Assignment 7: Stanford 1-2-3

作业 7：斯坦福 1-2-3

This is your last CS106X assignment! It is a chance to pull together your stellar C++ skills, design a complicated data structure, use a variety of existing classes, design and implement a few new ones, and build an awesome piece of productivity software. It's a wonderful and sophisticated task that is a capstone to all you've done so far. We can't think of a better way to top off our intense journey.

这是 CS106X 的最后一项作业！这是将你们出色的 C++ 技能集中在一起，设计一个复杂的数据结构，使用各种现有类，设计并实现一些新类，并构建一个出色的生产力软件的机会。这是一项精彩而复杂的任务，是你迄今为止所做一切的顶点。没有比这更好的方式来结束我们紧张的旅程了。

Due: Tuesday, November 19th at 11:59 p.m. [1]

提交时间：11 月 19th 日（星期二）晚上 11:59 时 [1]

The Assignment 作业

Your mission is to build a simple spreadsheet, starting with a slightly modified version of the expression evaluator presented in Chapter 19. This assignment is designed to accomplish the following objectives:

你的任务是制作一个简单的电子表格，首先要对第 19 章中介绍的表达式求值器稍作修改。这项任务旨在实现以下目标：

To more fully explore the notion of object-oriented programming. The program is broken down into classes that cooperatively interact. Almost every one of the classes we studied this quarter plays some role in the overall program architecture.

为了更充分地探索面向对象编程的概念。程序被分解为多个类，这些类之间相互协作。本季度我们学习的几乎每一个类都在整个程序架构中扮演着某种角色。

To learn how C++ inheritance can be used for expression trees and how to implement simple recursive-descent parsing.

学习如何将 C++ 继承用于表达式树，以及如何实现简单的递归-后裔解析。

To give you practice working with graphs and graph algorithms.

练习使用图形和图形算法。

To get a taste of the Model/Niew/Controller (MVC) structure used by many modern applications.

了解许多现代应用程序使用的模型/视图/控制器（MVC）结构。

To learn how to adapt existing code (in this case, the expression interpreter) to solve a different but related task. The majority of programming people do in the industry consists of modifying existing systems rather than creating them from scratch.

学习如何调整现有代码（在本例中为表达式解释器），以解决不同但相关的任务。业内大多数编程工作都是修改现有系统，而不是从头开始创建。

To experience the joys and frustration of designing a class interface/implementation.

体验设计类界面/实现的乐趣和挫折。

The task may sound a bit daunting, but never fear, there is a fair amount of infrastructure in place already. However, there is still much for you to do! Make it your personal goal to have your final assignment be one that genuinely rocks.

这项任务听起来可能有点艰巨，但不用担心，我们已经有了相当数量的基础设施。不过，你还有很多事情要做！让你的最终作业成为真正震撼人心的作业，这是你的个人目标。

A note on open-ended design

关于开放式设计的说明

Although we give you a lot of starter code and many suggestions, this assignment is more open-ended than most and offers you the freedom to design things the way you want. There are a few isolated tasks where we mandate a particular implementation strategy, but other than that, it's up to you to make sensible decisions. Your program is expected to portray the described behavior and work similarly to the demo version, but you're being given broad authority over the choices you make to build out a working product. This kind of open-ended design can be creative and fun, but there is also the risk that you'll go astray with suboptimal choices that you later have to live with. We recommend starting the design process early and carefully thinking through alternatives and tradeoffs. We encourage you to run your design by your section leader-over email is fine-before you start coding to help you identify and correct potential problems earlier rather than later.

虽然我们为您提供了大量的启动代码和许多建议，但这项任务比大多数任务更加开放，您可以按照自己的想法自由设计。在一些孤立的任务中，我们会强制要求采用特定的实施策略，但除此之外，您可以自行做出明智的决定。我们希望您的程序能描绘出所描述的行为，并能与演示版本类似地运行，但您在构建可运行产品的过程中拥有广泛的选择权。这种开放式的设计可能很有创意，也很有趣，但也有可能会误入歧途，做出次优的选择，最终不得不接受。我们建议您尽早开始设计过程，仔细考虑各种选择和权衡。我们鼓励您在开始编码之前，通过电子邮件向您的部门领导汇报您的设计，以帮助您及早发现和纠正潜在的问题。

The Goal 目标

One of the most important commercial programs to emerge from the personal computer revolution was the electronic spreadsheet. The original VisiCalc system was a runaway success for Apple in the early 1980s, and many more advanced products, such as Microsoft Excel and Google Sheets, have extended that basic idea so much that spreadsheet applications are now used as the basis for an astonishingly wide range of commercial use cases. At its core, a spreadsheet consists of a two-dimensional grid of cells, each indicated by a letter representing a column and a number representing a row, as illustrated in the diagram below:

电子表格是个人电脑革命中出现的最重要的商业程序之一。最初的 VisiCalc 系统在 20 世纪 80 年代初为苹果公司带来了巨大的成功，而许多更先进的产品，如 Microsoft Excel 和 Google Sheets，都对这一基本理念进行了扩展，以至于电子表格应用程序现在已被用作各种商业用途的基础，其范围之广令人吃惊。电子表格的核心是一个由单元格组成的二维网格，每个单元格由代表一列的字母和代表一行的数字组成，如下图所示：

	A	B	c	D	E	F	G
1	Item 项目	Quantity 数量	Unit price 单位价格	Cost 费用			
2	Partridge 鹧鸪	1	129.99	129.99			
3	Turtle dove 龟鸽	2	31.17	62.34			
4	French hens 法国母鸡	3	15.79	47.37			
5	Calling birds 呼唤鸟儿	4	35.99	143.96			
6	Gold rings 金戒指	5	499.95	2499.75			
7	Geese 鹅	6	59.24	355.44			
8	Swans 天鹅	7	40.15	281.05			
9	Maids 女仆	8	112.37	898.96			
10	Ladies 女士们	9	115.21	1036.89			
11	Lords 上议院	10	115.21	1152.1			
12	Pipers 管道工	11	88.55	974.05			
13	Drummers 鼓手	12	75.99	911.88			
14	tOTAL 总计			8493.78			
15							
16							
17							
18							
19							

In the spreadsheet, each cell contains a value, which can be:

在电子表格中，每个单元格都包含一个值，可以是

a string, such as “Item” in A1 or “French hens” in A4.

