```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "sequeue.h"

sequeue * queue_create() {
	sequeue *sq;

	if ((sq = (sequeue *)malloc(sizeof(sequeue))) == NULL) {
		printf("malloc failed\n");
		return NULL;
	}

	memset(sq->data, 0, sizeof(sq->data));
	sq->front = sq->rear = 0;
	return sq;
}

int enqueue(sequeue *sq, datatype x) {
	if (sq == NULL) {
		printf("sq is NULL\n");
		return -1;
	}

	if ((sq->rear + 1) % N == sq->front) {
		printf("sequeue is full\n");
		return -1;
	}

	sq->data[sq->rear] = x;
	sq->rear = (sq->rear + 1) % N;

	return  0;
}

datatype dequeue(sequeue *sq) {
	datatype ret;

	ret = sq->data[sq->front];

	sq->front = (sq->front + 1) % N;

	return ret;
}

int queue_empty(sequeue *sq) {
	if (sq == NULL) {
		printf("sq is NULL\n");
		return -1;
	}

	return (sq->front == sq->rear ? 1 : 0);
}

int queue_full(sequeue *sq) {
	if (sq == NULL) {
		printf("sq is NULL\n");
		return -1;
	}
```

malloc → sizeof 接 判

→ 初始化 → data
front
rear

→ 放数 → rear = value
改地址 → (rear + 1) % N

→ 放数 → ret = front
改地址 → (front + 1) % N

→ return front == rear

→ return (sq->rear + 1) % N == sq->front

```
	if ((sq->rear + 1) % N == sq->front) {
		return 1;
	}
	else {
		return 0;
	}
}

int queue_clear(sequeue *sq) {
	if (sq == NULL) {
		printf("sq is NULL\n");
		return -1;
	}

	sq->front = sq->rear = 0;

	return 0;
}

sequeue * queue_free(sequeue *sq) {
	if (sq == NULL) {
		printf("sq is NULL\n");
		return NULL;
	}

	free(sq);
	sq = NULL;

	return NULL;
}
```

→ front = rear = 0

→ free → sq (主动置空)