实验室 3 活动 [30 分]

自动评分评估

对于提交自动评分的实验室部分，请勿使用printf()**或者**扫描函数（）。

重要的： **您提交的内容必须经过编译且没有语法错误才能获得成绩。不可编译的解决方案将不会被评分。**

重要的： **不良的内存管理实践将受到处罚。请参阅标记方案以了解需要注意的细节。**

关于实验室 3

- 使用以下链接接受实验室邀请：https://classroom.github.com/a/UgAAsy80 实验 3 的截
- 止日期为**2023 年 11 月 10 日星期五晚上 11:59**

- 您可以使用 <stdio.h> 和 <stdlib.h> 中的函数，它们都包含在框架代码中。
- **不允许使用字符串库和其他 C 库。不允许使用全局变量。**
  包括它们将导致该问题的结果为零。

- **内存分析器工具将用于检查源自实现内部的任何内存泄漏。**

  如果内存管理实践不佳，将会被扣分。

关于内存泄漏评估的重要注意事项

鉴于内存泄漏可能发生在已安装的工具链和第三方库中，因此您不需要消除系统上内存分析器工具捕获的所有内存泄漏。但是，您需要**分析报告并确保没有任何泄漏源自代码实现中的 malloc() 调用** 。

回顾：如何执行内存使用情况分析

视窗

在 VSCode 终端中，在包含编译后的 Lab3.exe 的 Lab 3 文件夹下，输入：

博士记忆./Lab3。

将生成内存泄漏报告。

苹果系统

在终端中，使用已编译的 Lab3 可执行文件导航到 Lab 3 文件夹下，执行以下操作：

1. 输入：导出 MallocStackLogging=1
2. 输入：泄漏 --atExit --list -- ./Lab3

将生成内存泄漏报告。

Linux/Chrome

在终端中，使用已编译的 Lab3 可执行文件导航到 Lab 3 文件夹下，输入：

valgrind –泄漏检查=是./Lab3

将生成内存泄漏报告。

实验问题 1 – 自定义字符串库原型 [15 分]

这将是我们将在 2SH4 中构建的第一个自定义 C 库。请放心，这个库在您的项目开发过程中将非常方便。本实验题要求您构建以下四个字符串库函数，并确保具有自定义实现的每个库函数在打包到 PPA3 中的可重用字符串库之前可以通过所有单元测试。

- **整数**我的_strlen(**常量字符 * 常量**)
  哦 此函数返回直到但不包括空终止字符的字符数。例如，给定输入str =“你好”，返回值应该是 5，
  哦 而不是 6。

- **整数**我的_strcmp（**常量字符 * 常量**字符串1,**常量字符 * 常量**字符串2)
  哦该函数比较两个字符串，如果内容不同则返回 0，如果相同则返回 1。

- **整数**my_strcmpOrder(**常量字符 * 常量**字符串1,**常量字符 * 常量**字符串2)
  哦 此功能提供了以下扩展功能my_strcmp()通过识别之间的 ASCII 字符顺序字符串1和海峡2。

    - 如果内容字符串1大于字符串2按 ASCII 顺序，返回 1
    - 如果内容字符串1小于字符串2按 ASCII 顺序，返回 0
    - 如果两个字符串的内容相同，则返回-1

  哦例如：
    - 给定str1 =“你好”，str2 =“世界”，my_strcmpOrder（str1，str2）将返回 0，因为之间的第一个不同字符字符串1和字符串2是 'H'（72) vs 'W'(87）。
    - 给定str1 =“约翰尼”，str2 =“杰弗里”，my_strcmpOrder（str1，str2）将返回 1、因为第一个字符之间不同字符串1和字符串2是 '哦（111) 与 'e'(101）。
    - 给定str1 =“保佑”，str2 =“保佑你”，my_strcmpOrder(str1，str2)将要返回 0，因为之间的第一个不同字符字符串1和字符串2是无效的（0) 与 ' '(32)

