```
#include <stdio.h>
#include <stdlib.h>
#include "tree.h"
#include "linkqueue.h"

linkqueue * queue_create() {
      linkqueue *lq;

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

      lq->front = lq->rear = (linklist)malloc(sizeof(listnode));
      if (lq->front == NULL) {
            printf("malloc node failed\n");
            return NULL;
      }
      lq->front->data = 0;
      lq->front->next = NULL;

      return lq;
}

int enqueue(linkqueue *lq, datatype x) {
      linklist p;

      if (lq == NULL) {
            printf("lq is NULL\n");
            return -1;
      }

      if ((p = (linklist)malloc(sizeof(listnode))) == NULL) {
            printf("malloc node failed\n");
            return -1;
      }
      p->data = x;
      p->next = NULL;

      lq->rear->next = p;
      lq->rear = p;

      return 0;
}

datatype dequeue(linkqueue *lq) {
      linklist p;

      if (lq == NULL) {
            printf("lq is NULL\n");
            return NULL;
      }

      p = lq->front;
      lq->front = p->next;
      free(p);
      p = NULL;

      return (lq->front->data);
}

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

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

int queue_clear(linkqueue *lq) {
	linklist p;

	if (lq == NULL) {
		printf("lq is NULL\n");
		return -1;
	}

	while (lq->front->next) {
		p = lq->front;
		lq->front = p->next;
		//printf("clear free:%d\n", p->data);
		free(p);
		p = NULL;
	}
	return 0;
}

linkqueue * queue_free(linkqueue *lq) {
	linklist p;

	if (lq == NULL) {
		printf("lq is NULL\n");
		return NULL;
	}

	while (lq->front) {
		p = lq->front;
		lq->front = p->next;
		//printf("free:%d\n", p->data);
		free(p);
	}

	free(lq);
	lq = NULL;

	return NULL;
}
```