哈尔滨工业大学 计算学部

2023 年秋季学期《开源软件开发实践》

Lab3：自选开源项目编程

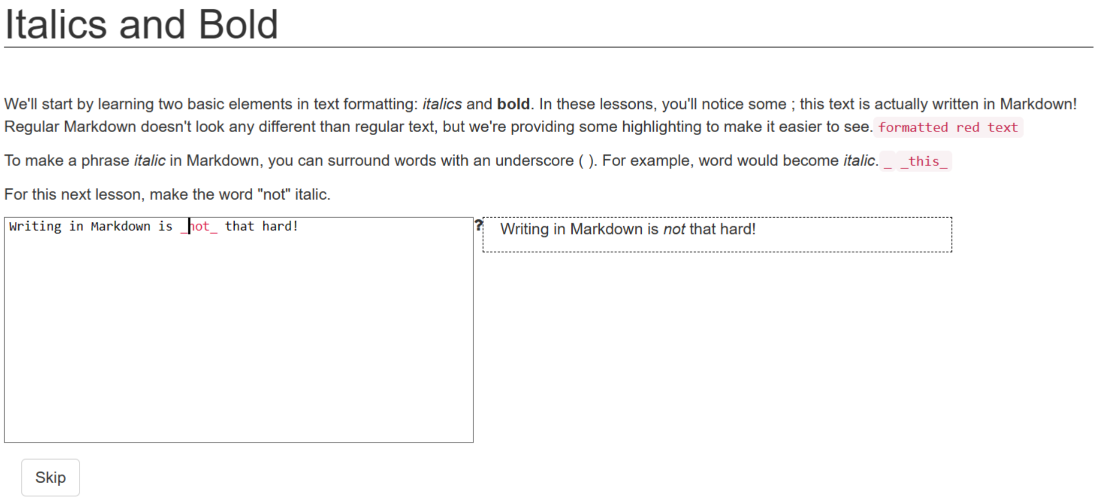
学号	姓名	联系方式
2023140022	雷文俊	514530706@qq.com/13394013296
2023140020	郭庆熠	1464265376@qq.com/18114361686
2021110906	周翔	1752515941@qq.com/13523973379

目 录

1 实验要求

能够根据开源项目提供的文档，对项目进行了解和分析
掌握基本的 Markdown 语法
实际参与某开源项目，贡献代码
本次实验由不超过 3 人的小组团队完成

2 实验内容 1 掌握基本 Markdown 语法



Italics and Bold

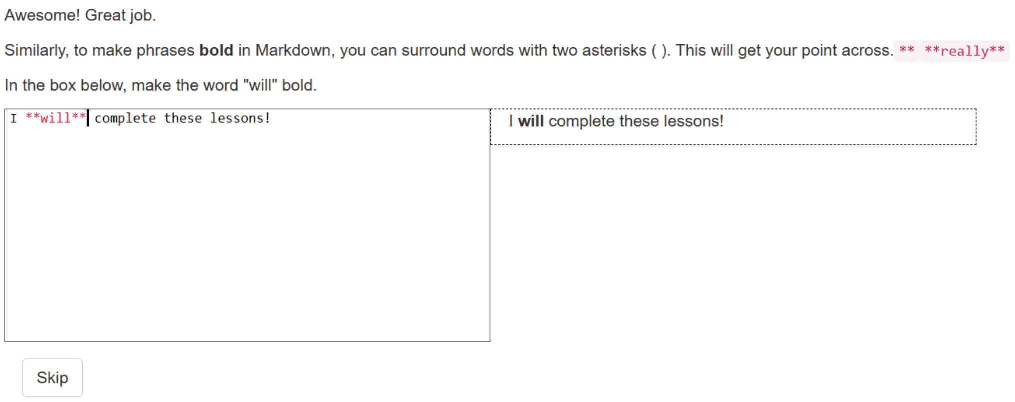
We'll start by learning two basic elements in text formatting: *italics* and **bold**. In these lessons, you'll notice some ; this text is actually written in Markdown! Regular Markdown doesn't look any different than regular text, but we're providing some highlighting to make it easier to see. `formatted red text`

To make a phrase *italic* in Markdown, you can surround words with an underscore ( ). For example, word would become *italic*. `_` `_this_`

For this next lesson, make the word "not" italic.

```
Writing in Markdown is _not_ that hard!
```

Writing in Markdown is *not* that hard!



