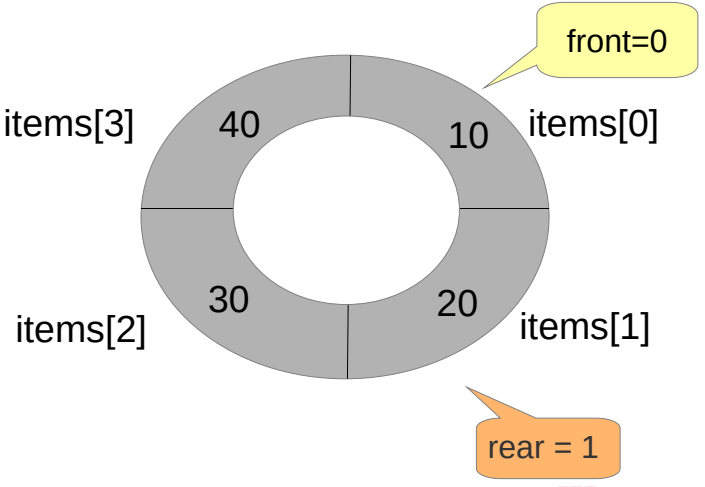
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

EMERTXE

Data Structure –Array Implementation

enqueue(queue,element)

```
if(is_queue_full(queue))
   return e_false
if(queue ⟶ front  = -1)
    queue ⟶ front = 0
queue  ⟶ rear = (queue ⟶ rear + 1 ) % (queue⟶  capacity )
queue  ⟶ items[queue ⟶ rear ] = element
++(queue ⟶ count)
return e_true
```

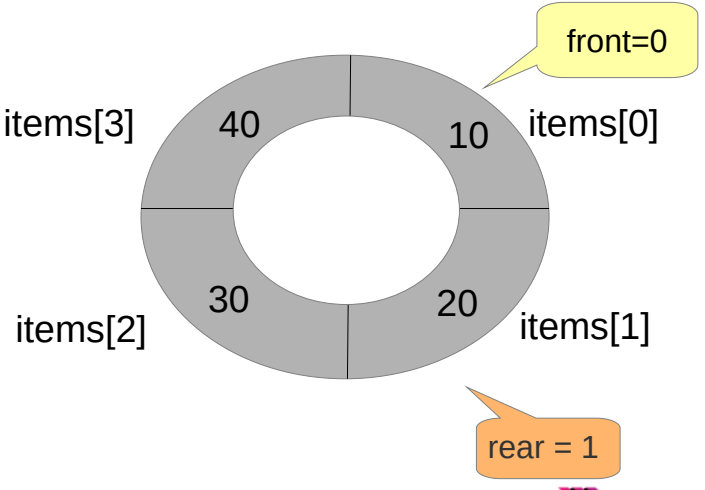
capacity = 4

count = 4

front = 0

rear = 3

element = 40



is_queue_full(queue)

```
if(queue ⟶ count = queue ⟶ capacity)
            return e_true
else
    return e_false
```

Queue is full

EMERTXE

Algorithm - dequeue