安徽大学 20 18 —20 19 学年第 2 学期

《 数据结构 》期中考试试卷

（闭卷　　时间 100 分钟）

考场登记表序号_____

题 号	一	二	三	四	五	六	七	总分
得 分								
阅卷人								

一、算法阅读题（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 x = 91;
    int y = 100;
    while(y > 0){
        if (x > 100){
            x = x - 10;
            y -= 2;
        }else{
            x++;
            n++;
        }
    }
}
```

答：由于算法复杂度并不受到 n 值大小的影响，因而复杂度为 O(1)。(5 分)

（2）

```
void test2(int n){
    int num1=0,num2=0;
    for(int i=0;i<n;i++){
        num1 += 1;
        for(int j=1;j<=n;j*=2){
            num2 += num1;
        }
    }
```

```
        }
```

答：外部 for 循环复杂度为 O(n)，内部 for 循环复杂度为 $O(\log_2 n)$，故整体复杂度为 $O(n\log_2 n)$。（5 分）

2. 以下算法用于处理带头结点的单链表 L，请仔细阅读下列算法，分析其完成了什么功能。

（1）

```
void test3(LinkList &L)
{
    LinkNode *p=L->next;
    LinkNode *r;
    r=(LinkNode *)malloc(sizeof(LinkNode));
    L->next=NULL;
    while(p!=NULL)
    {
        r=p->next;
        p->next=L->next;
        L->next=p;
        p=r;
    }
}
```

答：算法实现了单链表中所有节点的逆序排列，即链表的逆置。（5 分）

（2）

```
LinkList test4(LinkList L)
{
   LinkNode* p, p1, q;
   int m0, m1;
   p = L->next;
   p1 = p;
   if(p1 == NULL)
      return  p1;
   q = p->next;
   if(q == NULL)
      return  p1;
   m0 = p->data + q->data;
   while (q->next  !=  NULL)
   {
      p = q;
      q = q->next;
      m1 = p->data + q->data;
      if(m1 > m0)
      {
```

```
            p1 = p;
            m0 = m1;
        }
    }
    return p1;
}
```

答：算法寻找链表中相邻两个节点数值和最大的结点对，并返回这对结点中的第一个结点的指针。（5 分）

3. 已知一个顺序表中的元素存在重复值，下列算法删除顺序表中关键字的重复值，使得关键字相同的元素在表中只保留一个。

```
void purge_sq(SqList &la)
{
    int k = -1;
    for(int i=0;i<la.length;++i)
    {
        int j=0;
        while(j<=k && la.elem[j]!=la.elem[i])
            ++j;
//注释 1：_查找下一个元素是否与顺序表中已有元素的值重复，直到找到重复元素 j 或跳出循环__
        if(k==-1 || j>k)
            la.elem[++k] = la.elem[i];
//注释 2：_若没有找到重复元素，则将下一个元素存入新表中___
    }
    la.length=k+1;
}
```

(1) 请在代码中的空格处补充注释，分别说明 while 语句和 if 语句的功能。（6 分，每空 3 分）

(2) 如果已知某类顺序表中的元素按照关键字值非递减有序，原来的代码应该如何修改使得其效率更高？请在下面的代码中补充完整。

```
void purge_sq(SqList &la)
{
    int k = -1;
    for(int i=0;i<la.length;++i)
    {
        if(la.elem[i]!=la.elem[k])
            la.elem[++k] = la.elem[i]; （4 分）
    }
    la.length=k+1;
}
```

二、简答题（每小题 10 分，共 30 分）

得分	

4. 写出模式串“abcaabbcab”的 next 函数值和 nextval 函数值。

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

5. 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：

#8-3*（4-2）#

OPTR	OPND	INPUT	OPERATION
#		8-3*（4-2）#	Push(OPND,’ 8’)
#	8	-3*（4-2）#	Push（OPTR, ‘-’）
#, -	8	3*（4-2）#	Push（OPND, ‘3’）
#, -	8, 3	*（4-2）#	Push（OPTR, ‘*’）
#, -, *	8, 3	（4-2）#	Push（OPTR, ‘(’）
#, -, *, (	8, 3	4-2）#	Push（OPND, ‘4’）
#, -, *, (	8, 3, 4	-2）#	Push（OPTR, ‘-’）
#, -, *, (, -	8, 3, 4	2）#	Push（OPND, ‘2’）
#, -, *, (, -	8, 3, 4, 2	）#	Operate（‘4-2’）
#, -, *, (	8, 3, 2	）#	Pop（OPTR）
#, -, *	8, 3, 2	#	Operate（‘3*2’）
#, -	8, 6	#	Operate（‘8-6’）
#	2	#	GetTop（OPND）

6. 二维数组 A[10][10]按照行优先的顺序存储在起始地址为 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？元素 A[5][8]存储的起始位置是多少？

（2） 如果该数组是对称矩阵，可以采用压缩存储的方法节省存储空间，如果压缩时仅存储下三角，一共需要多少个单位的存储空间？元素 A[5][8]将从哪个起始位置开始存储？

答：（1） 4*10*10=400，一共需要 400 个存储空间。（2 分）
起始位置：1000+4*（5*10+8）=1000+4*58=1232。（3 分）
（2）4*（1+2+3+…+10）=220，一共需要 220 个存储空间。（2 分）
起始位置：A[5][8]存储在 A[8][5]的位置，即 1000+4*（1+2+3+4+5+6+7+8+5）=1000+4*41=1164。（3 分）

三、算法设计题（每小题 10 分，共 40 分）

得 分	

7. 将一个带头结点的单链表 head 拆分成两个带头结点的单链表 head1 和 head2。单链表 head 中每个结点存放一个整数，拆分后使得 head1 中结点值仅含有正整数或 0，head2 中仅含有负整数。