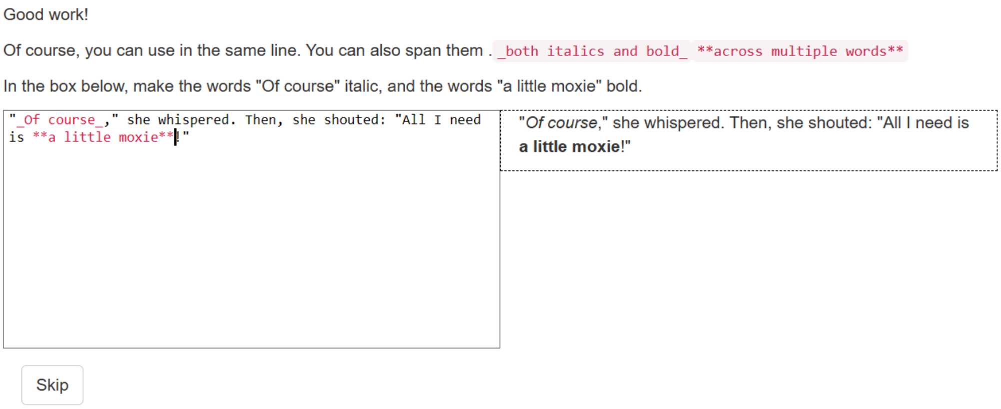
Awesome! Great job.

Similarly, to make phrases **bold** in Markdown, you can surround words with two asterisks ( ). This will get your point across. `**` `**really**`

In the box below, make the word "will" bold.

```
I **will** complete these lessons!
```

I **will** complete these lessons!



Good work!

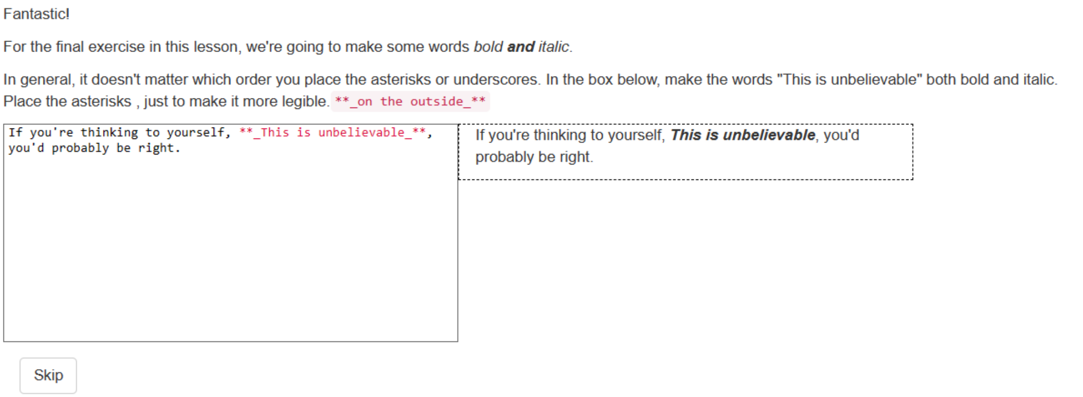
Of course, you can use in the same line. You can also span them . `_both italics and bold_` `**across multiple words**`

In the box below, make the words "Of course" italic, and the words "a little moxie" bold.

```
"_Of course_," she whispered. Then, she shouted: "All I need
is **a little moxie**!"
```

"*Of course*," she whispered. Then, she shouted: "All I need is **a little moxie**!"

