CSE 101
数据结构和算法介绍 编程作业1

我们在这个项目中的目标是在C语言中建立一个Integer List ADT，并使用它来间接地将文件中的行按字母顺序排列。这个ADT模块还将在未来的编程作业中使用（经过一些修改），所以你应该彻底测试它，尽管这里不会用到它的所有功能。首先阅读课堂网页上张贴的ADT.pdf讲义，了解在本课中用C语言实现ADT所需的编程实践和惯例。

程序操作
这个项目的主程序将被称为Lex.c。你的List ADT模块将包含在名为List.h和List.c的文件中，并将其服务输出给客户模块Lex.c。下面将详细说明所需的List操作。Lex.c将接受两个命令行参数，分别给出一个输入文件和一个输出文件的名称。

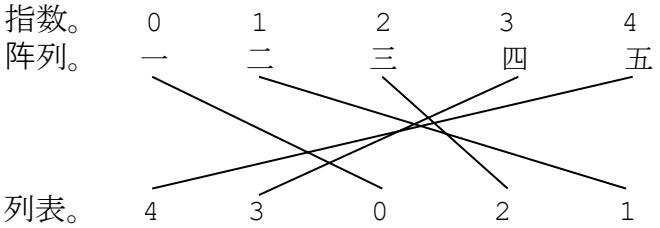
```
Lex <输入文件><输出文件><输出文件>。
```

输入可以是任何文本文件。输出文件将包含与输入相同的行，但按词法（即字母顺序）排列。比如说

输入文件。	输出文件。
一	五
二	四
三	一
四	三
五	二

Lex.c将遵循下面的草图。

1. 检查是否有两个命令行参数（除程序名称Lex外）。如果给出的命令行参数多于或少于两个，则退出，并向stderr发送一条使用信息。
2. 计算输入文件中的行数*n*。创建一个长度为*n*的字符串数组，将文件中的行作为字符串读入，并将其放入数组中。(使用头文件stdlib.h中定义的函数calloc()或malloc()从堆内存中分配这个数组。不要使用一个可变长度的数组。关于这个问题，请看这里的评论）。
3. 创建一个List，其元素是上述字符串数组的索引。这些索引应该按照间接对数组进行排序的顺序来排列。以上述输入文件为例，我们会有。



为了按照正确的顺序建立整数列表，从最初的空列表开始，然后将数组的索引一个一个地插入到列表的适当位置。使用插入排序算法（文本CLRS的第2.1节）来指导你思考如何完成这个任务。(请多读几遍前面的两句话，以便你理解所要求的内容。我们*并没有*要求你使用插入式排序对输

入数组进行排序）。你可以只使用下面定义的List ADT操作来操作

列表。请注意，C语言标准库string.h提供了一个名为`strcmp()`的
函数，可以确定两个字符串的lexicographic排序。如果`s1`和`s2`是字符串，那么。

`strcmp(s1, s2)<0` 当且仅当`s1`在`s2`之前时为真 `strcmp(s1, s2)>0` 当且仅当`s1`在`s2`之后时为真 `strcmp(s1, s2)==0`
当且仅当`s1`与`s2`相同时为真

4. 使用(3)中构建的List，按字母顺序将数组打印到输出文件。请注意，在任何时候，数组都不会被排序。相反，你是通过建立一个按一定顺序排列的索引列表来*间接地*对数组进行排序。

如果你还不熟悉C语言中的文件输入-
输出操作，请看例子FileIO.c来了解这些操作。我将在例子部分放置一些匹配的输入-
输出文件对，以及一个创建随机输入文件的python脚本，以及它们匹配的输出文件。你可以在你的程序启动和运行后使用这些工具来测试它。