字符串，如 A1 中的 "项目 "或 A4 中的 "法国母鸡"。

a number, such as the quantities in column B or the prices in column C. Note that these values can have decimal fractions and must therefore be represented using type double. (The expression hierarchy has been updated to accommodate this change.)

请注意，这些值可以是十进制分数，因此必须使用 double 类型表示。(表达式层次结构已更新以适应这一变化）。

a formula linking other items in the spreadsheet. Presumably, cell D2 was set so that its value is calculated by multiplying the values in cells B2 and C2. Similarly, cell D14 is the sum of the values in cells D2 through D13. Cells that reference other cells are said to have dependencies. If the value in **B2** or **C2** changes, D2 will also need to be updated since it depends on those inputs.

连接电子表格中其他项目的公式。据推测，单元格 D2 的值是通过单元格 B2 和 C2 的值相乘计算得出的。同样，单元格 D14 是单元格 D2 至 D13 中数值的总和。引用其他单元格的单元格被称为具有依赖关系。如果 **B2** 或 **C2** 中的值发生变化，D2 也需要更新，因为它依赖于这些输入。

Commands for the Spreadsheet Controller

电子表格控制器的命令

Let’s first examine the program from a user perspective and postpone our discussion of the internal machinery until we know what the program does. It opens with a new empty spreadsheet. Using a simple command-line interpreter interface reminiscent of a bad flashback to the 70 s, the user can enter text commands that operate

on the spreadsheet.

首先，让我们从用户的角度来研究一下这个程序，在了解程序的功能之前，我们暂且不讨论它的内部机制。程序打开时是一张新的空白电子表格。通过一个简单的命令行解释器界面，用户可以输入文本命令对电子表格进行操作。

The command load <filename> reads the contents of a previously saved spreadsheet from the named file.

加载 <filename> 命令从指定文件中读取先前保存的电子表格内容。

The command save <filename> writes the current contents of the spreadsheet to the named file.

保存 <filename> 命令将电子表格的当前内容写入指定文件。

The command clear clears the current contents of the spreadsheet.

命令清除电子表格的当前内容。

The command set <cellname> = <value> sets the current contents for the given cell, replacing any previous contents for that cell. The cell name is specified by column and row such as A3. The value can be a string enclosed in double-quotes or a numeric expression. Below are some examples:

命令设置 <ellname> = <value> 将设置给定单元格的当前内容，并替换该单元格以前的内容。单元格名称由列和行指定，如 A3。值可以是用双引号括起来的字符串，也可以是数字表达式。下面是一些示例：

```
set A2 = "Beat Cal"
set B2 = 13.5
set C2 = B2 * (3 + A1)
```

If the cell name or value is invalid or malformed, an error is reported and the command discarded. Otherwise, the new contents are displayed and all cells that depend on the updated value are themselves updated.

如果单元格名称或值无效或畸形，则会报错并放弃命令。否则，将显示新的内容，并且所有依赖于更新值的单元格都会被更新。

The command get <cellname> prints information for a given cell which include its contents and a list of the cell's dependencies: both those cells that this one directly depends on and those cells that directly depend on it. (We'll explain more about dependencies later in this handout). For example, given the cell contents above in the set command, set C2 would print:

获取 <ellname> 命令将打印指定单元格的信息，包括其内容和该单元格的依赖关系列表：既包括该单元格直接依赖的单元格，也包括直接依赖该单元格的单元格。(本手册稍后将详细介绍依赖关系）。例如，如果在 set 命令中给出上述单元格内容，set C2 将打印：

```
C2 = (B2 * (3 + A1))
Cells that C2 directly depends on: A1 B2
Cells that directly depend on C2:
```

You are also welcome to use the get command to print any additional cell information (such as the indirect dependents) useful to your development and debugging. In fitting with our general philosophy for this assignment, your output doesn't have to match this output exactly; you just need to ensure that all our required information is present.

您也可以使用 get 命令来打印任何对您的开发和调试有用的附加单元格信息（如间接依赖项）。根据我们本次作业的总体思路，您的输出不必与此输出完全一致；您只需确保我们要求的所有信息都已存在。

The command help prints a simple help message describing the available commands.

命令 help 会打印一条简单的帮助信息，说明可用的命令。

The command quit quits from the program. #obvi

命令 quit 可以退出程序。#obvi

Overview: Program Structure

概述：计划结构

The spreadsheet program is internally structured using the Model/View/Controller (MVC) design pattern favored by modern GUI applications. The model manages the data being stored. A view displays a visual representation of the model. The controller provides a user interface (be it graphical widgets or a retro command-line) that offers the user a way to make changes to the model. The controller responds to the user actions by messaging the model to update the data. When the model is changed, it notifies its views to render the new data. The benefit of MVC is that it divides the code into clean areas of responsibility, makes it possible to have multiple views/controllers on the same model, and allows you to easily try out different implementations for each component.

电子表格程序的内部结构采用了现代图形用户界面应用程序所青睐的模型/视图/控制器（MVC）设计模式。模型管理存储的数据。视图显示模型的可视化表示。控制器提供用户界面（无论是图形部件还是复古的命令行），为用户提供更改模型的方法。控制器通过向模型发送更新数据的消息来响应用户的操作。当模型发生变化时，控制器会通知其视图来呈现新数据。MVC 的好处在于，它将代码划分为清晰的责任区域，使在同一模型上使用多个视图/控制器成为可能，并允许您轻松尝试每个组件的不同实现方式。

In the case of the spreadsheet, we supply the controller (the command-line interface) and the view (the graphical display). The model is left for you to design and implement. This class is where you will concentrate your efforts, while making compatible modifications to some of the other modules.

