**第三章作业答案**

**一、思考题**

栈与队列

1、设abcdef 以所给的次序进栈，若在进栈操作时，允许退栈操作，则下面得不到的序列为（ ） 。

(A)fedcba (B) bcafed (C) dcefba (D) cabdef

2、若已知一个栈的入栈序列是 1,2,3,…,n，其输出序列为 p1,p2,p3，…，pN,若pN 是 n，则 pi 是( )。

(A) i (B) n-i (C) n-i+1 (D) 不确定

3、设计一个判别表达式中左，右括号是否配对出现的算法，采用（ ）数据结构最佳。

(A)线性表的顺序存储结构 (B) 队列

(C) 线性表的链式存储结构 (D) 栈

4、用链接方式存储的队列，在进行删除运算时（ ） 。

(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

5、递归过程或函数调用时，处理参数及返回地址，要用一种称为（ ）的数据结构。

(A)队列 (B)多维数组 (C)栈 (D) 线性表

6、假设以数组 A[m]存放循环队列的元素,其头尾指针分别为 front 和rear，则当前队列中的元素个数为（ ） 。

(A)(rear-front+m)%m (B)rear-front+1

(C)(front-rear+m)%m (D)(rear-front)%m

7、若用一个大小为 6的数组来实现循环队列，且当前 rear和 front 的值分别为 0和 3，当从队列中删除一个元素，再加入两个元素后，rear 和 front 的值分别为多少？( )

(A) 1 和 5 (B) 2 和4 (C) 4 和2 (D) 5 和1

8、最大容量为 n 的循环队列，队尾指针是 rear，队头是 front，则队空的条件是（ ） 。

(A) (rear+1) MOD n=front (B) rear=front

(C)rear+1=front (D) (rear-l) MOD n=front

9、栈和队列的共同点是（ ） 。

(A) 都是先进先出 (B) 都是先进后出

(C) 只允许在端点处插入和删除元素 (D) 没有共同点

10、设栈S和队列 Q 的初始状态为空，元素 e1，e2，e3，e4,e5 和e6 依次通过栈 S，一个元素出栈后即进队列 Q，若 6 个元素出队的序列是 e2，e4，e3,e6,e5,e1 则栈 S 的容量至少应该是( )。

(A) 6 (B) 4 (C) 3 (D) 2

**二、应用题**

1、编号为1,2,3,4的四辆列车，顺序开进一个栈式结构的站台，问开出车站的顺序有多少种可能？(设栈容量为2)

第一种情况

|  |
| --- |
|  |
| 1 |

（1）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2 | 4 | 3 |
| **1** | 3 | 4 |
| 3 | 4 | 2 |
|  | 2 | 4 |

第二种情况

|  |
| --- |
| 2 |
| 1 |

（2）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 4 | 3 |
| **2** | 3 | 4 |
| 3 | 4 | 1 |
|  | 1 | 4 |

2、设表达式的中缀表示为a\*x-b/x\*x，试利用栈将它改为后缀表达式ax\*bxx\*/-。写出转换过程中栈的变化。

|  |  |  |  |
| --- | --- | --- | --- |
| 扫描到的元素 | 表达式数组 | 运算符栈 | 说明 |
| a | a | null | 操作数保存 |
| \* | a | \* | 栈空，运算符直接入栈 |
| x | ax | \* | 操作数保存 |
| - | ax\* | - | 优先级小于栈顶运算符，弹出\*，压入- |
| b | ax\*b | - | 操作数保存 |
| / | ax\*b | -/ | 优先级大于栈顶，压入堆栈 |
| x | ax\*bx | -/ | 操作数保存 |
| \* | ax\*bx | -/\* | 优先级等于栈顶，弹出/,压入\* |
| x | ax\*bxx | -/\* | 操作数保存 |
| # | ax\*bxx\*/- | null | 弹出所有符号 |

