软工23C1《数据结构与算法》- 绪论和线性表

开始时间 2024/03/27 12:47:00 结束时间 2024/07/10 14:47:00 答题时长 151320分钟

答卷类型 标准答案 总分 220

判断题

得分：暂无 总分：20

1-1 在具有N个结点的单链表中，访问结点和增加结点的时间复杂度分别对应为$O(1)$和$O(N)$。 (1分)

○ T ◉ F

1-2 线性表L如果需要频繁地进行不同下标元素的插入、删除操作，此时选择顺序存储结构更好。 (1分)

○ T ◉ F

1-3 若用链表来表示一个线性表，则表中元素的地址一定是连续的。 (1分)

○ T ◉ F

1-4 将长度分别为m,n的两个单链表合并为一个单链表的时间复杂度为O(m+n)。 (1分)

○ T ◉ F

1-5 在单链表中，要访问某个结点，只要知道该结点的指针即可。因此，单链表是一种随机存取结构。 (1分)

○ T ◉ F

1-6 链表是采用链式存储结构的线性表,进行插入、删除操作时，在链表中比在顺序存储结构中效率高。 (1分)

◉ T ○ F

1-7 算法分析的两个主要方面是时间复杂度和空间复杂度的分析。 (1分)

◉ T ○ F

1-8 对于某些算法，随着问题规模的扩大，所花的时间不一定单调增加。 (1分)

◉ T ○ F

1-9 线性表的插入、删除总是伴随着大量数据的移动。 (1分)

○ T ◉ F

1-10 算法可以没有输入，但是必须有输出。 (1分)

◉ T ○ F

1-11 带头结点的单循环链表中，任一结点的后继结点的指针域均不空。 (1分)

◉ T ○ F

1-12 循环链表可以做到从任一结点出发，访问到链表的全部结点。 (1分)

T F

1-13 在单链表中，逻辑上相邻的元素，其物理位置必定相邻。 (1分)

T F

1-14 在双向链表中，可以从当前结点出发访问到任何一个结点。 (1分)

T F

1-15 线性表采用链式存储结构时，各个数据元素的存储单元地址一定是不连续的。 (1分)

T F

1-16 对单链表来说，只有从头结点开始才能访问到表中所有结点。 (1分)

T F

1-17 抽象数据类型中基本操作的定义与具体实现有关。 (1分)

T F

1-18 算法最坏情况下的时间复杂度是指算法求解输入规模为n的问题实例所需的最多基本运算次数。 (1分)

T F

1-19 时间复杂度是根据算法写成的程序在执行时耗费时间的长度，往往与输入数据的规模有关。 (1分)

T F

1-20 空间复杂度是根据算法写成的程序在执行时占用存储单元的长度，往往与输入数据的规模有关。 (1分)

T F

单选题

得分：暂无 总分：70

2-1 线性表若采用链式存储结构时，要求内存中可用存储单元的地址 (2分)

A. 必须是连续的

B. 连续或不连续都可以

C. 部分地址必须是连续的

D. 一定是不连续的

2-2 在具有N个结点的单链表中，实现下列哪个操作，其算法的时间复杂度是$O(N)$? (2分)

A. 在地址为p的结点之后插入一个结点

B. 删除开始结点

C. 遍历链表和求链表的第i个结点

D. 删除地址为p的结点的后继结点

2-3 线性表L在什么情况下适用于使用链式结构实现? (2分)

A. 需不断对L进行删除插入

B. 需经常修改L中的结点值

C. L中含有大量的结点

D. L中结点结构复杂

2-4 链表不具有的特点是: (2分)

A. 插入、删除不需要移动元素

B. 方便随机访问任一元素

C. 不必事先估计存储空间

D. 所需空间与线性长度成正比

2-5 在单链表中，要删除某一指定结点，必须先找到该结点的（）。 (2分)