就电子表格而言，我们提供控制器（命令行界面）和视图（图形显示）。模型则由您来设计和实现。在对其他一些模块进行兼容修改的同时，您可以将精力集中在该类上。

Here is a summary of the program structure:

以下是计划结构概要：

sscontroller This module houses the main function, which is responsible for the text-based interface. It uses a TokenScanner (documentation is accessible via the CS106X web site) to break apart the command line and messages the spreadsheet model with the changes. Most of this module is written, but you'll need to extend it to support one additional command.

sscontroller 该模块包含主函数，负责基于文本的界面。它使用 TokenScanner（可通过 CS106X 网站获取文档）来分解命令行，并向电子表格模型发送更改信息。该模块的大部分内容已经编写完成，但您还需要对其进行扩展，以支持另外一条命令。

SSView This class provides a graphical spreadsheet display. We provide the complete class and you will not

need to make changes to it unless you want to.

SSView 该类提供图形电子表格显示。我们提供了完整的类，除非您愿意，否则无需对其进行修改。

SSModel SSM 模型
This class manages the spreadsheet and cell data model. We provide a skeletal public interface and you will finish the design and provide the class implementation.

该类管理电子表格和单元格数据模型。我们提供了一个骨架公共接口，您将完成设计并提供类的实现。

Expression The Qt project uses the exp and parser modules from the Chapter 19 expression evaluator with a few adjustments. Most of this code will be used as is, but you will make additional modifications to support features required for the spreadsheet formulas.

表达式 该 Qt 项目使用了第 19 章表达式评估器中的 exp 和解析器模块，并做了一些调整。大部分代码将按原样使用，但您还需要进行额外的修改，以支持电子表格公式所需的功能。

ssutil This module provides a few utility functions and the range formula functions (median, sum, max, etc.) for use in formulas. You may or may not need to make modifications to this module.

ssutil 该模块提供一些实用函数和范围公式函数（中值、和、最大值等），供公式使用。您可能需要也可能不需要对该模块进行修改。

The rest of this handout focuses on the modules you'll work in and even suggests a course of action as to how you might approach the assignment. We've put the task breakdown last this time for a reason; we strongly suggest you read this handout thoroughly and get an idea of how all the pieces mesh together before you start doing any design or coding!

本手册的其余部分重点介绍了您将要完成的模块，甚至还就如何完成任务提出了建议。我们将任务分解放在最后是有原因的；我们强烈建议您在开始设计或编码之前，彻底阅读本手册，了解所有部分是如何组合在一起的！

The sscontroller, ssview, and ssutil modules

sscontroller、ssview 和 ssutil 模块

We provide these three modules to you in complete or nearly complete form.

我们为您提供这三个模块的完整版或基本完整版。

The sscontroller module contains the main program loop that interacts with the user, reading and acting on commands. It uses a TokenScanner to process the user's input and determines the appropriate action using a little command-dispatch table. Our code correctly implements the controller responsibilities, except that it is missing the clear command, which clears the current spreadsheet contents. You're to add this command to the controller.

sscontroller 模块包含与用户交互的主程序循环，读取并执行命令。它使用一个 TokenScanner 来处理用户的输入，并使用一个小命令调度表来决定适当的操作。我们的代码正确地实现了控制器的职责，只是缺少了清除当前电子表格内容的 clear 命令。您需要将该命令添加到控制器中。

Note that the controller is tightly coupled to the expression evaluator code from Chapter 19. One difference from the code given in the text is that the interpreter loop for the spreadsheet controller contains additional code to recover from errors. If you are entering a formula and make a syntax error, you do not want your entire spreadsheet to bomb out. Even so, it is extremely convenient in the code to call error to produce the error messages. Our implementation of the error function-which you've been seduced into believing automatically terminates program execution-actually throws an exception that's caught in the primary read-evaluate-print

loop (sometimes abbreviated as the repl).

请注意，控制器与第 19 章中的表达式求值器代码紧密耦合。与文中给出的代码不同的是，电子表格控制器的解释器循环包含了额外的代码，用于从错误中恢复。如果您在输入公式时出现语法错误，您不会希望整个电子表格都炸掉。即便如此，在代码中调用 error 来生成错误信息还是非常方便的。我们对 error 函数的实现--你曾误以为它会自动终止程序的执行--实际上是抛出一个异常，在主读取-评估-打印循环（有时简称为 repl）中被捕获。

The ssview module provides the class that manages the appearance of the spreadsheet in the graphics window. It includes member functions for displaying an empty spreadsheet and displaying the contents of a cell. Your model should send messages to the view as needed to update the display when changes are made to the model. Comments in the **ssview. h** file describe the public features of the class.

ssview 模块提供了管理电子表格在图形窗口中的外观的类。它包括用于显示空电子表格和显示单元格内容的成员函数。当模型发生变化时，模型应根据需要向视图发送信息，以更新显示内容。 **ssview. h** 文件中的注释描述了该类的公共功能。

The ssutil module has a few little utilities that didn't quite fit anywhere else. It defines location and range types, functions to convert a cell name to location and vice versa, and code for the range formula functions (average, sum, max, etc.). You are free to extend, change, and otherwise cannibalize the code here in any way you find helpful.

ssutil模块有一些其他地方没有的小工具。它定义了位置和范围类型、将单元格名称转换为位置的函数（反之亦然），以及范围公式函数（平均值、总和、最大值等）的代码。您可以扩展、更改或以其他任何您认为有用的方式拆用此处的代码。

Hints and requirements for these modules:

这些模块的提示和要求：

Adding the clear command to the controller requires just a few lines of code, but you must first work through the controller code to understand how to fit the required code into the overall program architecture.

