

安徽大学 2020—2021 学年第 2 学期

《 数据结构 》 期中考试试卷参考答案

一、算法分析题（每题 10 分，共 30 分）

得分	
----	--

1. 请仔细阅读如下的算法，分析其算法时间复杂度。

(1)

```
void test1(int n){
    int a = 100;
    int b = 50;
    while(n > 0){
        n = n/2;
        if (a > n){
            a = a/3;
            b = b*3;
        }else{
            a = a*3;
            b = b/3;
        }
    }
}
```

答：while 循环共执行  $\log_2 n$  次，因而算法复杂度为  $O(\log_2 n)$ 。（5 分）

(2)

```
void test2(int A[n][n], int B[n][n], int C[n][n])
{
    int i, j, k;
    for(i=0; i<n; i++) {
        for (j=0; j<i; j++) {
            C[i][j] = 0;
            for (k=0; k<n; k++) {
                C[i][j] = C[i][j] + A[i][k] * B[k][j];
            }
        }
    }
}
```

答：最内层循环执行频次为  $(n^3 - n^2) / 2$ ，因而时间复杂度为  $O(n^3)$ ，直接写出复杂度也可以得分。（5 分）

2. 请仔细阅读下列算法，分析其完成了什么功能。

(1) 下面算法中 L 为带头结点的单链表

```
void test3(LinkList &L)
{
    p=L->next;  L->next=NULL;
    while ( p) {
        q=p->next;
        p->next=L->next;
        L->next=p;
        p = q;
    }
}
```

答：将链表中所有结点的链接方向逆转。(5 分)

(2) 下面算法中 L 为带头结点的有序递增单链表

```
void test4(LinkList &L, int mink, int maxk) {
    p=L->next;
    while (p && p->data<=mink) {
        pre=p;
        p=p->next;
    }
    if (p) {
        while (p && p->data<maxk)  p=p->next;
        q=pre->next;
        pre->next=p;
        while (q!=p) {
            s=q->next;
            delete q;
            q=s;
        }
    }
}
```

答：删除递增有序链表中值大于 mink 且小于 maxk 的所有元素。(5 分)

3. 回文是指正读反读均相同的字符序列，如“abba”和“abdba”均是回文，但“good”不是回文。如下算法判定给定的字符向量是否为回文，请仔细阅读代码并回答问题。

```
#define StackSize 100 //假定预分配的栈空间最多为 100 个元素
typedef char DataType; //假定栈元素的数据类型为字符
typedef struct {
    DataType data[StackSize];
    int top;
} SeqStack;
```

```

int IsHuiwen( char *t) { //判断 t 字符向量是否为回文，若是，返回 1，否则返回 0
    SeqStack s;
    int i , len;
    char temp;
    InitStack( &s);
    len=strlen(t); //求向量长度
    for ( i=0; i<len/2; i++)
        Push( &s, t[i]); (2分)
    while( !EmptyStack( &s)) { // 每弹出一个字符与相应字符比较 (2分)
        temp=Pop ( &s); (2分)
        if( temp!=S[i]) return 0 ; (2分)
        else i++;
    }
    return 1 ; (2分)
}

```

- (1) 请在代码中//后的空格处补充注释，说明该 while 语句的功能。  
 (2) 根据代码的功能，请在 4 处空白代码处填充合适的代码。

## 二、简答题（4 小题，共 40 分）

得分	
----	--

4. 假设广义表  $L=((a,b,c),d,(e,(f,g,(h))))$ ，请回答如下问题。

(1) 求  $\text{Head}(\text{Tail}(\text{Head}(\text{Tail}(\text{Tail}(L))))$  的值。

答：(f, g, (h)) (4 分)

(2) 写出使用多次嵌套的 Head() 和 Tail() 函数如何计算可以获得结果为 (h)？

答：Head(Tail(Tail(Head(Tail(Head(Tail(Tail(L))))))) (4 分)

5. 写出模式串“abcaabbcab”的 next 函数值和 nextval 函数值。(10 分，每空 0.5 分，可四舍五入)

j	1	2	3	4	5	6	7	8	9	10
模式串	a	b	c	a	a	b	b	a	b	c
next	0	1	1	1	2	2	3	1	2	3
nextval	0	1	1	0	2	1	3	0	1	1