Skip



Fantastic!

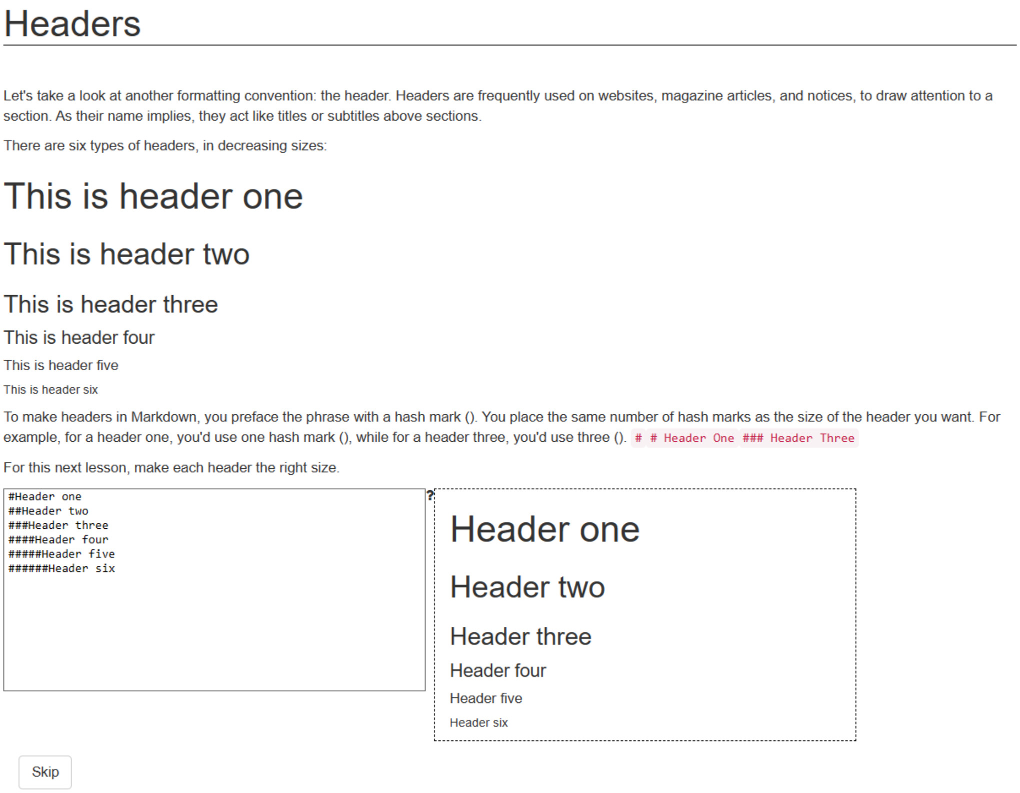
For the final exercise in this lesson, we're going to make some words *bold **and** italic*.

In general, it doesn't matter which order you place the asterisks or underscores. In the box below, make the words "This is unbelievable" both bold and italic. Place the asterisks , just to make it more legible. `**_on the outside_**`

```
If you're thinking to yourself, **_This is unbelievable_**,
you'd probably be right.
```

If you're thinking to yourself, ***This is unbelievable***, you'd probably be right.

Skip



Headers

Let's take a look at another formatting convention: the header. Headers are frequently used on websites, magazine articles, and notices, to draw attention to a section. As their name implies, they act like titles or subtitles above sections.