列出ADT规格

本项目的列表模块将是一个双向队列，包括一个用于迭代的
"光标"。把光标看作是突出显示或下划线的列表中的一个区分元素。请注意，对于这个ADT来说，*没有*区分的元素是一个有效的状态，也就是说，光标可以是 "未定义 "或 "离开列表"，这实际上是其默认状态。因此，这个ADT的 "数学结构 "集合包括所有整数的有限序列，其中最多只有一个元素是下划线的。一个列表有两个端点，分别被称为 "前 "和 "后"。客户端将使用游标在任一方向上遍历列表。每个列表元素都有一个索引，范围从0（前）到n-1（后），其中n是列表的长度。你的列表模块将输出一个列表类型以及以下操作。

```
// 构建器-分解器                  ------------------------------------------------------------------------
List newList(void);          //创建并返回一个新的空List。
void freeList(List* pL); // 释放与*pL相关的所有堆内存，并设置
                          // *pL为NULL。

// 访问功能            ---------------------------------------------------------------------------------
int length(List L); // Returns the number of elements in L.
int index(List L); // 如果定义了游标元素，则返回游标元素的索引，否则返回-1。 int
front(List L); // 返回L的前面元素。 Pre: length()>0
int back(List L);    //返回L的背面元素。 Pre: length()>0
int get(List L);     //返回L的游标元素。前提是：length()>0, index()>=0 bool
equals(List A, List B); // 如果列表A和列表B在一起，返回true。
                           //状态，否则返回false。

// 操纵程序                        ------------------------------------------------------------------------
void clear(List L);        //将L重置为原来的空状态。
void set(List L, int x); // 用x覆盖游标元素的数据。
                         // Pre: length()>0, index()>=0
void moveFront(List L); // 如果L是非空的，将光标设置在前面的元素下面。
                        //否则什么都不做。
void moveBack(List L);    //如果L是非空的，将光标设置在后面的元素下面。
                        //否则什么都不做。
void movePrev(List L);    //如果光标被定义，并且不在前面，则将光标移到前面。
                        //向L的前面走去；如果光标被定义并且在
                        //前面，游标变成未定义的；如果游标未定义
                        //什么都不做
```

```
void moveNext(List L);     //如果光标被定义并且不在后面，则将光标移动一个
                           //向L的后面走去；如果光标被定义并且在
                           //返回，游标变成未定义的；如果游标未定义
                           //什么都不做
```

```
void prepend(List L, int x); // 在L中插入新元素，如果L是非空的。
                             // 如果L是非空的，就在前面的元素之前插入。 void
append(List L, int x); // 在L中插入新元素。
                             //插入发生在后面的元素之后。 void insertBefore(List
L, int x); //在光标之前插入新元素。
                                    // Pre: length()>0, index()>=0
void insertAfter(List L, int x);    //在光标后插入新元素。
                                    // Pre: length()>0, index()>=0
void deleteFront(List L); // 删除前面的元素。Pre:length()>0 void
deleteBack(List L);         // Delete the back element.预设：length()>0
void delete(List L);        //删除游标元素，使游标无法定义。
                            // Pre: length()>0, index()>=0

// 其他操作          ----------------------------------------------------------------------------
void printList(FILE* out, List L); // 打印到out所指向的文件，a
                                    // L的字符串表示法，包括
                                    //一个空格分隔的整数序列的。
                                    //与左前方的人在一起。
List copyList(List L); // Returns a new List representing same integer.
                       //序列为L，新列表中的光标是未定义的。
                       //不管光标在L中的状态如何。
                       L的//没有变化。
```

上述操作是获得满分的必要条件，尽管我们并不期望客户模块在本项目中全部使用。下面的操作是可选的，可能会在未来的一些任务中派上用场。

```
List concatList(List A, List B); // Returns a new List which is concatenation of
                                 // A和B，新的列表中的光标是未定义的。
                                 //不管A和B中光标的状态如何。
                                 // A和B的状态没有变化。
```