A. 直接前驱

B. 自身位置

C. 直接后继

D. 直接后继的后继

2-6 以下关于链式存储结构的叙述中，（）是不正确的。 (2分)

A. 结点除自身信息外还包括指针域，因此存储密度小于顺序存储结构

B. 逻辑上相邻的结点物理上不必邻接

C. 可以通过计算直接确定第i个结点的存储地址

D. 插入、删除运算操作方便，不必移动结点

2-7 对于一个具有N个结点的单链表，在给定值为x的结点后插入一个新结点的时间复杂度为 (2分)

A. $O(1)$

B. $O(N/2)$

C. $O(N)$

D. $O(N^2)$

2-8 链表 - 存储密度 (2分)

链表的存储密度 _____ 。

A. 大于 1 B. 等于 1 C. 小于 1 D. 不能确定

2-9 在数据结构中，从逻辑上可以把数据结构分成（）。 (2分)

A. 动态结构和静态结构

B. 紧凑结构和非紧凑结构

C. 线性结构和非线性结构

D. 内部结构和外部结构

2-10 以下说法正确的是（ ）。 (2分)

A. 数据元素是数据的最小单位

B. 数据项是数据的基本单位

C. 数据结构是带有结构的各数据项的集合

D. 一些表面上很不相同的数据可以有相同的逻辑结构

2-11 与数据元素本身的形式、内容、相对位置、个数无关的是数据的（ ）。 (2分)

A. 存储结构

B. 存储实现

C. 逻辑结构

D. 运算实现

2-12 以下数据结构中，（ ）是非线性数据结构。 (2分)

A. 树

B. 字符串

C. 队列

D. 栈

2-13 假设某个带头结点的单链表的头指针为head，则判定该表为空表的条件是（ ） (2分)

来源：

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. head==None

B. head.next==None

C. head!=None

D. head.next==head

2-14 单链表的结点指针域为next，其头结点由指针head指向，则删除第一个数据结点（由指针p指向）的语句为（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. p.next = head.next

B. head.next = p

C. p = head.next

D. head.next = p.next

2-15 单链表的指针域为next，其头结点由指针head指向，则把指针p指向的结点链接到头结点之后的语句序列为（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. p.next=head.next; head.next=p

B. head.next=p; p.next=head.next;

C. head.next = p.next; p= head.next

D. p.next=head; head=p

2-16 不带头结点的单链表（头指针为head）为空的判定条件是（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. head==None

B. head.next==None

C. head.next=head

D. head!=None

2-17 带头结点的循环单链表（头指针为head）为空的判定条件是（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. head==None

B. head.next==None

C. head.next==head

D. head!=None

2-18 在单链表中，指针域为next，要将q所指结点链接到p所指结点之后，其语句序列应为（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. q.next=p.next; p.next=q

B. p.next=q; q.next=p.next

C. q.next=p+1; p.next=q

D. p.next=q; q.next=p

2-19 在双向链表中，前驱指针为prior，后继指针为next，在p指针所指的结点后插入q所指的新结点，其语句序列是（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. q.prior=p; q.next=p.next; p.next=q; p.next.prior=q

B. q.prior=p; q.next=p.next; p.next.prior=q; p.next=q

C. p.next=q; p.next.prior=q; q.prior=p; q.next=p.next

D. p.next=q; q.prior=p; p.next.prior=q; q.next=q

2-20 在双向链表中，前驱指针为prior，后继指针为next，删除p所指的结点的语句序列为（ ） (2分)

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

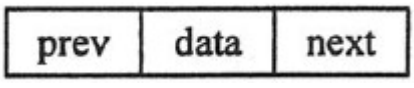
A. p.prior.next=p; p.prior=p.prior.prior

B. p.prior=p.next.next; p.next=p.prior.prior

C. p.next.prior=p.prior; p.prior.next=p.next

D. p.next=p.next.next; p.next.prior=p

2-21 现有非空双向链表 L，其结点结构为： (2分)