There are six types of headers, in decreasing sizes:

This is header one

This is header two

This is header three

This is header four

This is header five

This is header six

To make headers in Markdown, you preface the phrase with a hash mark (). You place the same number of hash marks as the size of the header you want. For example, for a header one, you'd use one hash mark (), while for a header three, you'd use three (). `# # Header One ### Header Three`

For this next lesson, make each header the right size.

```
#Header one
##Header two
###Header three
####Header four
#####Header five
######Header six
```

Header one

Header two

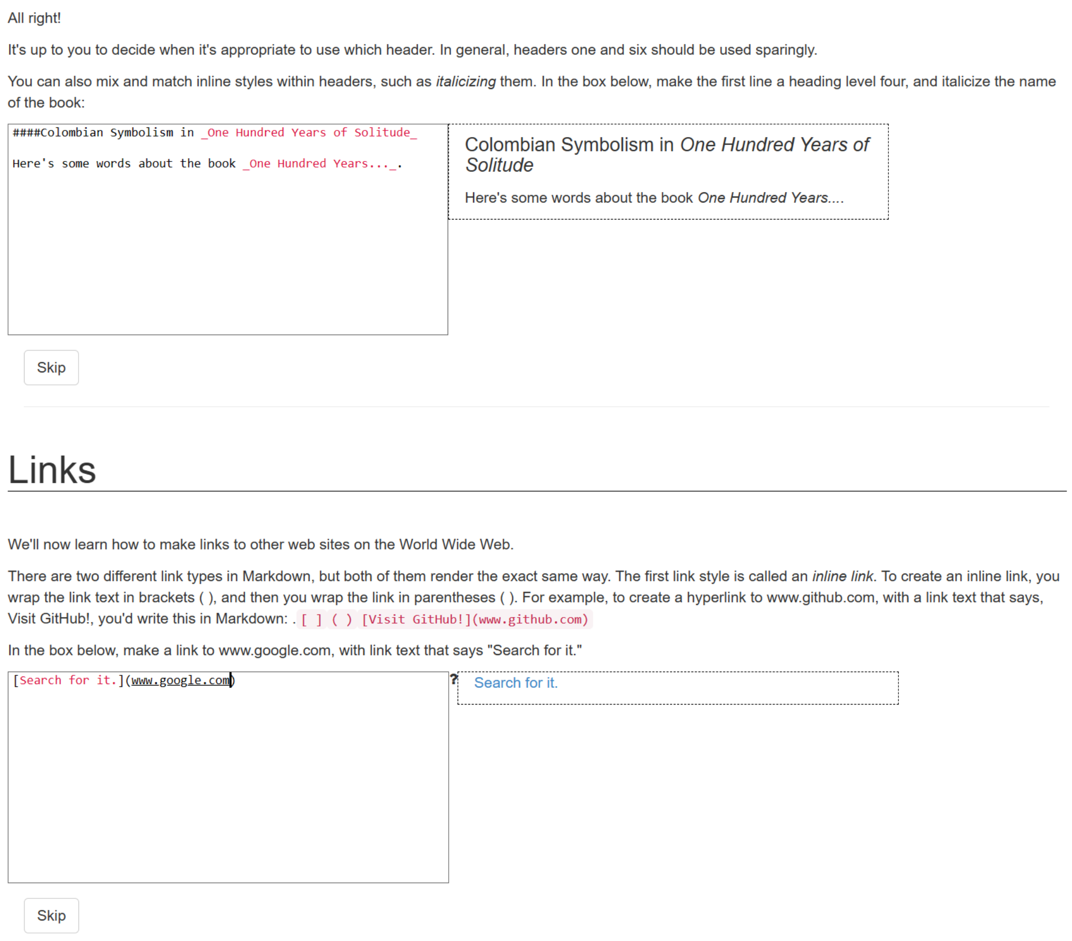
Header three

Header four

Header five

Header six

Skip



All right!

It's up to you to decide when it's appropriate to use which header. In general, headers one and six should be used sparingly.

You can also mix and match inline styles within headers, such as *italicizing* them. In the box below, make the first line a heading level four, and italicize the name of the book:

```
####Colombian Symbolism in _One Hundred Years of Solitude_

Here's some words about the book _One Hundred Years..._.
```

Colombian Symbolism in *One Hundred Years of Solitude*

Here's some words about the book *One Hundred Years...*.