在控制器中添加清除命令只需几行代码，但您必须首先通读控制器代码，了解如何将所需代码融入整个程序架构。

A range represents a set of cells between a start and stop cell inclusively. A range can span just one row or column or enclose a two-dimensional rectangular block of cells. Thus, ranges like **A1 : A4**, **A1 : A1**, **A1 : D1**, and **A1 : D6** are valid. One thing to note is that a valid range is required to be nonempty, which means the stop cell must be at a position that's at least equal to the row and column of the start cell.

范围表示起始单元格和终止单元格之间的一组单元格。范围可以只跨一行或一列，也可以包含一个二维矩形单元格块。因此，像 **A1 : A4**, **A1 : A1**, **A1 : D1** 和 **A1 : D6** 这样的范围都是有效的。需要注意的是，有效的范围必须是非空的，这意味着停止单元格的位置必须至少等于起始单元格的行和列。

The range record defined in ssutil stores a location for start and stop. An alternative definition might represent the start and stop as string cell names. In some places, you refer to a cell by its string name, other times you need its components, so both approaches will require translation. ssutil supplies simple conversion functions.

在 ssutil 中定义的范围记录存储了开始和停止的位置。另一种定义可能是以字符串单元格名称表示开始和停止。ssutil 提供了简单的转换函数。

You can modify the supplied range formula functions to fit with your mechanism to access a range of cells (e.g. have the functions directly operate on your model) or just use the functions as given (you first extract the

needed values into a Vector). Either approach is fine with us.

您可以修改所提供的单元格区域公式函数，使其符合您访问单元格区域的机制（例如，让函数直接对您的模型进行操作），或者直接使用所提供的函数（首先将所需值提取到矢量中）。无论哪种方法，我们都可以接受。

Matching a range formula function name median to the appropriate function to execute should be implemented using the command table approach, like that used in the controller module. It should be a simple task to add new range formula functions if you've designed it well.

将范围公式函数名称中值与要执行的适当函数相匹配，应使用命令表方法来实现，就像在控制器模块中使用的方法一样。如果设计得当，添加新的范围公式函数应该是一件很简单的事情。

Expressions and Parsing 表达式和解析

Cell formulas can be built out of real numbers, references to other cells, parentheses, the four operators +, −, *, and /, as well as built-in functions applied to cell ranges. Here are examples of some possible cell formulas:

单元格公式可以由实数、对其他单元格的引用、括号、四个运算符 +, −, * 和 / 以及应用于单元格区域的内置函数组成。下面是一些可能的单元格公式示例：

```
5 * 1.08
A1 + A2
"Beat Cal!"
SUM(B1:B5)
(3 + A5)/average(A1:B5) + D10 - 5.5
```

With a few modifications, the Expression classes are the perfect mechanism for managing cell contents. We give you the code from Chapter 19 as your starting point, with our changes to allow floating point constants (instead of integer constants) and to add string literals enclosed in double-quotes as a new expression type. Although most of the expression code will be used as is, you do need to understand it thoroughly so you can properly leverage it for your own purposes.

只需稍加修改，Expression 类就能成为管理单元格内容的完美机制。我们将以第 19 章中的代码为起点，对其进行修改，允许使用浮点常量（而不是整数常量），并添加双引号括起来的字符串字面量作为新的表达式类型。虽然大部分表达式代码都将按原样使用，但您仍需要彻底理解这些代码，这样才能根据自己的目的正确使用这些代码。

The changes you are to make:

您要做出的改变：

Adapt expressions to work in the context of the spreadsheet model. Whereas the ordinary expression evaluator allows arbitrary use of identifier names for variables through a variable table, variables now must be references to spreadsheet cells. Modify the parsing code so it accepts only cell names as identifiers, and update the evaluation code to retrieve the values for cells from the spreadsheet model.

调整表达式，使其在电子表格模型环境中运行。普通表达式求值器允许通过变量表任意使用变量的标识符名称，而现在变量必须是电子表格单元格的引用。修改解析代码，使其只接受单元格名称作为标识符，并更新评估代码，以便从电子表格模型中获取单元格的值。

Add support for a new expression type of a function applied to a range. A range function expression applies a named function to a cell range, e.g. sum(A1:A5) applies the sum function to all cells from A1 to A5 inclusive.

The modified grammar for terms becomes:

增加对应用于单元格区域的函数新表达式类型的支持。单元格区域函数表达式将指定函数应用于单元格区域，例如，sum(A1:A5) 将求和函数应用于从 A1 到 A5（包括 A5）的所有单元格。修改后的术语语法变为

```
T -> "string"
T -> number
T -> cellname
T -> function(cellname:cellname)
T -> (E)
```

This will require adding a new Expression subclass and making changes to the parsing code. The named functions that you are required to support are listed in the ssutil.h interface file.

这需要添加一个新的 Expression 子类并修改解析代码。ssutil.h接口文件中列出了您需要支持的命名函数。

3. Add support for identifying dependencies. In order to report cell dependencies, you need to be able to find the dependent references within an expression. This is a matter of walking the expression tree and reporting the dependencies found within the subexpressions.

3.增加识别依赖关系的支持。为了报告单元格的依赖关系，需要在表达式中找到依赖的引用。这就需要遍历表达式树，并报告在子表达式中发现的依赖关系。

Hints and requirements for expressions and parsing:

表达式和解析的提示和要求：

Starting from a working program that solves a different problem (in this case, the expression evaluator from the textbook) is a blessing and a curse. The Expression classes and parsing code will be a great help, but you may find yourself swimming in code at first and unsure of how to proceed. It is essential that you understand the workings of the given expression code. We recommend re-reading Chapter 19 and going over the code with a fine-toothed comb. If there's anything you don't understand, be sure to ask. You do not need to make significant modifications but figuring out how and where to make changes requires you understand the existing code base.