```
typedef struct LNode{
    float data;
    LNode *next;
}LNode,*LinkList;

void separate(LinkList &head, LinkList &head1, LinkList &head2)
{
   head1 = (LinkList)malloc(sizeof(LNode));
    head1->next = NULL;
    head2 = (LinkList)malloc(sizeof(LNode));
    head2->next = NULL;
    LNode* p = head->next;
    while(p)
    {
        LNode *q = p->next;
        if(p->data>=0)
        {
            p->next = head1->next;
            head1->next = p;
        }
        else
        {
            p->next = head2->next;
            head2->next = p;
        }
        p = q;
    }
    free(head);
```

```
}
```

8. 采用一个不带头节点只有一个尾节点指针 rear 的循环单链表存储队列，相关的类型定义如下：

```
typedef  struct node{
    ElemType  data;
    struct node *next;
}Node,*NPtr;//结点类型

typedef  struct{
  NPtr rear;
}Que;//队列类型
```

（1）写出队列 Q 为空的判定条件。（2 分）

答：Q.rear==NULL

（2）分别写出入队和出队算法。（8 分）

```
void enque(Que  &Q, ElemType  e)
{  p=(NPtr)malloc(sizeof(Node));
   p->data=e;
   if(Q.rear==NULL)
   {
      Q.rear=p;
      Q.rear->next=Q.rear;
    }//2 分
   else{
      p->next=Q.rear->next;
      Q.rear->next=p;
      Q.rear=p;
    } //  2 分
  return;
}

 Status  Deque(Que &Q,  ElemType &e)
{
   if(Q.rear==NULL) return ERROR;
   if(Q.rear->next==Q.rear)
    {  p=Q.rear;
       Q.rear=NULL;
    }  //   2 分
    else{
    p=Q.rear->next;
```

```
    Q.rear->next=p->next;
   }  //  2分
  e=p->data;
  free(p);
  return OK;
}
```

9. 在一个带头结点的单链表中查找元素值等于 x 的结点，若找到则将其从链表中删除，如果有重复的元素值 x 也都全部删除，如果找不到则输出“未找到这样的元素”。

```
void elemdelete(LinkList &L, float x)
{
    LNode *n = 0;
    LNode *pre = L;//记录结点 p 的前驱结点
    LNode *p = L->next;
    while(p)
    {
        while(p&&p->data!=x)
        {
            pre = p;
            p = p->next;
        }
        if(p)//找到 x 值，删除该结点
        {
            pre->next = p->next;
            LNode *q = p;
            p = p->next;
            free(q);
            n++;
        }
    }
    if(n==0)
        printf("未找到这样的元素\n");
}
```

10. 将两个递增的有序链表合并为一个递增的有序链表。要求结果链表仍使用原来两个链表的存储空间，不另外占用其它的存储空间。表中不允许有重复的数据。

```
void MergeList(LinkList &La,LinkList &Lb,LinkList &Lc)
{
    pa=La->next;
    pb=Lb->next;
```

```
    Lc=pc=La;
    while(pa && pb)
    {
      if(pa->data<pb->data)
      {
        pc->next=pa;
        pc=pa;
        pa=pa->next;
       }
       else if(pa->data>pb->data)
      {
        pc->next=pb;
        pc=pb;
        pb=pb->next;
       }
      else
      {
        pc->next=pa;
        pc=pa;
        pa=pa->next;
       q=pb->next;
        delete pb;
        pb =q;
      }
   }
    pc->next=pa?pa:pb;
   delete Lb;
}
```