4. 写出下列中缀表达式的后缀形式：

（1）A \* B \* C （2）- A + B – C + D （3）A && B|| !(E>F)

（1）A输出，\*进栈

B输出，\*出栈并输出，\*进栈

C输出，\*出栈并输出

结果为：AB\*C\*

（2）-进栈

A输出，-出栈并输出，+进栈

B输出，+出栈并输出，-进栈

C输出，-出栈并输出，+进栈

D输出，+出栈并输出

结果为：A-B+C-D+

（3）A输出，&&进栈

B输出，&&出栈并输出，||进栈，!进栈，(进栈

E输出，>进栈

F输出，遇“）”将>出栈并输出，(出栈，!出栈并输出，||出栈并输出

结果为：AB&&EF>! ||

**三、算法设计**

1. 判断长度为n的字符串（放单链表中）是否中心对称

栈s[N] (N≧n/2)

|  |
| --- |
|  |
|  |
| **Top** |
|  |

|  |
| --- |
| 函数接口描述： |
| **输入： 链表地址head、 字符串长度n;** |
| **输出： “对称” 或 “非对称”;** |
| 数据结构描述： |
| **// 链表结点描述** |
| **typedef struct node** |
| **{ datatype data;** |
| **struct node \*next;** |
| **} LinkList;** |
| **// 栈描述** |
| **#define MAX 100；** |
| **typedef struct** |
| **{ datatype array[MAX];** |
| **int top;** |
| **} SeqStack;** |
| 算法描述：   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | **C** | **B** | **A** |  | **A** | **B** | **C** | **C** | **B** | **A** | |
| **顶部伪代码描述：** |
| **将链表中n/2的结点顺序入栈， p指向链表的（n/2+n%2+1）结点处，顺次比较s[top]与p->data，直至栈空。（结点从1开始计数）** |
| 第一步细化描述 |
| **p=head top=0** |
| **当（top< n/2 ）** |
| **{ 链表中p结点字符入栈;** |
| **p指向下一结点}** |
| **若n为奇数，则 p指向下一结点** |
| **当 （ 链结点值=栈顶元素值）** |
| **{ p指向下一结点；** |
| **出栈操作一次；** |
| **若到栈底，显示 “对称”，返回** |
| **}** |
| **显示“非对称”，返回** |
| 第二步细化描述 |
| **p=head; top=0;** |
| **while（top< n/2 ）** |
| **{ s[top]=p->data；** |
| **top++; p=p->next; }** |
| **if (n%2==1) p=p->next;** |
| **while ( p->data==s[top] )** |
| **{ p=p->next;** |
| **top--;** |
| **if (top==-1) return(“对称”);** |
| **}** |
| **return(“非对称”);** |

代码实现：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MAX 100

typedef char DataType;

typedef struct

{

DataType data[MAX];

int top;

}SeqStack;

void Init(SeqStack \*&s)

{

s=(SeqStack \*)malloc(sizeof(SeqStack));

s->top=-1;

}

void push(SeqStack \*s,DataType e)

{

if( s->top == MAX )

{

printf("栈满\n");

}

else

{

s->top++;

s->data[s->top]=e;

}

}

DataType pop(SeqStack \*s)

{

if( s->top == -1 )

{

printf("栈空\n");

return '\0';

}

return (s->data[s->top--]);

}

int main()

{

SeqStack \*s;

int i=0,length,mid,flag=1;

char str[MAX],y,z;

printf("请输入需要判断的字符串：\n");

gets(str);

length = strlen(str);

mid = length/2-1;

Init(s);

for( i=0; i<length/2; i++)

{

push(s,str[i]);

}

if( length%2 != 0 )

mid += 1;

for( i=1; i<=length/2; i++)

{

if( str[mid+i] == pop(s) )

{

flag = 1;

}

else

{

flag = 0;

break;

}

}

if( flag == 1 )

printf("yes\n");

else

printf("no\n");

return 0;

}