从解决不同问题的工作程序（本例中为教科书中的表达式求值器）入手，是一件好事，也是一件坏事。表达式类和解析代码会给你很大的帮助，但一开始你可能会发现自己在代码中游刃有余，不知如何下手。理解所给表达式代码的工作原理至关重要。我们建议您重新阅读第 19 章，并仔细梳理代码。如果有不明白的地方，一定要问清楚。您不需要进行重大修改，但要想知道如何修改以及在哪里修改，就必须了解现有的代码库。

In the original expression evaluator, the assignment operator = could be part of an expression. In the spreadsheet, the = is a throwaway character in a set statement, and the expression that follows it can involve the arithmetic operators but not the = operator. We removed that feature from the expression modules we give you to avoid confusion.

在最初的表达式求值器中，赋值运算符 = 可以是表达式的一部分。在电子表格中，= 是集合语句中的一个空闲字符，它后面的表达式可以包含算术运算符，但不包括 = 运算符。为了避免混淆，我们从提供的表达式模块中删除了这一功能。

When parsing a cell formula, the parser should reject all malformed inputs (improper cell reference, unknown range formula function, invalid range, and so on). There are a lot of cases to consider, so do be thoughtful and test carefully. Use error to report the problem and the exception handling in the controller will catch it and go

on.

解析单元格公式时，解析器应拒绝所有畸形输入（不正确的单元格引用、未知的单元格区域公式函数、无效的单元格区域等）。需要考虑的情况很多，因此一定要深思熟虑并仔细测试。使用错误报告问题，控制器中的异常处理将捕获问题并继续处理。

When evaluating a cell formula, a reference to an empty or string cell is assumed to have value 0 . If $\mathbf{A1} = \mathbf{A2} + \mathbf{5}$ and $\mathbf{A2}$ = "hello", then $\mathbf{A1}$ will show the result $\mathbf{5}$. Similarly, a function such as sum applied to a range of string cells would have a zero result.

在求值单元格公式时，对空单元格或字符串单元格的引用被假定为值为 0。如果 $\mathbf{A1} = \mathbf{A2} + \mathbf{5}$ 和 $\mathbf{A2}$ = 为 "hello"，那么 $\mathbf{A1}$ 将显示结果 $\mathbf{5}$ 。类似地，应用于字符串单元格区域的函数（如 sum）的结果也是 0。

Formulas should be case-insensitive: cell references A2 and a2 and functions median and Median are the same thing. Be sure to use the virtual keyword to get the proper dynamic dispatch for any new member functions added to the base Expression class that are intended to be overridden by subclasses. Just for reference, the expression/parsing modifications involve changing/adding about 50 lines of code.

公式应区分大小写：单元格引用 A2 和 a2 与函数 median 和 Median 是一回事。请务必使用 virtual 关键字，以便为添加到基本表达式类的任何新成员函数获取适当的动态调度，这些函数将被子类覆盖。作为参考，表达式/解析修改涉及更改/添加约 50 行代码。

The spreadsheet model 电子表格模型

Your main task is designing and implementing the SSModel class to manage the cell data. We provide a skeletal interface that lists exactly those public features needed to interact properly with our controller and view. You are to finish the design of the interface and implement the class. This is an excellent opportunity for you to think through the various options and make the decisions to suit yourself. In the real world, it is rare that code you must write comes to you fully specified, and we want you to gain some experience in the issues and tradeoffs that come up.

您的主要任务是设计和实现 SSModel 类来管理单元格数据。我们提供了一个骨架接口，其中准确列出了与控制器和视图正确交互所需的公共功能。你要完成接口的设计并实现该类。这是一个绝佳的机会，让你思考各种选项，并做出适合自己的决定。在现实世界中，很少有你必须编写的代码是完全指定的，我们希望你能在遇到问题和权衡时获得一些经验。

In general, the model is responsible for storing the contents of the cells, managing the relationships between cells, responding to requests from the controller, and notifying the view when things change. You're likely to find designing this class to be an iterative process-you sketch out the features, but as you move forward, you uncover problems or features that necessitate further refinement. Perhaps as the implementer you find some operations difficult or inefficient to support, or as the client, you encounter tasks that are awkward or even impossible. Back to the design drawing board, to make additions and adjustments as needed.

一般来说，模型负责存储单元格的内容、管理单元格之间的关系、响应控制器的请求以及在情况发生变化时通知视图。你可能会发现设计这个类是一个反复的过程--你勾勒出功能，但在前进的过程中，你会发现一些问题或功能需要进一步完善。也许作为实施者，你会发现某些操作难以支持或效率低下，或者作为客户，你会遇到一些笨拙甚至不可能完成的任务。回到设计绘图板，根据需要进行补充和调整。

Most of the implementation strategy is left unspecified. Consider the options and make your own well-reasoned decisions. Keep in mind that you have the full collection of our classes at your disposal (sets, grids, maps, and so on) and more than one may be useful

大部分实施战略都没有具体说明。请考虑各种选项，并做出合理的决定。请记住，您可以使用我们的全部类库（集合、网格、地图等），而且不止一个类库可能有用

here. Here are some questions to get you started thinking about the kind of issues that you need to address:

在这里。以下是一些问题，可帮助您开始思考需要解决哪些问题：

Should cells be created for each cell in the spreadsheet from the start or only on demand? What reasons are there to prefer one approach to the other?

应该从一开始就为电子表格中的每个单元格创建单元格，还是只在需要时才创建？有什么理由更倾向于一种方法而不是另一种方法？

What might the model use to store cells: A grid? A set? A map? What supports easy lookup up by cell name? What about by location? What allows easy iteration/mapping over the cells? Would two ways of accessing the cells make sense or would it be overkill?

模型用什么来存储单元？网格？集合？地图？什么能支持按单元格名称轻松查找？按位置查找呢？怎样才能方便地迭代/映射单元格？有两种访问单元格的方式吗？

How are the contents for each cell stored? What needs to happen when a cell's contents change?