Links

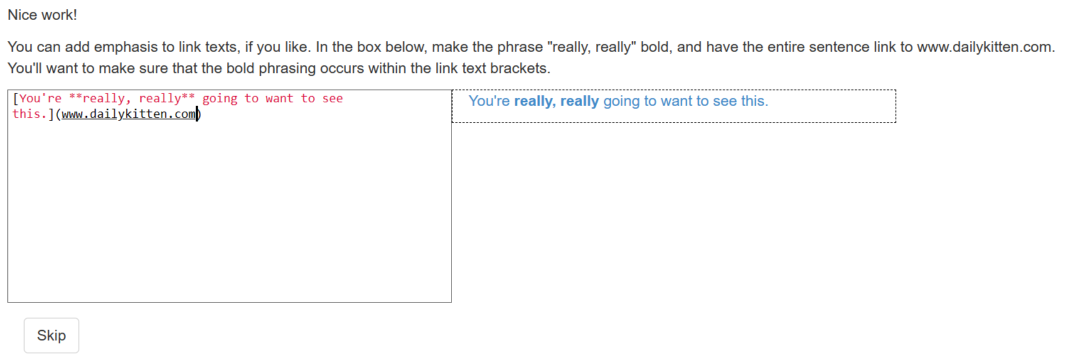
We'll now learn how to make links to other web sites on the World Wide Web.

There are two different link types in Markdown, but both of them render the exact same way. The first link style is called an *inline link*. To create an inline link, you wrap the link text in brackets ( ), and then you wrap the link in parentheses ( ). For example, to create a hyperlink to www.github.com, with a link text that says, Visit GitHub!, you'd write this in Markdown: . `[ ] ( ) [Visit GitHub!](www.github.com)`

In the box below, make a link to www.google.com, with link text that says "Search for it."

```
[Search for it.](www.google.com)
```

Search for it.

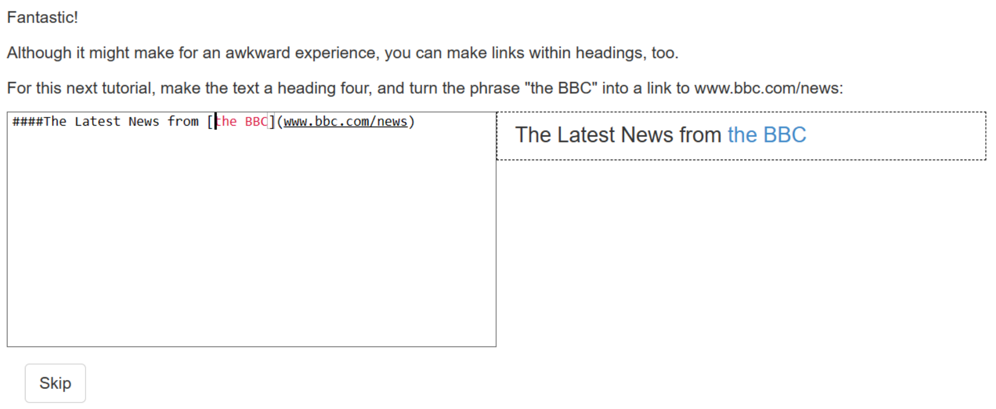


Nice work!

You can add emphasis to link texts, if you like. In the box below, make the phrase "really, really" bold, and have the entire sentence link to www.dailykitten.com. You'll want to make sure that the bold phrasing occurs within the link text brackets.

```
[You're **really, really** going to want to see
this.](www.dailykitten.com)
```

You're **really, really** going to want to see this.



