1. **试设计一个算法：判定给定的字符向量是否为回文。回文即正读和反读均相同的字符序列，比如字符序列“abba”和字符序列“abdba”均是回文，但字符序列“good”不是回文。提示：将一半字符入栈。**

解答：

#define Stack\_Size 100 //假定预分配的栈空间最多为100个元素

typedef struct{

ElemType \*elem; //栈存储空间基地址

int top; //栈顶指针，栈非空是指向栈顶元素

}SqStack;

//以上为顺序栈的存储结构定义

int IsHuiwen(ElemType \*t){

//判断t字符向量是否为回文，若是，返回1，否则返回0

InitStack\_Sq(&S);

len=strlen(t); //求向量长度

for (i=0; i<len/2; i++) //将一半字符入栈

Push( &S, t[i]);

while(!EmptyStack(&S)){

// 每弹出一个字符与相应字符比较

temp=Pop (&S);

if(temp!=S[i]) return 0; // 不等则返回0

else i++;

}

return 1; // 比较完毕均相等则返回 1

}

1. **假设以带头结点的循环链表表示队列，且只设一个指针指向队尾元素。试设计一个算法：对该循环队列进行置空队、判队空、入队和出队操作。**

解答：

typedef struct QNode {

ElemType data;

struct QNode \*next;

}QNode \*QueuePtr;

//以上是结点类型的定义

typedef struct {

QueuePtr rear; //只设一个指向队尾元素的指针

}LinkQueue;

void InitQueue\_L(LinkQueue &Q) {

//置空队：就是使头结点成为队尾元素

Q.rear=new QNode;

Q.rear = Q.rear->next;//将队尾指针指向头结点

while(Q.rear!=Q.rear->next){ //当队列非空，将队中元素逐个出队

s=Q.rear->next;

Q.rear->next=s->next;

free(s); //回收结点空间

}

}

bool EmptyQueue\_L(LinkQueue &Q) { //判队空

//当头结点的next指针指向自己时为空队

return Q.rear->next->next==Q.rear->next;

}

void EnQueue\_L(LinkQueue &Q, ElemType e) { //入队

//在尾结点处插入元素

p=new QNode;//申请新结点

p->data=e;

if(Q.rear==NULL){//处理空表

Q.rear=p;

Q.rear->next=p;

}

else{

p->next=Q.rear->next;//初始化新结点并链入

Q.rear->next=p;

Q.rear=p;//将尾指针移至新结点

}

void DeQueue\_L(LinkQueue &Q,ElemType &e) { //出队，把头结点之后的元素摘下

//若队列为空，给出相应信息并退出运行，否则用e返回队头结点数据域的值

if(EmptyQueue( Q )) //判断表空

Error(“Queue Empty!”);

else{

e=p->data; //保存结点中数据

if (Q.rear==p) //当队列中只有一个结点时

Q.rear=NULL;

else

Q.rear->next->next=p->next;//摘下结点p

delete p;//释放被删结点

}

//return e;

}

**3.假设顺序栈S中有2n个元素，从栈顶到栈底的元素依次为a2n、a2n-1、…、a1。试设计一个算法：通过一个循环队列Q重新排列该栈中的元素，使得从栈顶到栈底的元素依次为a2n、a2n-2、…、a2、a2n-1、a2n-3、…、a1。要求空间复杂度和时间复杂度均为O(n)。**

算法概述：

1. 将所有元素出栈入队；
2. 依次将队列元素出队，如果是偶数结点则再入队；如果是奇数结点则入栈；
3. 将奇数结点出栈并入队；
4. 将偶数结点出队并入栈；
5. 将所有元素出栈并入队；
6. 将所有元素出队并入栈。

解答：

#define MAXSIZE 100

typedef struct{

ElemType \*elem;

int top;

Int stacksize;

}SqStack;

//以上为定义栈结构

typedef struct{

ElemType \*elem;

int front;

int rear;

}SqQueue;

//以上为定义循环队列结构

void Rearrange\_Sq(SqStack &S){

InitQueue\_Sq(Q);

for(i=0;i<2\*n;i++){ //将所有元素出栈入队

p=arrange[i];

EnQueue\_Sq(Q,Pop\_Sq(S,p));

}

for(i=0;i<2\*n;i++){ //依次将队列元素出队

if(i%2==0)

EnQueue\_Sq(Q,DeQueue\_Sq(Q,p)); //如果是偶数结点则再入队

else

Push\_Sq(S,DeQueue\_Sq(Q,p)); //如果是奇数结点则入栈

}

for(i=0;i<n;i++){ //将奇数结点出栈并入队

EnQueue\_Sq(Q,Pop\_Sq(S,p));

}

for(i=0;i<n;i++){ //将偶数结点出队并入栈

Push\_Sq(S,DeQueue\_Sq(Q,p));

}

for(i=0;i<n;i++){ //将所有元素出栈并入队

EnQueue\_Sq(Q,Pop\_Sq(S,p));

}

for(i=0;i<2\*n;i++){ //将所有元素出队并入栈

Push\_Sq(S,DeQueue\_Sq(Q,p));

}

}