- **字符***my_strcat(**常量字符 * 常量**字符串1,**常量字符 * 常量**字符串2)
  哦该函数按以下方式返回两个输入字符串的连接字符串：
    - 分配一个能够保存组合字符串内容的字符数组字符串1和字符串2 在堆上。
    - 复制非无效的内容在字符串1,然后字符串2,到字符数组中，然后用 a 终止连接的字符串无效的特点。
    - 返回指向堆上连接字符串的指针。

  哦例如：
    - 给定str1 =“你好”，str2 =“世界！”，my_strcat(str1，str2)将返回一个指向字符串的指针“你好世界！”。

  哦 **警告！！！** 这是第一个**内存不安全** 需要您注意堆内存分配和释放的函数。为了您的学习目的，您只需担心malloc()目前在功能实现中。这自由的（）对应部分已在单元测试设置中得到处理测试用例.c。但是，您需要仔细分析代码如何与单元测试用例交互 - 重点了解组合字符串的堆内存的分配和释放位置，以及其在堆上的完整生命周期。熟悉这一点将有助于您完成问题 2 中内存管理的完整过程。

为每个功能添加至少 2 个测试用例，以确保其功能得到正确实现。想一些***边缘情况***不完整的算法可能会失败。

评分方案

-代码实现

哦 **[1 分]**正确执行my_strlen()功能 **[2 分]**正确执行my_strcmp()功

哦 能

哦 **[3分]**正确执行my_strcmpOrder()功能

哦 **[4分，下面有细分]**正确执行my_strcat()功能

- [1 分] 分配了正确数量的字符空间来保存连接的字符串
- [2 分] 以正确的顺序复制两个字符串中的字符
- [1标记] NULL字符的正确插入

-代码行为、分析和测试计划 哦

[1分]正确的程序行为 – 通过所有默认测试用例。 **[4分，每个功能**

哦 **1分]**额外的测试用例，至少 2 个。

（该项目是**全有或全无**， IE，**仅提供 1 个测试用例将不会奖励任何分数**）

-内存管理惩罚 哦

分割失败将导致额外扣2分

哦 任何检测到的内存泄漏都会额外扣除 4 分

实验问题 2 – 对文件中的单词数组进行排序 [15 分]

在本问题中，您将应用问题 1 中的自定义字符串库对从提供的输入文件读取的字符串数组进行排序单词表.txt使用我们在过去两个实验中介绍的两种排序方法——冒泡排序和选择排序。您必须完成以下四个接口函数和一个辅助函数才能完成所需的任务。实验手册末尾提供了有关如何访问 C 语言文件的快速说明。您需要阅读它才能完成此问题。

作为另一个调试器实践，提供了单词排序功能的选择排序实现，但有 2 个语义错误。使用调试器找到它们，修复错误并生成调试报告。

- **字符****读取单词（**常量字符***输入文件名，**整数***nPtr)
  - 哦 该函数打开由输入文件名在 ASCII 读取模式下，将文件中的每一行作为字符串读取，并将各个字符串保存在堆上的字符串数组（字符的二维数组）中。返回字符串数组在堆上的地址。
  - 哦 使用框架代码注释中的提示来帮助您开发和测试实现。
- **空白**交换（**字符****字符串1,**字符****字符串2）
  - 哦 该函数通过指针操作交换两个字符串的内容。
  - 哦 该函数以引用传递模式接受两个字符串的起始地址。注意：您不应将这两个参数解释为普通双指针。
  - 哦 交换两个字符串的内容就像交换两个 str 指针中存储的起始地址一样简单。查看讲义以了解其机制。
- **空白**删除单词列表（**字符****单词表，**整数**尺寸）
  - 哦 该函数释放在堆上分配的字符串数组。
  - 哦 该函数接受指向字符串数组的指针以及数组中字符串的数量。查看讲义以开发正确的内存释
  - 哦 放例程。
- **空白**排序词气泡（**字符****字，**整数**尺寸）
  - 哦 此函数使用冒泡排序算法按升序 ASCII 值顺序对所有单词进行排序。使用
  - 哦 my_strcmpOrder()来自问题 1 的内容可帮助您比较字符串。
- **空白**排序词选择（**字符****字，**整数**尺寸）
  - 哦 此函数包含使用选择排序的字排序算法的错误实现。该函数包含 2 个语义错误。使用 GNU 调试器找
  - 哦 到它们并修复它们，并生成调试报告。

