



## 一、字符串处理 (30 分)

编制一个程序，程序结构如下：

### 1. 函数 `input()` (5 分)

从键盘输入字符串 `s` (长度不超过 24)，并将 `s` 返回给主函数。

### 2. 函数 `triangle_matrix(char *)` (20 分)

接收主函数的实参 `s`，并将其输出为如下形式的上三角矩阵到屏幕。

不允许 `triangle_matrix()` 函数内部占用新的数组空间。

### 3. 主函数 `main()` (5 分)

当且仅当输入的字符串 `s` 为空时，主函数结束循环，退出程序。

### 4. 编程限制

编程时，限制使用 `string.h` 中的任何字符串操作函数。

程序运行示例 1：

```
请输入字符串s:
abcdefghijklmn
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, n
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, m
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, l
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, k
0, 1, 2, 3, 4, 5, 6, 7, 8, j
0, 1, 2, 3, 4, 5, 6, 7, i
0, 1, 2, 3, 4, 5, 6, h
0, 1, 2, 3, 4, 5, g
0, 1, 2, 3, 4, f
0, 1, 2, 3, e
0, 1, 2, d
0, 1, c
0, b
a
请输入字符串s:
程序结束!
```



## 二、数组处理 (30 分)

若有两个二维正整数数组  $A$ 、 $B$  分别为 2 行  $M$  列、2 行  $N$  列，如果它们的第 0 行的元素  $a_{0,i} \ i = 0, 1, 2, 3 \dots M$ ， $b_{0,j} \ j = 0, 1, 2, 3 \dots N$  分别是严格递增有序的，则可以定义两个数组的一种合并运算  $C = A \div B$  如下：

- (1)  $C$  中的第 0 行元素  $c_{0,i} (i = 0, 1, 2, 3 \dots)$  也是严格递增有序的, 并且由  $A$ 、 $B$  中所有的第 0 行元素构成; 若  $A$ 、 $B$  的第 0 行元素相等时, 则只保留其中一个。
- (2)  $C$  中的第 1 行元素  $c_{1,i} (i = 0, 1, 2, 3 \dots)$  由生成相应  $c_{0,i}$  元素时对应的  $A$ 、 $B$  的第 1 行元素组成; 若生成  $c_{0,i}$  时, 由  $A$ 、 $B$  中两个相同的元素  $a_{0,k}$ 、 $b_{0,l}$  保留其中一个生成, 则对应的该  $c_{1,i} = (a_{1,k} + b_{1,l}) / 2$  后取整。

如:  $A = \begin{pmatrix} 1 & 4 & 7 & 9 \\ 3 & 5 & 4 & 2 \end{pmatrix}$ ,  $B = \begin{pmatrix} 2 & 3 & 4 & 5 & 8 \\ 3 & 5 & 5 & 2 & 1 \end{pmatrix}$ , 则  $C = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 7 & 8 & 9 \\ 3 & 3 & 5 & 5 & 2 & 4 & 1 & 2 \end{pmatrix}$

编制一个程序,要求实现如下功能:

1. 函数 input() (4分)

清晰提示从键盘输入一个二维正整数数组（2行N列）各元素值，并能传回给主函数。

- 假设数组的最大列数  $N$  不超过 10000。
- 输入格式要求每次输入数组的一列元素，若遇到某个数对中表示 0 行元素的值为 0 时，数组输入结束。
- 并假设在输入过程中，已保证输入的第 0 行的各元素互不重复并且均为正整数，输入的第 1 行各元素也均为正整数，程序不用再判断检查。

2. 函数 sort() (8分)

函数 sort() (8分)。  
接收并完成对于一个 2 行  $N$  列二维数组的排序，排序后将二维数组变为按照第 0 行元素值递增有序，且保留原数组中各元素的对应关系。

- 如输入的数组为： $\begin{pmatrix} 4 & 2 & 8 & 3 & 5 \\ 5 & 3 & 1 & 5 & 2 \end{pmatrix}$ ，排序后的数组则为 $\begin{pmatrix} 2 & 3 & 4 & 5 & 8 \\ 3 & 5 & 5 & 2 & 1 \end{pmatrix}$ 。

3. 函数 merge () (8分)

**函数 merge () (8分)**  
按照前述二维数组合并计算的定义，计算得到一个新的二维数组，并能够将计算结果传回给主函数输出。

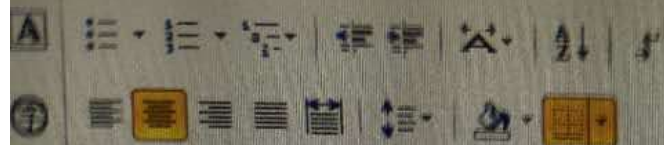
4. 函数 peak() (6 分)

函数 peak() (6分)  
在合并后的结果数组中,找到第 1 行元素值最大的所有峰值点,并传回给主函数输出。

5. 编写主函数 main() (4分)

