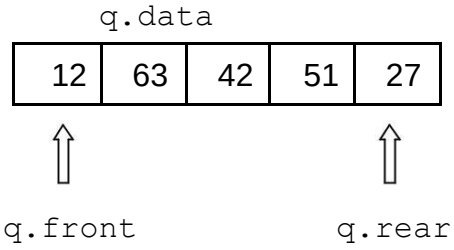
Queue Exercises

Example: Given 5 integers, insert them into a queue.

```
queue_t q;
int num, k;
initialize_q(&q);
for (k = 1; k <= 5; k++) {
     scanf("%d", &num);
     insert(&q, num);
}
```

- Assume that our input data is 12, 63, 42, 51, 27. After the loop terminates, all data will be put into the queue, and the queue will look like the following, if the QUEUE_SIZE was 5.



- If you are asked to fill a queue with data, no matter what the size of the queue is or no matter how many items the queue already contains, you can go on reading data until the queue becomes full. Therefore our program segment should be as follows:

```
while (!is_full_q(&q)) {
     scanf("%d", &num);
     insert(&q, num);
}
```

Example: Output the data in the queue.

- When outputting, there is an important decision to make: Do you want the data to remain in the queue, or do you want to empty the queue as you output? If you want to empty the queue each time you output the value, you need to remove it, and don't forget that this value will be lost after outputting.

```
while (!is_empty_q(&q))
     printf("%5d\n", remove(&q));
printf("\n");
```

- After the loop terminates, we will see the following on the screen:

  12 63 42 51 27

- Notice that, the output is in the same order as the input data.

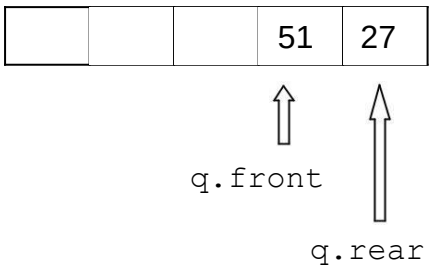
- If we don't want to destroy the queue, we can define the above program segment as a function, sending the queue as a value parameter, so that even if it becomes empty in the function, the actual queue will remain the same:

```
void display_q(queue_t q) {
     while (!is_empty_q(q))
         printf("%5d\n", remove(&q));
     printf("\n");
}
```

Example: Remove the first 3 items (means we want to remove 12, 63 and 42).

```
for (k = 1; k <= 3; k++)
     num = remove(&q);
```

- After the loop, the queue is as follows: `q.data`

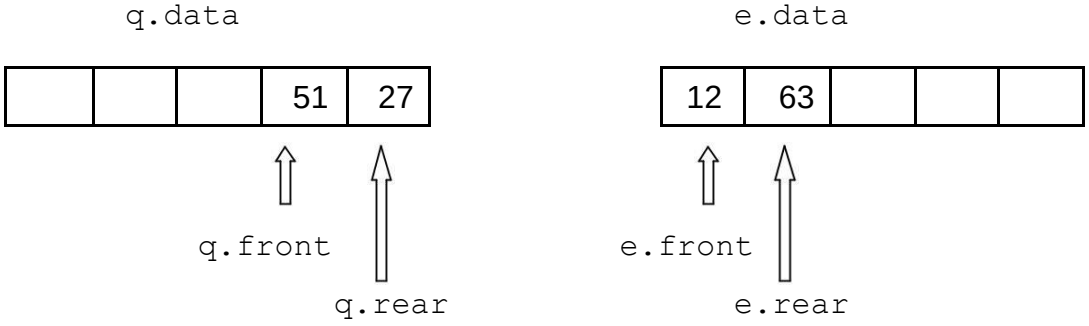


Example: Remove only the third item from the front (means we want to remove 42, but we want to keep the others).

- Consider our example queue. We want to delete the third item, 42. If we use an extra queue `e`, as follows:

```
queue_t e;
initialize_q(&e);
for (k = 1; k < 3; k++)
     insert(&e, remove(&q));
// Remove the third item from q, but do not insert it into
e num = remove(&q);
```

- At this point, the first three items are removed from `q`, and the first two are inserted into `e`. Thus, the two queues look like as follows:

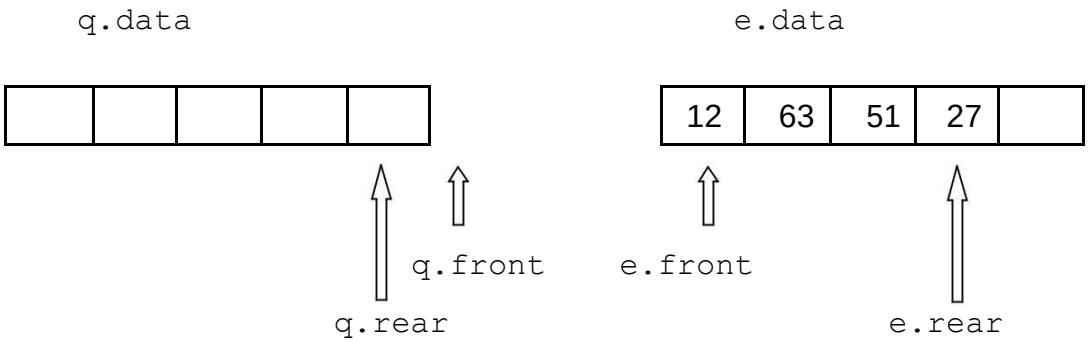


- Notice that, it is not possible to put back 12 and 63 into the original queue. Because, first of all, there is no space at the rear of the queue. Even if the data array was larger, or it was a circular queue, if we inserted 12 and 63 back into `q`, they would go to the end, instead of being at the front. Thus, the order of the items would change.

