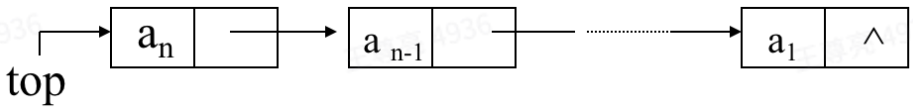
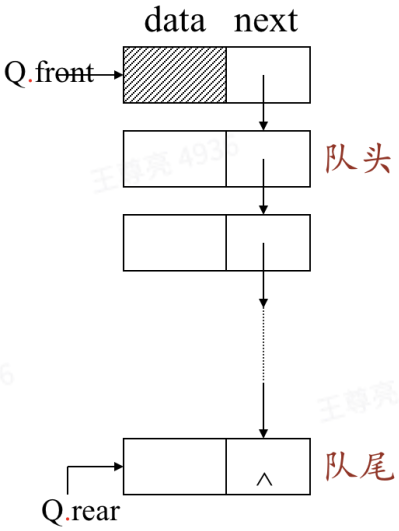
第三、四章 答案

一、选择题

1. 向一个栈顶指针为top的链栈中插入一个p所指结点时，其操作步骤为（ C ）
   a. top->next=p;
   b. p->next=top->next; top->next=p;
   c. p->next=top; top=p;
   d. p->next=top; top=top->next;



2. 一个栈的入栈序列是a,b,c,d,e,则栈的不可能的输出序列是（ C ）
   a. edcba;
   b. decba;
   c. dceab;
   d. abcde;
3. 在一个链队列中，若f,r分别为队首、队尾指针，则插入p所指结点的操作为（ B ）
   a. f->next=p; f=p;
   b. r->next=p; r=p;
   c. p->next=r; r=p;
   d. p->next=f; f=p;



4. 用不带头结点的单链表存储队列时，在进行删除运算时（ D ）
   a. 仅修改头指针

b. 仅修改尾指针;

c. 头尾指针都要修改;

d. 头、尾指针可能都要修改;

二、简答题

1. 简述以下算法的功能

C

```c
Status algo1(Stack S){
    int i,n, A[255];
    n=0;
    while(!StackEmpty(S)){n++; Pop(S, A[n]);};
    for(i=1; i<=n; i++) Push(S,A[i]);
}
```

答：利用数组辅助将栈中的数据元素逆置

2. 简述以下算法的功能

C

```c
Status algo2(Stack S, int e){
    Stack T; int d;
    InitStack(T);
    while(!StackEmpty(S)){
        Pop(S,d);
        if(d!=e) Push(T,d);
    }
    while(!StackEmpty(T)){
        Pop(T, d);
        Push(S,d);
    }
}
```

答：利用栈T辅助将栈S中所有值为e的数据元素删除

3. 简述以下算法的功能

C

```c
void algo3(Queue &Q){
    Stack S; int d;
    InitStack(S);
    while(!QeueuEmpty(Q)){
        DeQueue(Q,d);
        Push(S,d);
    }
    while(!StackEmpty(S)){
        Pop(S,d);
        EnQueue(Q,d);
    }
}
```

答：利用栈T辅助将队列中的数据元素进行逆置

4. 画出对表达式：3-2*8/4+3^2 求值时，操作数栈和运算符栈的变化过程。

读字符	运算对象栈	运算符栈	说明
		(	为了使第一个运算符入栈，预设一个最低级运算符
3	3	(	3入运算对象栈
-	3	(-	-入运算符栈
2	3 2	(-	2入运算对象栈
*	3 2	(-*	*入运算符栈
8	3 2 8	(-*	8入运算对象栈
/	3 16	(-	/优先级低于运算符栈栈顶元素*，则出栈两个运算对象，出栈一个运算符，即计算2*8=16，然后将16入栈
	3 16	(-/	/优先级高于运算符栈顶元素-，则入栈运算符
4	3 16 4	(-/	4入运算对象栈
+	3 4	(-	+优先级低于运算符栈栈顶元素/，则出栈两个运算对象，出栈一个运算符，即计算16/4=4，然后将4入栈
	-1	(	+优先级低于运算符栈栈顶元素-，则出栈两个运算对象，出栈一个运算符，即计算3-4=-1，然后将-1入栈
	-1	(+	+优先级高于运算符栈栈顶元素(, + 入运算符栈