如何存储每个单元格的内容？当单元格的内容发生变化时，需要做些什么？

Should a cell cache its computed result or re-compute on the fly? If you store only the formula, each time you need the result you must re-evaluate it. Instead you could evaluate the expression once and cache the result, and use it repeatedly, only discarding and re-computing when a dependency changes. What are the tradeoffs between the two approaches?

单元格应该缓存其计算结果还是在运行中重新计算？如果只存储公式，每次需要结果时都必须重新评估。相反，您可以对表达式进行一次评估并缓存结果，然后重复使用，只有在依赖关系发生变化时才丢弃并重新计算。这两种方法之间的权衡是什么？

How might you support accessing the cells/values within a range: iteration? a function that returns a set/vector of cell names or values? What is more convenient to use/implement? Are there good reasons to support more than one technique?

如何支持访问范围内的单元格/值：迭代？返回单元格名称或值的集合/向量的函数？哪种方法更方便使用/实现？是否有充分的理由支持一种以上的技术？

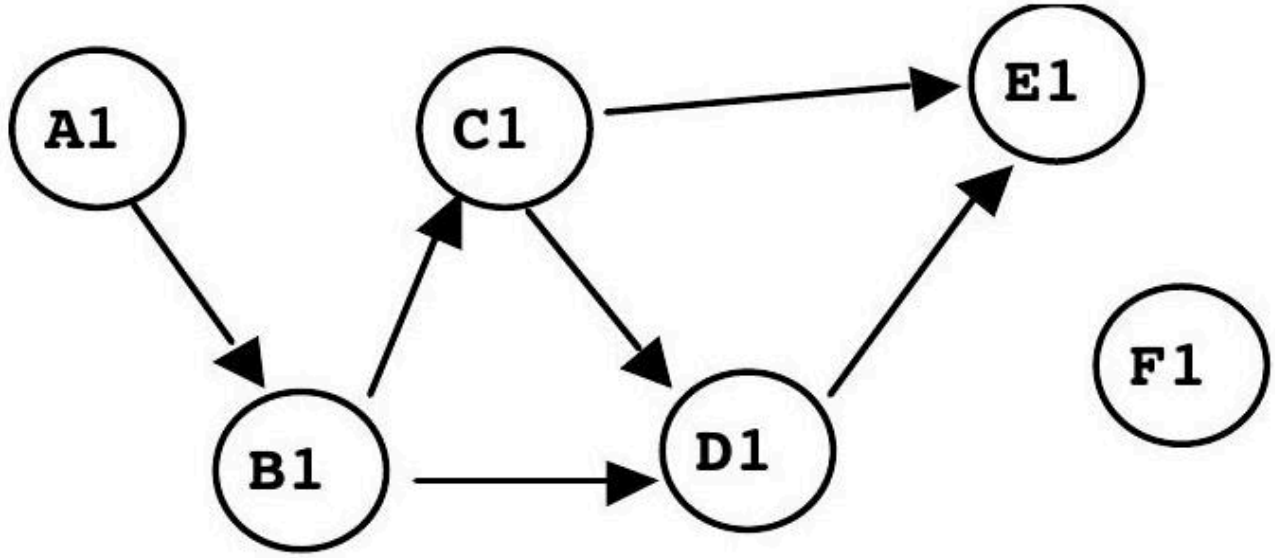
One of your greatest challenges is dealing with cell formulas. The trickiness comes in the fact that changing a value in one cell may start a chain reaction of updates. A cell that has a reference to another cell in its formula is said to be dependent on that cell. If the value in a cell is changed, the cells that depend on it also must be updated. Dependencies can either be direct (where a cell has an explicit reference to another in its formula), or indirect (where a cell has a chain of references that eventually lead to that cell). Consider the following spreadsheet file:

处理单元格公式是最大的挑战之一。其中的奥妙在于，改变一个单元格中的值可能会引发连锁更新。在公式中引用了另一个单元格的单元格被称为依赖单元格。如果某个单元格中的值发生变化，那么依赖该单元格的单元格也必须更新。依赖关系可以是直接的（一个单元格在其公式中对另一个单元格有明确的引用），也可以是间接的（一个单元格有一连串的引用，这些引用最终都指向该单元格）。请看下面的电子表格文件：

```
A1 = 10
B1 = A1 * 2
C1 = B1 + 5
D1 = C1 / B1
E1 = SUM(C1:D1)
F1 = 22
```

B1 directly depends on **A11C1** directly depends on B1, D1 directly depends on **B1** and **C1**, and **E1** directly depends on **C1** and **D1**. Both **C1** and **D**1 indirectly depend on **A1** (through **B1**), and **E1** indirectly depends on **A1** and **B1**. One useful way to visualize the dependencies is in terms of a directed graph, as shown below:

B1 直接依赖于 **A11C1** 直接依赖于 B1，D1 直接依赖于 **B1** 和 **C1** ， **E1** 直接依赖于 **C1** 和 **D1** 。 **C1** 和 **D**1 间接依赖于 **A1** （通过 **B1** ），而 **E1** 间接依赖于 **A1** 和 **B1** 。直观显示依赖关系的一种有用方法是有向图，如下图所示：



In this graph, the arcs trace the direction of propagating/outgoing dependencies. The same arcs in reverse show the incoming dependencies. For example, when **A1** changes, it needs to propagate an update to **B1** because the formula for **B1** directly references **A1**. Indirect dependencies are those cells connected through a longer path. D1 indirectly depends on **A1**, since it relies on **B1**, which in turn relies on **A1**, this indirect dependency is represented as a path of arcs from **A1** to **D**1.