- So, we will go on removing items from `q`, and inserting them into `e`, until `q` becomes empty.

```
// Remove the remaining items from q, and insert them into
e while (!is_empty_q(&q))
     insert(&e, remove(&q));
```

- Now `q` is empty, and `e` contains all items except the third one. Thus, the two queues look like as follows:



- Now, we should copy all values in `e` back into `q`. Notice that, although `q` is empty now, we can not insert any items in it, because `q.rear` is equal to `QUEUE_SIZE-1`, thus the queue seems as if it is full. Therefore, first of all we have to initialize `q` as an empty queue, so that `front` and `rear` indices will go to the beginning of the queue. Notice that, this problem does not happen if we are using a circular queue.

```
// Copy all values in e back into q
initialize_q(&q); // needed for a linear queue
while (!is_empty_q(&e))
     insert(&q, remove(&e));
```

- Remember that, with two variables of the same structure, direct assignment is possible. Thus, we can copy all values in `e` back into `q` with a single assignment statement as:

```
q = e;
```

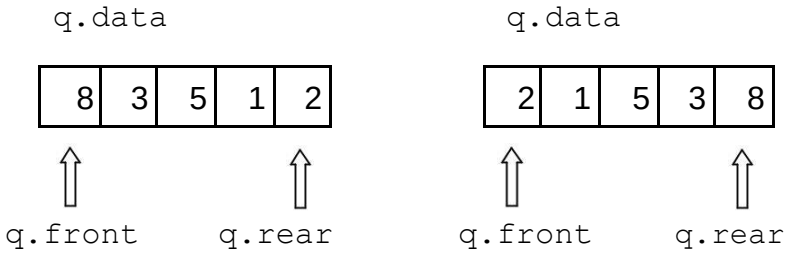
Home Exercises:

1) Define a function that removes all occurrences of a certain item from a queue, without changing the order of the other items.

2) Define a function that inserts an item after a certain item in a queue, without changing the order of the other items. Assume that there is enough space in the queue.

Example: Define a function that reverses a queue.

- For instance, if the given queue is as on the left, it should be as on the right after the function call.



- As you know, reversing operation can best be done using a stack. Thus, if we push each item of the queue onto an empty stack, and then pop the stack back into the queue, we can solve the problem.

- The function will take the queue as a reference parameter, and use a stack as a local variable. Since we need to use both a queue and a stack in our function, we should include the header files for both structures, and then define our function.

```
#include <stack_int.h>
#include <queue_int.h>

void reverse_queue(queue_t *q) {
     stack_t s; //local stack structure
     initialize_s(&s);
     /* Remove all items from the queue and push them
        onto the stack */
     while (!is_empty_q(q))
         push(&s, remove(q));
     initialize_q(q);
     /* Pop all items from the stack and insert them into
        the queue */
     while (!is_empty_s(&s))
         insert(q, pop(&s));
}
```

Example: Trace the following program segment, and show the output. Assume that `QUEUE_SIZE` is 5. (Remember that `q.rear` is initialized to -1 in `queue_int.h`)

```
#include <stack_int.h>
#include <queue_int.h>
...
stack_t s;
queue_t q;
int n;

initialize_s(&s);
initialize_q(&q);
```

```
while (!is_full_q(&q)) {
     insert(&q, q.rear);
     push(&s, s.top);
}

while (!is_empty_s(&s)) {
     n = pop(&s);
     printf("%d   %d\n", n, s.top);
     n = remove(&q);
     printf("%d   %d\n", n, q.front);
}
```

Home Exercise:

Write a C program that will get the queue values from the user until the user enters a sentinel value (-9). Then the program will display a menu with the options below:

1) Print Queue
2) Clear Queue
3) Count Queue
4) Remove Maximum Element
5) Send Nth To End
6) Exit

After getting the choice of the user, the program will do the operations according to the choice by using the functions following:

printQueue	: Display the elements of the queue (Queue content will not change)
clearQueue	: Remove all elements
countQueue	: Count the elements in the queue
remMaxQueue	: Remove the maximum element from the queue
sendNthToEnd	: Send the Nth element from the start to the end of the queue

Example Run:

```
Enter the numbers for the queue (-9 to stop):
3
4
5
2
-9


1) Print Queue
2) Clear Queue
3) Count Queue
4) Remove Maximum Element
5) Send Nth To End
6) Exit
Enter your choice: 1
3  4  5  2
```

```
Enter your choice: 3
Number of elements in the queue: 4

Enter your choice: 5
Enter N: 5
N must be between 1 and 4

Enter your choice: 5
Enter N: 2

Enter your choice: 1
3  5  2  4

Enter your choice: 4

Enter your choice: 1
3  2  4

Enter your choice: 2
The queue is empty!!

Enter your choice: 3
Number of elements in the queue: 0

Enter your choice: 7

Enter your choice: 6
```