为每个功能添加至少 2 个测试用例，以确保其功能得到正确实现。想一些***边缘情况***不完整的算法可能会失败。

对于您的工作负载管理，**不要** 尝试找出一个大于 10 的数组。

评分方案

- 代码实现
  - 哦 **[2 分]**正确执行读单词（）功能 **[1分]**正确执行交换（）功能
  - 哦
  - 哦 **[2分]**正确执行删除_wordTable()功能 **[2 分]**正确执行sort_words_Bubble()
  - 哦 功能 **[2 分]**修复sort_words_Selection()功能
  - 哦
- 代码行为、分析和测试计划
  - 哦 **[1分]**正确的程序行为 – 通过所有测试用例 **[4分，细分如**
  - 哦 **下]**调试器报告
    - 对于这两个错误中的每一个
      - **[0.5分]**设置有效断点 **[0.5分]**设置有效的变量监视
      - **[0.5分]**显示越野车行为的证据 **[0.5分]**显示固定行
      - 为的证据
      -
  - 哦 **[1分]**额外的测试用例，至少 2 个。
    （该项目是**全有或全无**， IE，**仅提供 1 或 2 个测试用例将不会奖励任何分数**）

-内存管理惩罚

- 哦 **分割失败将导致额外扣2分**
- 哦 **任何检测到的内存泄漏都会额外扣除 4 分**

C/C++ 文件访问快速入门指南

C 和 C++ 中的文件访问库内容广泛且用途广泛，但主要在二进制模式或 ASCII 字符模式下运行。本快速入门指南将涵盖实验 3 – 以 ASCII 读取模式读取文件中所需的文件访问功能。其余的功能留给您自主学习。要使用 C 文件系统，必须包含标准库文件在C源代码中。

第 1 步：创建文件句柄（指向文件开头的指针）

要开始文件访问过程，我们必须创建一个文件指针来保存文件开头的地址。这是通过在文件访问例程的开头声明来完成的：

```
文件*我的文件；          // 俗称“文件句柄”
```

步骤 2：以 ASCII 读取模式打开文件

然后，我们将使用以下命令打开所需的文件fopen()函数，它接受第一个参数中的文件名，第二个参数中的访问模式 - “”用于 ASCII 读取模式。您还可以使用许多其他访问模式。

成功打开文件后，fopen()返回文件开头的地址，我们应该使用步骤 1 中的文件句柄捕获该地址。如果访问失败，fopen()将返回一个 NULL 指针，可用于错误检查。

```
myFile = fopen(“文件名.doc”，“r”);
// 以 ASCII 读取模式打开文件 FileName.doc
```

从此时起，可以通过以下方式访问文件内容我的文件文件句柄。然后，您打开的文件将保持锁定并被您的程序独占，直到您在步骤 4 中关闭文件访问权限。

步骤 3：以所需格式读取内容行

要从文件中读取内容行，我们将使用 fscanf() 函数，如下所示：

```
字符缓冲区[20];
fscanf(myFile，“%s”, 缓冲区);
```

该代码片段执行以下操作：

- 从文件中读取一行内容，直到**空格键** 特点。
- 将读取的内容解释为字符串。更改不同值解释的转换说明符。将字符串保存到字符串缓冲区中。如果字符串
- 大于缓冲区大小，请注意缓冲区溢出。 **不明显但很重要**：将文件句柄中的地址前进到文件的下一行，这样当
- 第二次调用fscanf()时，就会从后续行的内容中读取。

称呼fscanf()根据需要多次从文件中读取内容。您可以研究其他文件读取功能**文件结束符 (EOF)**从文件读取时提高代码可靠性的标志。

步骤4：关闭文件句柄（即释放对文件的访问权限）

文件访问完成后，您必须关闭对该文件的文件访问。这一步是一个***必须做***!! 否则，其他程序将无法访问该文件。

```
fclose(myFile);          // 俗称“释放句柄”
```