1、完善第 1 题程序。

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -2
typedef int Status;
typedef int QElemType;
typedef struct QNode
{
    QElemType       data;
    struct QNode    *next;
}QNode, *QueuePtr;
typedef struct
{
    QueuePtr front;     //队头指针
    QueuePtr rear;      //队尾指针
}LinkQueue;

Status InitQueue(LinkQueue &Q)
{   //构造一个空队列 Q
    Q.front=Q.rear=(QueuePtr)malloc(sizeof(QNode));
    if (!Q.front) exit(OVERFLOW);               //存储分配失败
    ____________
    return OK;
}

Status DestroyQueue(LinkQueue &Q)
{   //销毁队列 Q，Q 不再存在
    while(Q.front){
        ____________
        ____________
        ____________
    }
    return OK;
}

Status CLearQueue(LinkQueue &Q)
{   //将 Q 清为空队列
    QueuePtr p,q;
    if (!Q.front) exit(OVERFLOW);               //队列不存在
    p=Q.front->next;
    Q.front->next=NULL;
    ____________
    ____________
        ____________
```

```
        _____
    Q.rear=Q.front;
    return OK;
}

Status QueueEmpty(LinkQueue Q)
{   //若队列 Q 为空队列，则返回 TRUE，否则返回 FALSE
    if   _____   return TRUE;
    else return FALSE;
}

int QueueLength(LinkQueue Q)
{   //返回 Q 的元素个数，即队列的长度
    QueuePtr p;
    int n=0;
    _____
    _____
    _____
    _____
    return n;
}

Status GetHead(LinkQueue Q, QElemType &e)
{   //若队列不空，则用 e 返回 Q 的队头元素，并返回 OK，否则返回 ERROR
    QueuePtr p;
    if (Q.front==Q.rear) return ERROR;
    _____
    _____
    return OK;
}

Status EnQueue(LinkQueue &Q, QElemType e)
{   //插入元素 e 为 Q 的新的队尾元素
    QueuePtr p;
    p=(QueuePtr)malloc(sizeof(QNode));
    if (!p) exit(OVERFLOW);                    //存储分配失败
    _____
    _____
    _____
    return OK;
}

Status DeQueue(LinkQueue &Q, QElemType &e)
{   //若队列不空，则删除 Q 的队头元素，用 e 返回其值，并返回 OK；否则返回 ERROR
    QueuePtr p;
    if (Q.front==Q.rear) return ERROR;
    _____
    _____
```

```
    _____
    if (Q.rear==p)   _____
    free(p);
    return OK;
}

Status QueueTraverse(LinkQueue &Q)
{    //从队头到队尾依次输出队列元素的值
    QueuePtr p=Q.front->next;
    if (p==NULL) return ERROR;
    _____
    _____
    _____
    printf("\n");
    return OK;
}

void main()
{
    int i,n;
    QElemType k,h,a,f;
    LinkQueue Q;
    printf("创建一个空队列！\n");
    InitQueue(Q);
    printf("判断队列是否为空！\n");
    printf("QueueEmpty(Q)=%d\n",QueueEmpty(Q));
    printf("创建队列的元素个数：\n");
    scanf("%d", &n);
    printf("输入%d 个插入队列的元素的值：\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d", &k);
        EnQueue(Q, k);
    }
    printf("输出队列元素的值：\n");
    QueueTraverse(Q);
    printf("输入插入队列的元素的值：");
    scanf("%d", &h);
    EnQueue(Q, h);
    printf("输出插入一个队列元素后队列元素的值：\n");
    QueueTraverse(Q);
    DeQueue(Q, a);
    printf("输出第 1 个删除的队头元素的值：%d\n", a);
    DeQueue(Q, a);
    printf("输出第 2 个删除的队头元素的值：%d\n", a);
    printf("输出两次删除队头元素后队列的元素值：");
    QueueTraverse(Q);
    If (GetHead(Q, f))
```