，`prev` 是指向直接前驱结点的指针，`next` 是指向直接后继结点的指针。若要在 L

中指针 `p` 所指向的结点（非尾结点）之后插入指针 `s` 指向的新结点，则在执行了语句序列 `s->next = p->next; p->next = s;` 后，下列语句序列中还需要执行的是：

A. `s->next->prev = p; s->prev = p;`

B. `p->next->prev = s; s->prev = p;`

C. `s->prev = s->next->prev; s->next->prev = s;`

D. `p->next->prev = s->prev; s->next->prev = p;`

2-22 已知带头结点的非空单链表 L 的头指针为 `h`，结点结构为 `data | next`，其中 `next` 是指向直接后继结点的指针。现有指针 `p` 和 `q`，若 `p` 指向 L 中非首且非尾的任意一个结点，则执行语句序列 `q = p->next; p->next = q->next; q->next = h->next; h->next = q;` 的结果是 (2分)

A. 在 `p` 所指结点后插入 `q` 所指结点

B. 在 `q` 所指结点后插入 `p` 所指结点

C. 将 `p` 所指结点移动到 L 的头结点之后

D. 将 `q` 所指结点移动到 L 的头结点之后

2-23 线性表L=(a_1， a_2， ...， a_n)，下列说法正确的是（ ） (2分)

来源：

黄龙军,等. 数据结构与算法, 上海:上海交通大学出版社, 2022.7. ISBN: 9787313269881

A. 每个元素都有一个直接前驱和一个直接后继

B. 表中至少有一个元素

C. 表中元素需有序

D. 除第一个和最后一个元素外，其他元素都有且仅有一个直接前驱和一个直接后继

2-24 线性表(a_1， a_2， ...， a_n)以顺序存储结构存储时，访问第i位置元素的时间复杂度为（ ） (2分)

出处：

黄龙军,等. 数据结构与算法, 上海:上海交通大学出版社, 2022.7. ISBN: 9787313269881

A. O(1)

B. O(n)

C. O(i)

D. O(i-1)

2-25 顺序表中第1个元素的存储地址是2000，每个元素的长度为4，则第5个元素的地址是（ ） (2分)

A. 2020

B. 2016

C. 2024

D. 2012

2-26 下列对顺序存储的有序表（长度为 n）实现给定操作的算法中，平均时间复杂度为 $O(1)$ 的是: (2分)

A. 查找包含指定值元素的算法

B. 插入包含指定值元素的算法

C. 删除第 i（$1 \leq i \leq n$）个元素的算法

D. 获取第 i（$1 \leq i \leq n$）个元素的算法

2-27 下述程序段的时间复杂度为（） (2分)

```
i=1
while i<=n:
    i=i*2
```

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. O($\log_2 n$)

B. O(n)

C. O($\sqrt{n}$)

D. O(n^2)

2-28 下述程序段的时间复杂度为（） (2分)

```
for i in range(m):
    for j in range(n):
        a[i][j]=0
```

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. O(n)

B. O(mn)

C. O(m^2)

D. O(n^2)

2-29 下述程序段的时间复杂度为（） (2分)

```
m,n=100,200
while n>0:
    c+=1
```

```
    if m>100: m-=10; n-=1
    else: m+=1
```

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. O(mn)

B. O(n)

C. O(m)

D. O(1)

2-30 下述程序段的时间复杂度为（ ） (2分)

```
a,b=n,0
while a>=b*b: b+=1
```

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. $O(\log_2 n)$

B. O(n)

C. $O(\sqrt{n})$

D. $O(n^2)$

2-31 下述程序段的时间复杂度为（ ） (2分)

```
for in range(n-1):
    for j in range(n-1-i):
        a[j], a[j+1]=a[j+1], a[j]
```

来源:

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. O(1)