在该图中，弧线表示传播/传出依赖关系的方向。相同的反向弧线表示传入的依赖关系。例如，当 **A1** 发生变化时，它需要向 **B1** 传播更新，因为 **B1** 的公式直接引用 **A1** 。间接依赖关系是指通过较长路径连接的单元格。D1 间接依赖于 **A1** ，因为它依赖于 **B1** ，而 **B1** 又依赖于 **A1** ，这种间接依赖关系表示为从 **A1** 到 **D**1 的弧路径。

When the value of a cell is updated, you must update all cells that depend on it, either directly or indirectly. Tracing the paths away from A1, you can see that changing the value in A1 will require four other cells to be updated. Changing F1, on the other hand, requires no changes to any other cells, since it has no outgoing dependencies. F1 also has no incoming dependencies, i.e., it is not affected by changes to any other cells.

当更新一个单元格的值时，必须直接或间接地更新所有依赖于该单元格的单元格。追溯从 A1 开始的路径，可以发现更改 A1 中的值需要更新其他四个单元格。另一方面，更改 F1 不需要更改任何其他单元格，因为它没有向外的依赖关系。F1 也没有输入依赖关系，也就是说，它不会受到任何其他单元格变化的影响。

Traversing the dependent cells sounds suspiciously like depth or breadth-first traversal of the graph. You can do this recursively or iteratively using a stack or a queue. A simple (and acceptable) version might update some cells multiple times because there is more than one path between them (consider how **D1** could be updated twice when traversing from A1). The really slick way is to do a topological sort to efficiently order the cell updates so that each dependent cell is updated at most once, only after its dependencies have been updated.

遍历从属单元格听起来很像深度或广度优先的图遍历。您可以使用堆栈或队列递归或迭代地执行此操作。一个简单的（可接受的）版本可能会多次更新某些单元格，因为它们之间存在不止一条路径（考虑一下从 A1 遍历时 **D1** 可能会被更新两次）。真正巧妙的方法是进行拓扑排序，有效地对单元格更新进行排序，这样每个从属单元格最多只能更新一次，而且只能在其从属单元格更新后更新。

There's one more bit of trickiness with dependencies. What if the formula for **A1** were **sum(A1: E1**) ? To calculate the value of **A1**, you need the value of **A1**, but you don't know what it is because you're still trying to calculate it! This kind of dependency is called a circular reference, and it's bad, bad news for spreadsheets. You

should disallow all circular references. An obvious circular reference would be an attempt to set $\mathbf{A1} = \mathbf{A1}$, e.g. where a cell directly references itself. The sneakier form is via an indirect reference such as assigning $\mathbf{A1}{=}\mathbf{E1}$ in the above example, which introduces a cycle in the graph.

依赖关系还有一个小技巧。如果 $\mathbf{A1}$ 的公式是 $\mathbf{sum(A1{:}\ E1}$ 呢？要计算 $\mathbf{A1}$ 的值，你需要 $\mathbf{A1}$ 的值，但你不知道它是什么，因为你还在试图计算它！这种依赖关系被称为循环引用，对电子表格来说是个坏消息。你应该禁止所有循环引用。一个明显的循环引用就是试图设置 $\mathbf{A1} = \mathbf{A1}$ ，例如单元格直接引用自身。更隐蔽的形式是通过间接引用，例如在上面的示例中指定 $\mathbf{A1}{=}\mathbf{E1}$ ，这会在图形中引入一个循环。

Before assigning a new formula to a cell, you should traverse the graph of dependencies to ensure it will not create a cycle. Consider if the user tried to set $\mathbf{A1}{=}\ \mathbf{F1} + \mathbf{E1}$ in the above graph. The two cells directly referenced by the formula are $\mathbf{F1}$ and $\mathbf{E1}$. We examine the incoming dependencies for $\mathbf{F1}$ and find none because it has no cells on which it depends, so this will be no problem. Next, we examine the arcs leading to E1, and find that it directly depends on $\mathbf{C1}$ and $\mathbf{D1}$. So far so good, but when we continue our recursive

在为单元格分配新公式之前，应遍历依赖关系图，以确保不会产生循环。假设用户试图在上图中设置 $\mathbf{A1}{=}\ \mathbf{F1} + \mathbf{E1}$ 。公式直接引用的两个单元格是 $\mathbf{F1}$ 和 $\mathbf{E1}$ 。我们检查了 $\mathbf{F1}$ 的输入依赖关系，没有发现任何依赖关系，因为它不依赖于任何单元格，所以不会有问题。接下来，我们检查通向 E1 的弧，发现它直接依赖于 $\mathbf{C1}$ 和 $\mathbf{D1}$ 。到目前为止一切顺利，但是当我们继续递归

traversal to find the cells they depend on, we eventually run into $\mathbf{A1}$, which is exactly what we didn't want to find. The formula is disallowed because it is circular.

遍历以查找它们所依赖的单元格时，我们最终会遇到 $\mathbf{A1}$ ，这正是我们不想找到的。这个公式是不允许的，因为它是循环的。

Hints and requirements for ssmodel:

ssmodel的提示和要求：

When setting a cell, first check for problems (invalid name, parsing issues, circular reference, etc.) and if any are found, discard the formula and leave the cell unchanged. It's best to do all the checks before making any changes, so you don't have to undo it halfway.

设置单元格时，首先检查是否存在问题（无效名称、解析问题、循环引用等），如果发现问题，则放弃公式，保留单元格不变。最好在进行任何更改之前进行所有检查，这样就不必中途撤销。

The file format used for the load and save commands is a simple text file containing a list of all non-empty cells, one cell per line. Here is an excerpt from the file displayed on page 2 :

加载和保存命令使用的文件格式是一个简单的文本文件，包含所有非空单元格的列表，每行一个单元格。下面是第 2 页显示的文件节选：

```
A2 = "Partridge"
B2 = 1
C2 = 129.99
D2 = B2 * C2
```

The ability to load and save files will be an invaluable time-saver for your testing. We provide some sample saved files in the starting project as test cases. You may assume that the contents of files being loaded are well

formed, unlike the user's typo-ridden input, which must be gracefully handled at the command line.