```
        printf("输出队头元素的值：%d\n",f);
    printf("输出队列元素的个数：%d\n",QueueLength(Q));
    printf("将 Q 清为空队列！\n");
    CLearQueue(Q);
    printf("输出队列元素的个数：%d\n",QueueLength(Q));
    printf("判断队列是否为空！\n");
    printf("QueueEmpty(Q)=%d\n",QueueEmpty(Q));
}
```

2、完善第 2 题程序。

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define MAXQSIZE 100
typedef int Status;
typedef int QElemType;
typedef struct {
    QElemType    *base;     //初始化的动态分配存储空间
    int front;              //头指针，若队列不空，指向队列头元素
    int rear;               //尾指针，若队列不空，指向队列尾元素的下一个位置
}SqQueue;

Status InitQueue(SqQueue &Q)
{  //构造一个空队列 Q
    Q.base=(QElemType*)malloc(MAXQSIZE*sizeof(QElemType));
    if (!Q.base) exit(OVERFLOW);       //存储分配失败
    ____________
    return OK;
}

Status CLearQueue(SqQueue &Q)
{  //将 Q 清为空队列
    if (!Q.base) exit(OVERFLOW);            //队列不存在
    ____________
    return OK;
}

Status QueueEmpty(SqQueue Q)
{  //若队列 Q 为空队列，则返回 TRUE，否则返回 FALSE
    If  ____________  return TRUE;
    else return FALSE;
}
```

```
int QueueLength(SqQueue Q)
{   //返回 Q 的元素个数，即队列的长度
    return _____
}


Status GetHead(SqQueue Q, QElemType &e)
{   //若队列不空，则用 e 返回 Q 的队头元素，并返回 OK，否则返回 ERROR
    if (Q.front==Q.rear) return ERROR;
    _____
    return OK;
}

Status EnQueue(SqQueue &Q, QElemType e)
{   //插入元素 e 为 Q 的新的队尾元素
    if  _____  return ERROR; //队列满
    _____
    _____
    return OK;
}

Status DeQueue(SqQueue &Q, QElemType &e)
{   //若队列不空，则删除 Q 的队头元素，用 e 返回其值，并返回 OK；否则返回 ERROR
    if  _____  return ERROR;
    _____
    _____
    return OK;
}

Status QueueTraverse(SqQueue &Q)
{    //从队头到队尾依次输出队列元素的值
    int p;
    if (Q.front==Q.rear) return ERROR;
    p=Q.front;
    _____
    _____
    _____
    printf("\n");
    return OK;
}

void main()
{
    int i,n;
    QElemType k,h,a,f;
    SqQueue Q;
    printf("创建一个空队列！\n");
    InitQueue(Q);
```

```
    printf("判断队列是否为空！\n");
    printf("QueueEmpty(Q)=%d\n",QueueEmpty(Q));
    printf("创建队列的元素个数：\n");
    scanf("%d", &n);
    printf("输入%d 个插入队列的元素的值：\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d", &k);
        EnQueue(Q, k);
    }
    printf("输出队列元素的值：\n");
    QueueTraverse(Q);
    printf("输入插入队列的元素的值：");
    scanf("%d", &h);
    EnQueue(Q, h);
    printf("输出插入一个队列元素后队列元素的值：\n");
    QueueTraverse(Q);
    DeQueue(Q, a);
    printf("输出第 1 个删除的队头元素的值：%d\n", a);
    DeQueue(Q, a);
    printf("输出第 2 个删除的队头元素的值：%d\n", a);
    printf("输出两次删除队头元素后队列的元素值：");
    QueueTraverse(Q);
    If (GetHead(Q, f))
        printf("输出队头元素的值：%d\n",f);
    printf("输出队列元素的个数：%d\n",QueueLength(Q));
    printf("将 Q 清为空队列！\n");
    CLearQueue(Q);
    printf("输出队列元素的个数：%d\n",QueueLength(Q));
    printf("判断队列是否为空！\n");
    printf("QueueEmpty(Q)=%d\n",QueueEmpty(Q));
}
```