编写主函数 main() (4分)。  
按照程序示例，主函数调用上述的函数，完成相应功能。





AaBb( AaB| AaBb( AaBbCcD( AaBbCcD

标题

标题 1

副标题

强调

要点

段落

样式

### 三、链表编程 (30 分)

某课程期末考试后,需要由助教对成绩进行录入处理,然后发给其中三个班的班主任查看。假设每位同学的成绩信息包括四项内容:学号(Num,长度为 10 个字节的字符串)、姓名(Name,长度为不超过 20 个字节的字符串)、班级(Class,整数类型)、课程成绩(Score,浮点数类型)。

助教在录入信息时是随机录入,当输入的班级号为 0 时,录入结束。

三位班主任只能查看本班同学的成绩信息,并且查看的习惯也不一样,1 班班主任需要按照成绩从高到低查看,2 班班主任需要按照学号从低到高查看,3 班班主任喜欢按照助教录入的顺序查看。

请按照以下要求,编写完成函数程序。

#### 1. 链表节点定义 (2 分)

请使用链表技术实现程序功能,合理正确的定义节点结构。

#### 2. 函数 creat() (8 分)

清晰提示助教从键盘输入同学的信息,并依照输入的顺序(先输入的在前,后输入的在后)生成一个链表 List0,返回给主函数。

- 当输入同学的信息中的班级信息为 0 时,表示该链表输入完成。
- 在输入过程中,要求能够判断助教是否输入了重复的学号,如果学号重复要求重新输入;其他内容在输入时已保证信息符合要求,程序不用再判断。

#### 3. 函数 division() (6 分)

接收主函数传递来的助教输入形成的链表 List0,并按照班级(Class)信息将助教输入的链表处理成为 3 个新的链表,将 3 个新的链表传回给主函数。

其中:

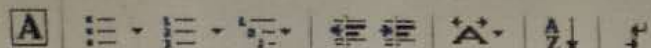
- 链表 List1 只包含班级信息为 1 的同学信息,并且符合成绩从高到低的顺序,
- 链表 List2 只包含班级信息为 2 的同学信息,并且符合学号从低到高的顺序,
- 链表 List3 只包含班级信息为 3 的同学信息,并且保持原始的输入时的顺序,
- 其他不属于上述 3 个班级的同学信息,仍保留在链表 List0 中。

#### 4. 函数 rank() (6 分)

接收主函数传递来的一个同学的学号,并在 1、2 或者 3 班同学链表 list1、list2 或者 list3 中查找该学号同学是否存在;如果不存在,函数返回名次为 0;如果存在,则返回该同学在该班中的排名,并传回所在班级号。

- 排名规则为:若本班中有 k 个人的成绩高于该同学成绩,则该同学在本班中排名





AaBb| AaB| AaBb| AaBbCcD| AaBbCcD



标题

标题 1

副标题

强调

要点

段落

样式

## 3. 函数 merge() (8分)

按照前述二维数组合并计算的定义，计算得到一个新的二维数组，并能够将计算结果传回给主函数输出。

## 4. 函数 peak() (6分)

在合并后的结果数组中，找到第 1 行元素值最大的所有峰值点，并传回给主函数输出。

## 5. 编写主函数 main() (4分)

按照程序示例，主函数调用上述的函数，完成相应功能。

(1) 调用 input() 函数，完成两个二维数组 A、B 的输入。

(2) 调用 sort() 函数，完成数组 A、B 的排序。

(3) 调用 merge() 函数，完成合并后新数组 C 的计算，并输出。

(4) 调用 peak() 函数，输出数组 C 所有峰值点。

程序运行示例 2：

请输入二维正整数数组（列数不超过10000）：

7 4  
4 5  
9 2  
1 3  
0 1

请输入二维正整数数组（列数不超过10000）：

4 5  
2 3  
8 1  
3 5  
5 2  
0 1

合并后的数组C为：

1 2 3 4 5 7 8 9  
3 3 5 5 2 4 1 2

数组C的峰值有2个，分别为：(3, 5) (4, 5)





AaBb| AaB| AaBb| AaBbCcD| AaBbCcD

标题

标题 1

副标题

强调

要点

段落

样式

- 链表 List1 只包含班级信息为 1 的同学信息，并且符合成绩从高到低的顺序，
- 链表 List2 只包含班级信息为 2 的同学信息，并且符合学号从低到高的顺序，
- 链表 List3 只包含班级信息为 3 的同学信息，并且保持原始的输入时的顺序，
- 其他不属于上述 3 个班级的同学信息，仍保留在链表 List0 中。

#### 4. 函数 rank() (6 分)

接收主函数传递来的一个同学的学号，并在 1、2 或者 3 班同学链表 list1、list2 或者 list3 中查找该学号同学是否存在；如果不存在，函数返回名次为 0；如果存在，则返回该同学在该班中的排名，并传回所在班级号。

