```
/*     bool.h        Simple boolean datatype */
#define TRUE    1
#define FALSE   0
typedef int boolean;
```

```
/*  queue.h          Struct for queue */
typedef struct {
           int *pq;        /* body of queue - array*/
           int first;      /* position of first element */
           int last;       /* position of last element */
           int count;      /* number of queue elements */
           int queuesize;  /* size of queue - array */
} queue;
```

```
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"
#include "bool.h"

void init_queue(int n, queue *q)           /* Initialisierung queue */
{
    q->first = 0;          /* Anfangsindex = 0 */
    q->last = n-1;         /* letztes Element */
    q->queuesize = n;      /* maximale Anzahl der Elemente */
    q->count = 0;          /* Anzahl Elemente in queue */
    q->pq = 0;            /* Array ohne Speicherplatz */
    if(n>0){
             q->pq = (int *)malloc(n*sizeof(int)); /* Array initialisieren */
    }
}

void enqueue(queue *q, int x)       /* Eintragen x in queue an Ende */
{
        if (q->count >= q->queuesize){ /* queue voll ? */
                     printf("Warning: queue overflow enqueue x=%d\n",x);
                     return;
              }
        q->last = (q->last+1) % q->queuesize; /* Modulo-Division */
        q->pq[ q->last ] = x;                 /* Eintragen x     */
        q->count = q->count + 1;              /* count erhoehen  */
}

int dequeue(queue *q)   /* Entfernen erstes Element aus queue */
{
        int x = 0;
        if (q->count <= 0){ printf("Warning: empty queue dequeue.\n");
                                  return 0;
              }
        x = q->pq[ q->first ];
        q->first = (q->first+1) % q->queuesize;
        q->count = q->count - 1;
        return x;
}
```

```
int empty(queue *q)  /* queue leer ? */
{
        if (q->count <= 0) return (TRUE);
        return (FALSE);
}

void print_queue(queue *q)  /* queue drucken */
{
        int i=0;
        if(!(q->count)) return;
        i=q->first;
        while (i != q->last) {
                printf("%d ",q->pq[i]);
                i = (i+1) % q->queuesize;
         }
         printf("%d\n",q->pq[i]);
}

int main(){
       int z=0, n=0;
       queue q;

       printf("Queuegroesse: ");
       scanf("%d", &n); while(getchar() != '\n');

       init_queue(n, &q);

       while(TRUE){
              printf("int-Zahl fuer Queue (F6=EOF): ");
              scanf("%d", &z); while(getchar() != '\n');
              if(feof(stdin)) break;

              enqueue(&q, z); z=0;
       }

       printf("entnommenes Element = %d\n",dequeue(&q));
       printf("entnommenes Element = %d\n",dequeue(&q));

       while(TRUE){
              printf("int-Zahl fuer Queue (F6=EOF): ");
              scanf("%d", &z); while(getchar() != '\n');
              if(feof(stdin)) break;

              enqueue(&q, z); z=0;
       }
       printf("Inhalt Queue der Groesse %d mit Anzahl %d: \n", q.queuesize, q.count);

       print_queue(&q);

       getchar();
}
```

```
/*
Queuegroesse: 4
int-Zahl fuer Queue (F6=EOF): 0
int-Zahl fuer Queue (F6=EOF): 1
int-Zahl fuer Queue (F6=EOF): 2
int-Zahl fuer Queue (F6=EOF): 3
int-Zahl fuer Queue (F6=EOF): 4
Warning: queue overflow enqueue x=4
int-Zahl fuer Queue (F6=EOF): ^Z
entnommenes Element = 0
entnommenes Element = 1
int-Zahl fuer Queue (F6=EOF): 0
int-Zahl fuer Queue (F6=EOF): ^Z
Inhalt Queue der Groesse 4 mit Anzahl 3:
2 3 0
*/
```