3	-1 3	(+	3入运算对象栈
^	-1 3	(+ ^	^优先级高于运算符栈栈顶元素+, ^入运算符栈
2	-1 3 2	(+ ^	2入运算对象栈
)	-1 9	(+	)优先级低于运算符栈栈顶元素^，则出栈两个运算对象，出栈一个运算符，即计算3^2=9，然后将9入栈
	8	(	)优先级低于运算符栈栈顶元素+，则出栈两个运算对象，出栈一个运算符，即计算-1+9=8
			()配对，出栈一个运算对象，8

5. 已知主串 s = 'ADBADABBAABADABBADADA', 模式串pat = 'ADABBADADA' ,写出模式串的nextval函数值，并由此画出KMP算法匹配的过程

j	1	2	3	4	5	6	7	8	9	10
模式	A	D	A	B	B	A	D	A	D	A
next[j]	0	1	1	2	1	1	2	3	4	3
nextval[j]	0	1	0	2	1	0	1	0	4	0

   a. 从i=1，j=1开始比对，当i=3，j=3时失配，然后j=nextval[3] = 0;
   b. 从i=4，j=1继续比对，当i=10, j=7时失配，然后j=nextval[7] = 1;
   c. 从i=10, j=1继续比对，当i=11，j=2时失配，然后j=nextval[2] =1;
   d. 从i=11, j=1继续比对，直接失配，然后j=nextval[1] =0;
   e. 从i=12，j=1继续比对，匹配成功

三、算法设计题

1. 假设以顺序存储结构实现一个双向栈，即在一维数组的存储空间中存在着两个栈，它们的栈底分别设在数组的两个端点。试编写实现这个双向栈tws的三个操作：
   a. 初始化栈InitStack(&tws);
   b. 入栈push(&tws, i, e);
   c. 出栈pop(&tws, i, &e);

   其中i为0或1，分别指示设在数组两端的两个栈

C

```c
typedef struct{
    ElemType * elem;
    int top1;
    int top2;
}TWStack;

Status InitStack(TWStack &tws){
    tws.elem = (ElemType *)malloc(MAXSIZE*sizeof(ElemType));
    if(!tws.elem) exit(OVERFLOW);
    tws.top1 = -1;
    tws.top2 = MAXSIZE;
    return OK;
}

Status push(TWStack &tws, i,  e){
    if(tws.top1 + 1 == tws.top2)
        return ERROR;
    if(i == 1){
        tws.elem[++tws.top1] = e;
    }else if(i == 2){
         tws.elem[--tws.top2] = e;
    }else{
        return ERROR;
    }
    return OK;
}

Status pop(TWStack &tws, i,  &e){
    if(i == 1){
        if(tws.top1 == -1) return ERROR;
        e = tws.elem[tws.top1--]
    }else if(i == 2){
        if(tws.top2 == MAXSIZE) return ERROR;
         e = tws.elem[tws.top2++]
    }else{
        return ERROR;
    }
    return OK;
}
```

2. 假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾元素结点（注意不设头指针），试编写相应的队列初始化、入队列和出队列的算法。

   3.28//注意队列中只有一个元素时需要单独处理

C

```c
typedef  struct QNode{
    QElemtype     data;
    struct QNode  *next;
}QNode, *QueuePtr;

Status InitLinkQueue(QueuePtr &rear){
     rear = (QNode *)malloc(sizeof(QNode); //头结点
     if(!rear) exit(OVERFLOW);
     rear->next = rear;
     return OK;
 }

Status EnLinkQueue(QueuePtr &rear, QElemtype e){
     QueuePtr p = (QNode *)malloc(sizeof(QNode);
     if(!p) exit(OVERFLOW);
     p->data = e;
     p->next = rear->next;
     rear->next = p;
     rear = p;
     return OK;
 }

Status DeLinkQueue(QueuePtr &rear, QElemtype &e){
    if(!rear || rear->next == rear) return ERROR;
     QueuePtr p = rear->next->next; //p指向队头结点
     e = p->data;
     rear->next->next = p->next;
     if(rear == p) rear = rear->next;//仅剩最后一个结点
     free(p);
     return OK;
 }
```

3. 编写算法，从串s中删除所有和串t相同的子串。