B. O(n)

C. $O(n^2)$

D. $O(n^3)$

2-32 以下Python程序段的空间复杂度为（ ） (2分)

```
for in range(n-1):
    for j in range(n-1-i):
        a[j], a[j+1]=a[j+1], a[j]
```

来源：

黄龙军, 等. 数据结构与算法（Python版）,上海: 上海交通大学出版社, 2023. ISBN: 9787313280732

A. O(1)

B. O(n)

C. $O(n^2)$

D. $O(n^3)$

2-33 下列关于数据的逻辑结构的叙述中，（）是正确的。 (2分)

A. 数据的逻辑结构是数据元素间关系的描述

B. 数据的逻辑结构反映了数据在计算机中的存储方式

C. 数据的逻辑结构分为顺序结构和链式结构

D. 数据的逻辑结构分为静态结构和动态结构

2-34 算法的时间复杂度与（）有关。 (2分)

A. 问题规模

B. 计算机硬件的运行速度

C. 源程序的长度

D. 编译后执行程序的质量

2-35 数据结构在计算机内存中的表示是指（）。 (2分)

A. 数据的存储结构

B. 数据结构

C. 数据的逻辑结构

D. 数据元素之间的关系

函数题

得分：暂无 总分：70

6-1 顺序表的删除操作（Python语言描述） (10分)

本题要求实现一个顺序表SeqList的方法，删除顺序表SeqList第i个位置（从1开始）的元素。删除成功返回True，否则返回False。

假设线性表长度为n，则删除位置i须满足$1 \leq i \leq n$

方法接口定义：

```
#删除顺序表第i个位置的元素，删除成功返回True，否则返回False
def erase(self,i):
```

其中顺序表SeqList的类定义如下:

```
class SqList:
    #0.构造方法
    def __init__(self):
        self.initcapcity = 10            #初始容量
        self.capcity = self.initcapcity  #最大存储
        self.data = [None] * self.capcity#顺序表的数据，列表
        self.size = 0                    #顺序表的长度
    #0.顺序表的最大容量修改为n
    def __resize(self,n):
        assert n >= 0
        #备份原来的数据
        a = self.data
        self.data = [None] * n
        for i in range(self.size):
            self.data[i] = a[i]
    #1.创建顺序表,数据源是列表a
    def create(self,a):
        for i in range(len(a)):
            if self.size == self.capcity:#顺序表满了，2倍扩容
                self.__resize(self.capcity * 2)
            self.data[i] = a[i]
            self.size += 1
    #2.输出顺序表
    def print(self):
        print("the Length of SqList:",self.size)
        print("the Elements of SqList:",*self.data[:self.size])

    #3.你的代码将被嵌在这里，注意整个方法的代码都要缩进4个空格（类里面的方法）

```

裁判测试程序样例:

```
sq = SqList()   #创建顺序表
a = list(map(int,input().split())) #输入数据到列表a
i = int(input())    #输入删除位置
sq.create(a)        #根据列表a整体创建顺序表sq
if sq.erase(i):     #删除位置i的元素成功
    print("Delete Success")
    sq.print()
else:               #删除失败
    print("Delete Fail,Index is Error!")
```

输入样例1:

输入共有2行，第1行表示顺序表的元素，第2行表示删除的位置

```
2 6 4
1
```

输出样例1:

删除成功，按样例格式输出。

```
Delete Success
the Length of SqList: 2
the Elements of SqList: 6 4
```

输入样例2:

输入共有2行，第1行表示顺序表的元素，第2行表示删除的位置

```
2 6 4
0
```

输出样例2:

删除失败，按样例格式输出。

```
Delete Fail,Index is Error!
```

6-2 顺序表的插入操作（Python语言描述） （10分）

本题要求实现一个顺序表SeqList的方法，在顺序表SeqList第i个位置（从1开始）前插入元素x。插入成功返回True，否则返回False。

假设线性表长度为n，则删除位置i须满足$1 \leq i \leq n+1$

方法接口定义:

```
#在顺序表SeqList第i个位置（从1开始）前插入元素x。插入成功返回True，否则返回False。
def erase(self,i,x):
```