加载和保存文件的功能将为您的测试节省宝贵的时间。我们在起始项目中提供了一些保存文件的示例作为测试用例。您可以假定加载的文件内容是完整的，而不像用户输入的错别字，后者必须在命令行中妥善处理。

We suggest getting basic cell assignment, display, load/save, etc. all working without tracking dependencies first. Handle dependencies only after the underlying infrastructure is implemented and debugged.

我们建议在不跟踪依赖关系的情况下，先让基本的单元格分配、显示、加载/保存等功能全部运行起来。只有在底层基础架构实施和调试完成后，才能处理依赖关系。

You're free to represent the dependencies in any way you like (using pointers, sets, vectors, etc). You'll find that you need both outgoing dependencies, i.e., those cells that must be notified when this one changes, and incoming dependencies, i.e. which cells when changed require this one to update. The incoming arcs are just the outgoing arcs reversed, but it'll be better if you store them in such a way to enable easy access in either direction. This little bit of redundancy makes some tasks simpler to manage.

您可以用任何您喜欢的方式来表示依赖关系（使用指针、集合、矢量等）。您会发现既需要出站依赖关系，即当这个单元格发生变化时必须通知的单元格，也需要入站依赖关系，即当哪个单元格发生变化时需要更新这个单元格。入库弧线只是出库弧线的反向，但如果能以方便从任一方向访问的方式存储，效果会更好。这一点点冗余会让某些任务的管理变得更简单。

Make sure you have a basic understanding of graphs and graph traversals as detailed in Chapter 18. We recommend drawing pictures to visualize. Accidentally introducing cycles into the graph will create opportunities for infinite recursion or iteration, so be extra careful to avoid them.

请确保您对图形和图形遍历有基本的了解，详见第 18 章。我们建议您绘制图片，以便直观地理解。如果不小心在图形中引入循环，就会产生无限递归或迭代的机会，因此要格外小心避免。

The SSModel class has much more room for design decisions than previous assignments. We provide some starting suggestions, but much of how you structure things is up to you. You will likely find yourself wrestling with various decisions and tradeoffs. There is no definitive "right" way, but there are better and worse choices. Part of your job is brainstorming your options, making thoughtful decisions, and documenting your reasoning. Taking the time to make good choices early in the design phase can significantly pay off during implementation. You are strongly encouraged to ask us for advice along the way-remember it is far easier to

与以前的作业相比，SSModel 类的设计决策空间要大得多。我们提供了一些起步建议，但如何安排结构在很大程度上取决于你自己。你很可能会发现自己在各种决策和权衡中挣扎。没有绝对 "正确 "的方法，但有更好和更差的选择。你的工作之一就是集思广益，做出深思熟虑的决定，并记录你的理由。在设计阶段的早期就花时间做出正确的选择，可以在实施过程中获得显著的回报。我们强烈建议您在设计过程中向我们寻求建议，请记住，以下做法要容易得多

correct a questionable decision early in the process than when the code is further along.

在早期阶段纠正有问题的决定，比在代码编写过程中纠正有问题的决定更有效。

And don't forget these: 别忘了这些：

Your design may have two or more classes that each depend on each other. An Expression class uses an SSModel object that in turn stores Expression objects. Think about the trouble of having ssmodel. **h** include **exp. h** while exp. h tries to include ssmodel. **h**！In C++, the mechanism for dealing with this is the forward reference. At the top of the ssmodel .h file, before you declare the SSModel class interface, you can insert a forward reference to a class it depends on, such as Expression, with this bit of syntax:

您的设计可能有两个或多个相互依赖的类。Expression 类使用 SSModel 对象，而 SSModel 又存储 Expression 对象。想想拥有 SSModel 的麻烦吧。 **h** include **exp. h** 而 exp.h 试图包含 ssmodel。 **h**！ 在 C++ 中，处理这种情况的机制是前向引用。在 ssmodel .h 文件的顶部，在声明 SSModel 类接口之前，可以使用以下语法插入一个指向它所依赖的类的前向引用，例如 Expression：

class Expression; 类表达；

This forward reference informs the compiler that there will be a class named Expression. This allows the SSModel to happily continue on declaring data members and method parameter/return types that are of type Expression * since the compiler has been assured such a class will exist and will be around later.

这个前向引用通知编译器将有一个名为 Expression 的类。这样，SSModel 就可以继续愉快地声明 Expression 类型的数据成员和方法参数/返回类型 *，因为编译器已经确信会存在这样一个类，而且以后还会出现。

You are expected to properly dispose of any dynamically allocated memory. This is particularly tricky, because the locations of many dynamically allocated objects are aliased and held in multiple places, and it's a struggle to keep tabs on them so that everything is freed exactly one time!

你需要正确处理任何动态分配的内存。这一点尤为棘手，因为许多动态分配对象的位置都是别名的，并保存在多个地方，因此要对它们进行监控，以便一次性释放所有内存是非常困难的！

In some situations, you'll benefit from calling an Expression's getType method to decide whether or not it's safe to downcast (e.g. explicitly cast an Expression * to, say, a TextStringExp *) so that you're able to invoke a method specific to the subclass.

在某些情况下，调用 Expression 的 getType 方法来决定下投（例如，显式地将 Expression * 投到 TextStringExp *）是否安全，以便能够调用子类的特定方法，这将使您受益匪浅。

Good luck with the assignment, and we hope you enjoy it and appreciate it as something you couldn't possibly have built seven weeks ago!

祝你完成任务好运，我们希望你喜欢并欣赏它，因为它是你七周前不可能建造出来的！

---

[1] Those needing an extra late day can submit as late as Thursday, November 21st at 11：59pm. Those wishing the consume a second late day can submit as late as Friday, November 22nd at 11:59pm.

[1] 需要额外延迟一天的用户最迟可在 11 月 21st 日（星期四）11：59pm 时提交。希望再晚一天提交者，可在 11 月 22nd 星期五晚上 11:59 之前提交。