- 排名规则为，若本班中有 k 个人的成绩高于该同学成绩，则该同学在本班中排名为第 k+1。

#### 5. 函数 list() (4 分)

在屏幕上打印链表，依次清晰输出链表各节点的信息。

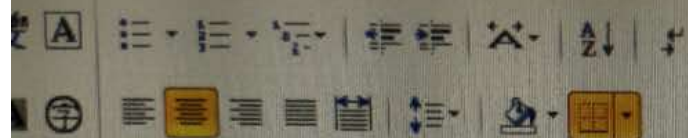
#### 6. 主函数 main() (4 分)

按照程序示例，主函数调用上述的函数，完成相应功能。

- (1) 调用 creat() 函数，生成链表 List0；
- (2) 调用 list() 函数，输出链表 List0；
- (3) 调用 division() 函数，生成链表 List1、List2、List3；
- (4) 调用 4 次 list() 函数，分别输出链表 List0、List1、List2、List3；
- (5) 清晰提示用户输入一个学号。
- (6) 调用函数 rank()，获得并输出该同学所在的班级，及在本班中的成绩排名。

程序运行示例 3：





#### 四、递归编程 (10 分)

请按照要求编写递归程序，实现矩阵的 $\otimes$ 运算。

矩阵的 $\otimes$ 运算定义如下：若矩阵  $A$ 、 $B$  均为  $N \times N$  的方阵，则可以定义矩阵  $C = A \otimes B$  中的矩阵  $C$  也为  $N \times N$  的方阵，并且  $C$  的元素为： $c_{ij} = \sum_{k=0}^{N-1} (-1)^k * a_{i,k} * b_{k,j}$

如  $2 \times 2$  的矩阵  $A = \begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{pmatrix}$  和  $B = \begin{pmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{pmatrix}$ ， $C = A \otimes B$

则  $C = \begin{pmatrix} c_{0,0} & c_{0,1} \\ c_{1,0} & c_{1,1} \end{pmatrix}$  中各元素的值为：

$$c_{0,0} = a_{0,0} * b_{0,0} - a_{0,1} * b_{1,0}$$

$$c_{0,1} = a_{0,0} * b_{0,1} - a_{0,1} * b_{1,1}$$

$$c_{1,0} = a_{1,0} * b_{0,0} - a_{1,1} * b_{1,0}$$

$$c_{1,1} = a_{1,0} * b_{0,1} - a_{1,1} * b_{1,1}$$

根据上述定义，请编写递归程序，实现两个  $N \times N$  的整数矩阵的 $\otimes$ 运算（其中  $N$  为 2 正整数指数幂，即  $N = 2^m$   $m = 1, 2, 3 \dots$ ）

编程要求如下：

##### 1. 函数 Input() (3 分)

清晰提示输入矩阵的阶数  $N$ ，根据用户输入的  $N$ ，动态创建矩阵  $A$ 、 $B$ ，清晰提示用户输入  $A$ 、 $B$  的各元素值。

假设用户输入的  $N$  满足 2 正整数指数幂的要求，不用判断。

##### 2. 函数 MyMultiply() (6 分)

编写递归函数 MyMultiply()，实现两个矩阵  $A$ 、 $B$  的 $\otimes$ 运算。

##### 3. 主函数 main() (1 分)

调用上述函数后，实现矩阵的 $\otimes$ 运算，并将计算结果在屏幕输出。





AaBb| AaB| AaBb| AaBbCcD| AaBbCcD| AaBbCcD|  
标题 标题 1 副标题 强调 要点 正文



段落

样式

根据上述定义，请编写递归程序，实现两个  $N \times N$  的整数矩阵的 $\otimes$ 运算（其中  $N$  为 2 正整数指数幂，即  $N=2^m$   $m=1,2,3 \dots$ ）

编程要求如下：

1. 函数 Input() (3 分)

清晰提示输入矩阵的阶数  $N$ ，根据用户输入的  $N$ ，动态创建矩阵  $A$ 、 $B$ ，清晰提示用户输入  $A$ 、 $B$  的各元素值。

假设用户输入的  $N$  满足 2 正整数指数幂的要求，不用判断。

2. 函数 MyMultiply() (6 分)

编写递归函数 MyMultiply()，实现两个矩阵  $A$ 、 $B$  的 $\otimes$ 运算。

3. 主函数 main() (1 分)

调用上述函数后，实现矩阵的 $\otimes$ 运算，并将计算结果在屏幕输出。

程序运行示例 4::

请输入矩阵的阶数N: 4

请输入矩阵A的元素:

1 2 3 4  
5 6 7 8  
9 10 11 12  
13 14 15 16

请输入矩阵B的元素:

9 10 11 12  
13 14 15 16  
1 2 3 4  
5 6 7 8

矩阵C的运算结果为:

-34 -36 -38 -40  
-66 -68 -70 -72  
-98 -100 -102 -104  
-130 -132 -134 -136

//以上为全部题目。