请注意，上述操作为客户端提供了一种标准的方法，可以在List中的元素上进行双向迭代。客户端中的一个典型的循环可能如下所示。

```
moveFront(L)。
while( index(L)>=0 ){
   x = get(L)。
   //对x做一些处理 moveNext(L);
}
```

要从后往前迭代，用moveBack()替换moveFront()，用movePrev()替换moveNext()。我们也可以把它设置成一个for循环。请注意，在L为空的特殊情况下，游标必然是未定义的，所以index(L)返回-1，使得循环的重复条件最初为假。因此这个循环执行了零次，就像它在一个空的List上应该执行的那样。我们要求函数index()能够有效地实现，也就是说，它本身不应该包含一个循环。

List ADT的底层数据结构将是一个双链表。因此，文件List.c应该包含一个名为NodeObj的私有（非输出）结构和一个名为Node的指向该结构的指针。NodeObj 结构应该包含一个 int 字段（数据）和两个 Node 引用（分别是上一个和下一个 Node）。你还应该为私有 Node 类型包含一个构造函数和析构函数。私有（非输出）结构ListObj应该包含Node类型的字段，分别引用

前面、后面和游标元素。ListObj还应该包含Int字段，用于表示List的长度，以及游标元素的索引。当游标未被定义时，索引字段的合适值是-
1，因为在这种情况下函数`index()`会返回这个值。学习课程网页上的例子Queue.c和Stack.c，可以随意使用这两个文件作为List.c的起点。

一个名为ListClient.c的测试客户端样本将被放在网页的Examples/pa1中，你将不会提交该样本。这个程序应该被认为是对你的List ADT的一个弱测试。它的正确输出被作为注释包含在文件的末尾。为List ADT创建你自己的测试客户端，名为ListTest.c，并与该项目一起提交。它应该包含你自己对所有ADT操作的测试。

你需要提交一个Makefile，创建一个名为Lex的可执行二进制文件，这是本项目的主要程序。在你的Makefile中包括一个 清 除 目标，删除Lex和任何相关的.o文件，以帮助评分员清理提交目录。一个可能的Makefile将包括在课程网页的
`Examples/pa1`下。你可以根据自己的需要修改这个Makefile，以执行其他任务，如提交。请参阅我的CMPS
12B的实验作业1（现已停用）https://classes.soe.ucsc.edu/cmps012b/Spring19/lab1.pdf，以了解关于Makefile的基本信息。

请注意，上述Makefile中提到的编译操作是调用带有标志-
`std=c17`的gcc编译器。这个项目（以及所有其他的C程序）的一个要求是，它在gcc下的编译没有警告或错误（带有c17标志），并且在ITS提供的UNIX Timeshare unix.ucsc.edu上的Linux计算环境下正常运行。你的C语言程序也必须在没有内存泄漏的情况下运行。使用unix.ucsc.edu上的`valgrind`测试它们，方法是

```
valgrind program_name argument_list。
```

你还必须为这个（以及每个）作业提交一个README文件。README将列出每一个提交的文件，以及对其在项目中的作用的简要描述，还有对我和评分员的任何特别说明。README基本上是项目的内容表，仅此而已。因此，你将提交总共六个文件。

你所写的List.	h
列表.	c由你来写
ListTest.	c由你来写
Lex.	cwritten by you
网页上提供的	Makefile，根据需要修改。
由你撰写的	README

如果你拼错了这些文件名，或者你提交了.o文件、可执行的二进制文件、输入-
输出文件或任何其他上面没有说明的文件，都会被扣分。你提交的每个源文件必须以一个注释块开始，包含你的名字、CruzID和作业名称（本例中为pa1）。

建议

网站上的例子Queue.c和Stack.c是本项目中List
ADT模块的良好起点。我们欢迎你从这些文件中的一个开始，重命名，然后增加功能，直到满足List ADT的规格。你应该首先设计和建立你的List
ADT，对它进行彻底的测试，然后才开始对Lex.c进行编码。关于如何交出你的项目的信息已张贴在班级网页上。