其中顺序表SeqList的类定义如下:

```
class SqList:
    #0.构造方法
    def __init__(self):
        self.initcapcity = 10            #初始容量
        self.capcity = self.initcapcity  #最大存储
        self.data = [None] * self.capcity#顺序表的数据，列表
        self.size = 0                    #顺序表的长度
    #0.顺序表的最大容量修改为n
    def __resize(self,n):
        assert n >= 0
        #备份原来的数据
        a = self.data
        self.data = [None] * n
        for i in range(self.size):
            self.data[i] = a[i]
    #1.创建顺序表,数据源是列表a
    def create(self,a):
        for i in range(len(a)):
            if self.size == self.capcity:#顺序表满了，2倍扩容
                self.__resize(self.capcity * 2)
            self.data[i] = a[i]
            self.size += 1
    #2.输出顺序表
    def print(self):
        print("the Length of SqList:",self.size)
        print("the Elements of SqList:",*self.data[:self.size])

```

```
    #3.你的代码将被嵌在这里，注意整个方法的代码都要缩进4个空格（类里面的方法）

```

裁判测试程序样例:

```
sq = SqList()   #创建顺序表
a = list(map(int,input().split())) #输入数据到列表a
i,x = map(int,input().split())    #输入插入位置i和元素x
sq.create(a)        #根据列表a整体创建顺序表sq
if sq.insert(i,x):   #位置i前插入元素x成功
    print("Insert Success")
    sq.print()
else:               #插入失败
    print("Insert Fail,Index is Error!")
```

输入样例1:

输入共有2行，第1行表示顺序表的元素，第2行表示插入的位置和元素

```
2 6 4
1 3
```

输出样例1:

删除成功，按样例格式输出。

```
Insert Success
the Length of SqList: 4
the Elements of SqList: 3 2 6 4
```

输入样例2:

输入共有2行，第1行表示顺序表的元素，第2行表示删除的位置

```
2 6 4
0 8
```

输出样例2:

删除失败，按样例格式输出。

```
Insert Fail,Index is Error!
```

6-3 求链表的长度（Python语言描述） （10分）

本题要求实现一个链表类LinkList的方法，求链表（含有带头结点）的长度。

方法接口定义:

```
#求链表的长度，空表返回0
def length(self):
```

其中链表结点类LinkNode和链表类LinkList的定义如下:

```
class LinkNode:
    def __init__(self,x = None):
        self.data = x        #数据域
        self.next = None     #指针域初始化为空
#单链表类
class LinkList:
    #0.构造方法
    def __init__(self):
        self.head = LinkNode()  #创建一个带头结点
    #1.输出单链表
    def print(self):
        p = self.head.next  #p指向链表的第一个有效结点
        if p is None:
            print("空链表")
        else:
            while p.next is not None:
                print(p.data,"->",sep = "",end = "")
                p = p.next
            print(p.data)
    #2.创建单链表-尾插法
    def createTail(self,a):
        tail = self.head   #p指向链表的带头结点
        #遍历列表a
        for x in a:
            #根据x创建新结点
            p = LinkNode(x)
            #新结点p链接到链表的尾部
            tail.next = p
            tail = p
    #3.创建单链表-头插法
    def createHead(self,a):
        for x in a:
            p = LinkNode(x)
            p.next = self.head.next
            self.head.next = p
    #4.你的代码将被嵌在这里，注意整个方法的代码都要缩进4个空格（类里面的方法）
```

裁判测试程序样例:

```
h = LinkList()   #创建链表
a = list(map(int,input().split())) #输入数据到列表a
h.createTail(a)     #尾插法创建单链表
print(h.length())   #输出链表的长度
```

输入样例:

输入共有1行，数据之间用空格分隔

```
2 0 2 3 0 5 0 4
```

输出样例:

输出链表的长度

```
8
```

6-4 统计单链表中正整数的个数（Python） （10分）

本题要求在带头结点的单链表中，统计并返回结点值为正整数的个数。

如结点，8 9 -1 2 -3 0

则该函数应该返回3。

函数接口定义：

```
# 统计并返回结点值为正整数的个数
def count(self):
```

裁判测试程序样例：

```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.next = None

class LinkList:
    def __init__(self):
        self.head = Node()
        self.head.next = None

    def createByTail(self, a):
        tail = self.head
        for i in range(len(a)):
            p = Node(a[i])
            tail.next = p
            tail = p
        tail.next = None

    ## 你的答案将被填在这里，请注意函数前面有4个前导空格##

    def showLinkList(self):
        p = self.head.next
        while p is not None:
            print(p.data, end=" ")
            p = p.next
        print()


if __name__ == '__main__':
    l = LinkList()
    a = [eval(x) for x in input().split()]
    l.createByTail(a)
    num = l.count()
    print(num)


```

输入样例1：

在这里给出一组输入。例如：

```
9 5 1 2 3
```

输出样例1:

在这里给出相应的输出。例如:

```
5
```

输入样例2:

在这里给出一组输入。例如:

```
9 -5 0 -2 3
```

输出样例2:

在这里给出相应的输出。例如:

```
2
```

6-5 统计单链表中奇数的个数（Python） （10分）

本题要求在带头结点的单链表中，统计并返回结点值为奇数的个数。

如结点，8 9 -1 2 -3 0

则该函数应该返回3。

函数接口定义:

```python
# 统计并返回结点值为奇数的个数
def oddNum(self):
```

裁判测试程序样例:

```python
class Node:
    def __init__(self, data=None):
        self.data = data
        self.next = None

class LinkList:
    def __init__(self):
        self.head = Node()
        self.head.next = None

    def createByTail(self, a):
        tail = self.head
        for i in range(len(a)):
            p = Node(a[i])
            tail.next = p
            tail = p
        tail.next = None

    ## 你的答案将被填在这里，请注意函数前面有4个前导空格##

    def showLinkList(self):
```

```
        p = self.head.next
        while p is not None:
            print(p.data, end=" ")
            p = p.next
        print()


if __name__ == '__main__':
    l = LinkList()
    a = [eval(x) for x in input().split()]
    l.createByTail(a)
    num = l.count()
    print(num)


```

输入样例1:

在这里给出一组输入。例如:

```
9 -5 1 7 3
```

输出样例1:

在这里给出相应的输出。例如:

```
5
```

输入样例2:

在这里给出一组输入。例如:

```
9 4 0 -2 3
```

输出样例2:

在这里给出相应的输出。例如:

```
2
```

6-6 在单链表指定位置插入元素（Python）（10分）

在一个带头结点的单链表的指定位置i，增加一个新的元素 e ，若给定的位置不合法则提示插入失败。
例如：单链表 9 5 1 2 3，
指定位置3插入元素4，则 新的单链表为 9 5 4 1 2 3
指定位置7插入元素8，则 提示插入失败。

函数接口定义:

```
def addi(self, i, e):
```

其中i 和 e 都是用户传入的参数。 L 表示单链表，i表示指定位置，e是待插入元素。函数须返回插入元素后的结果，成功则返回True，不成功则返回False。

裁判测试程序样例:

```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.next = None

class LinkList:
    def __init__(self):
        self.head = Node()
        self.head.next = None

    def createByTail(self, a):
        tail = self.head
        for i in range(len(a)):
            p = Node(a[i])
            tail.next = p
            tail = p
        tail.next = None

    def showLinkList(self):
        p = self.head.next
        while p is not None:
            print(p.data, end=" ")
            p = p.next
        print()

    def findi(self, i):
        p = self.head.next
        j = 1
        while p is not None and j < i:
            j += 1
            p = p.next
        return p

    ## 你的答案将被填在这里，请注意函数前面有4个前导空格##

if __name__ == '__main__':
    l = LinkList()
    n = int(input())
    a = list(map(int,input().split()))
    l.createByTail(a)
    b = list(map(int,input().split()))
    if l.addi(b[0], b[1]) is True:
        l.showLinkList()
    else:
        print("insert fail")
```