6. 栈的数据结构可以用于对包含括号的四则运算表达式实现求值运算，使用两个栈，其中 OPTR 栈用来暂存操作符，OPND 用来暂存操作数，请补充完成如下表达式的求值过程：

# 3\*(4+2\*2)-5#

(14 分，每行 1 分)

OPTR	OPND	INPUT	OPERATION
#		3*(4+2*2)-5#	Push (OPND, ' 3' )
#	3	*(4+2*2)-5) #	Push (OPTR, '*' )
#, *	3	(4+2*2)-5#	Push (OPTR, '(' )
#, *, (	3	4+2*2)-5#	Push (OPND, ' 4' )
#, *, (	3, 4	+2*2)-5#	Push (OPTR, '+' )
#, *, (, +	3, 4	2*2)-5#	Push (OPND, ' 2' )
#, *, (, +	3, 4, 2	*2)-5#	Push (OPTR, '*' )
#, *, (, +, *	3, 4, 2	2)-5#	Push (OPND, ' 2' )
#, *, (, +, *	3, 4, 2, 2	) -5#	Operate (2*2)
#, *, (, +	3, 4, 4	) -5#	Operate (4+4)
#, *, (	3, 8	) -5#	Pop (OPTR)
#, *	3, 8	-5#	Operate (3*8)
#	24	-5#	Push (OPTR, '-' )
#, -	24	5#	Push (OPND, ' 5' )
#, -	24, 5	#	Operate (24-5)
#	19	#	GetTop (OPND)

7. 二维数组 A[10][10] 是一个对称矩阵，按照压缩存储时只存储下三角，按照行优先的顺序 A[0][0] 存储在起始地址为 1000 的存储空间中，每个元素占用 4 个单位的空间。

(1) 保存该数组一共需要多少个单位的存储空间？

答：4\*(1+2+3+...+10)=4\*55=220 (4 分)

(2) 元素 A[5][8] 存储的起始位置是多少？

答：A[5][8] 存储在 A[8][5] 的位置，所以  
1000+4\*(1+2+3+4+5+6+7+8+5)=1164 (4 分)

### 三、算法设计题（每小题 10 分，共 30 分）

得分	
----	--

8. 假设某个栈中已存在很多整数，请编写一个算法，将该栈中的所有偶数排在所有奇数之后，且保持原来偶数之间和奇数之间的前后位置不变。

提示：可以使用如下函数：

```
Status InitStack( Stack &S )
bool StackEmpty( Stack S )
Status DestroyStack( Stack &S )
Status Push( Stack &S, SElemType e)
Status Pop( Stack &S, SElemType &e)
```

```
void ReOrder(Stack &S)
{
    Stack Even, Odd;
    int e;
    InitStack(Even);
    InitStack(Odd);
    while(!StackEmpty(S))
    {
        Pop(S, &e);
        if (e%2==0) //偶数
            Push(Even, e);
        else
            Push(Odd, e);
    }
    While(!StackEmpty(Odd))
    {
        Pop(Odd, &e);
        Push(S, e);
    }
    While(!StackEmpty(Even))
    {
        Pop(Even, &e);
        Push(S, e);
    }
    DestroyStack(Even);
    DestroyStack(Odd);
}
```

9. 设计一个算法删除带头结点的单链表 L 中第一个值为 x 的结点的前驱结点指针，若不存在返回 NULL。

```
LinkNode* FindPre(LinkList L, ElemType x)
{
    LinkNode *prepre = L, *pre = prepre->next, *p;
    if(pre==NULL || pre->data == x)
        return NULL; //链表为空或第一个节点为 x，则说明没有能删除的前驱结点
```

```

    p = pre->next;
    while((p!=NULL)&&(p->data!=x))
    {
        prepre=pre;
        pre = p;
        p=p->next;
    }
    if(p!=NULL)
    {
        prepre->next = p;
        free(pre);
        return p;
    }
    else
        return NULL;
}

```

10. 设计一个在带头结点的单链表 L 中删除最小值结点 (如果有多个最小值, 则只删除第一个出现的最小值) 的算法。

```

void DeleteMinimum(LinkList &L)
{
    LinkNode *pre = L, *p = pre->next, *minpre = L, *minp = minpre->next;
    while(p!=NULL)
    {
        if(p->data < minp->data)
        {
            minp = p;
            minpre = pre;
        }
        pre = p;
        p=p->next;
    }
    if(minp != NULL)
    {
        minpre = minp->next;
        free(minp);
    }
}

```