输入样例:

```
5
9 5 1 2 3
3 4
```

输出样例:

```
9 5 4 1 2 3
```

6-7 在单链表指定位置删除元素（Python） （10分）

在一个带头结点的单链表中，删除指定位置 i 的元素 ，若给定的位置不合法则提示删除失败。
例如：单链表 9 5 1 2 3，
删除指定位置3，则 新的单链表为 9 5 1 2 3
删除指定位置7，则 提示删除失败。

函数接口定义：

```
def deletei(self, i):
```

其中 `i` 是用户传入的参数。函数须返回删除元素后的结果，成功则返回True，不成功则返回False。

裁判测试程序样例：

```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.next = None

class LinkList:
    def __init__(self):
        self.head = Node()
        self.head.next = None

    def createByTail(self, a):
        tail = self.head
        for i in range(len(a)):
            p = Node(a[i])
            tail.next = p
            tail = p
        tail.next = None

    def showLinkList(self):
        p = self.head.next
        while p is not None:
            print(p.data, end=" ")
            p = p.next
        print()

    def findi(self, i):
        p = self.head.next
        if i < 0:
            return None
        j = 1
        while p is not None and j < i:
            j += 1
            p = p.next
        return p
    ## 你的答案将被填在这里，请注意函数前面有4个前导空格##
if __name__ == '__main__':
    l = LinkList()
    n = int(input())
```

```
    a = list(map(int,input().split()))
    l.createByTail(a)
    i = int(input())
    if l.deletei(i) is True:
        l.showLinkList()
    else:
        print("delete fail")
```

输入样例:

```
5
9 5 1 2 3
3
```

输出样例:

```
9 5 2 3
```

编程题 得分：暂无 总分：60

7-1 有序顺序表的插入（10分）

请设计一个算法，在有序顺序表L中插入元素x，使得表依然有序，并输出新增元素后的表数据。

例如：
L的元素 1 3 5 7
插入新元素 4
输出 1 3 4 5 7
其中，L的长度不超过1000，当中的元素为非递减排序。

输入格式:

第一行输入L的长度
第二行输入L的元素
第三行输入要插入的元素x的值

输出格式:

输入插入元素后顺序表中各元素的值，值之间用一个空格间隔。

输入样例:

```
4
1 3 5 7
4
```

输出样例:

```
1 3 4 5 7
```

7-2 单链表的创建及遍历（10分）

读入n值及n个整数，建立单链表并遍历输出。

输入格式:

读入n及n个整数。

输出格式:

输出n个整数，以空格分隔（最后一个数的后面没有空格）。

输入样例:

在这里给出一组输入。例如:

```
2
10 5
```

输出样例:

在这里给出相应的输出。例如:

```
10 5
```

7-3 求链式线性表的倒数第K项（10分）

给定一系列正整数，请设计一个尽可能高效的算法，查找倒数第K个位置上的数字。

输入格式:

输入首先给出一个正整数K，随后是若干非负整数，最后以一个负整数表示结尾（该负数不算在序列内，不要处理）。

输出格式:

输出倒数第K个位置上的数据。如果这个位置不存在，输出错误信息`NULL`。

输入样例:

```
4 1 2 3 4 5 6 7 8 9 0 -1
```

输出样例:

```
7
```

7-4 单链表就地逆置（10分）

7-5 有序顺序表的合并（10分）

7-6 约瑟夫环（10分）