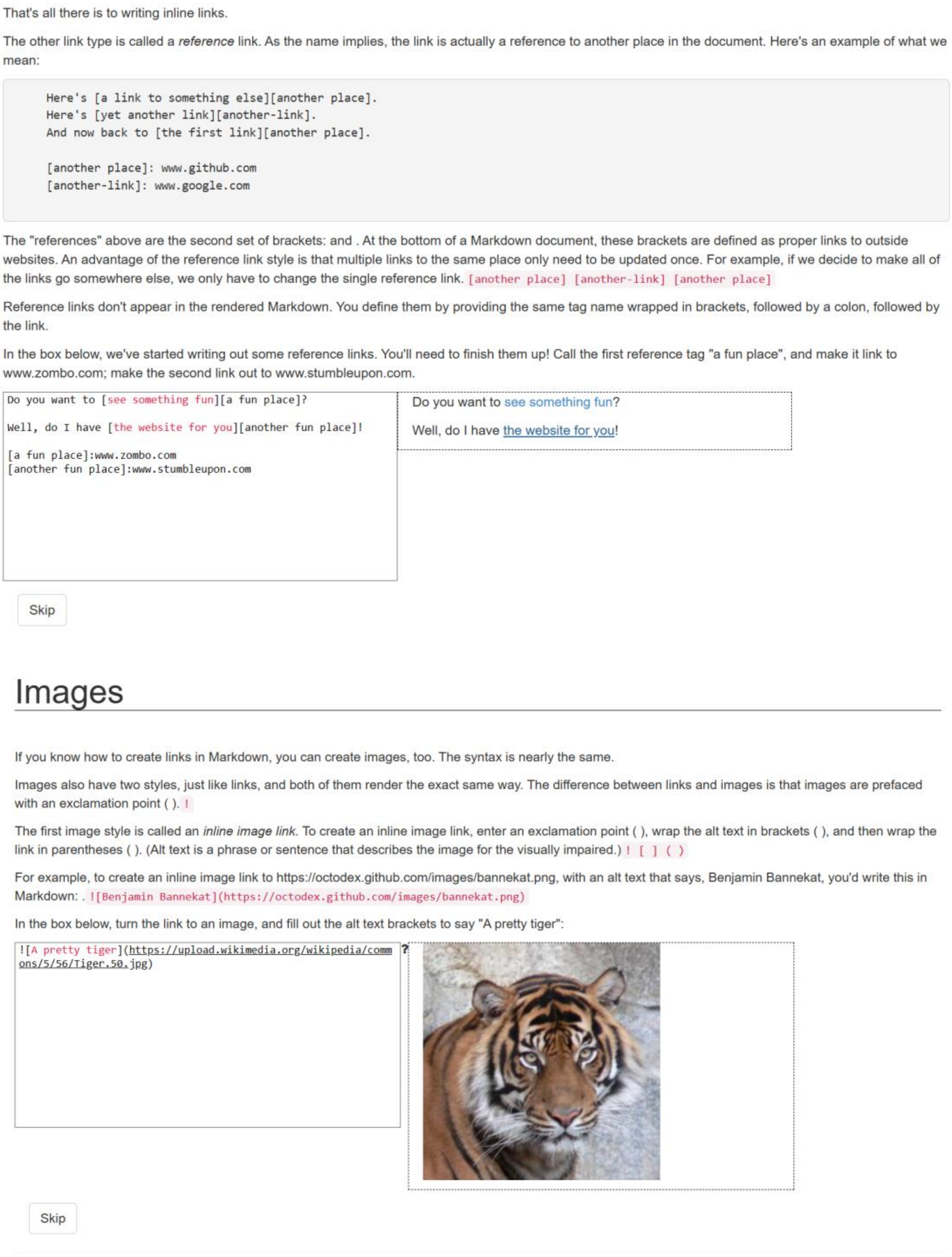
Fantastic!

Although it might make for an awkward experience, you can make links within headings, too.

For this next tutorial, make the text a heading four, and turn the phrase "the BBC" into a link to www.bbc.com/news:

```
####The Latest News from [the BBC](www.bbc.com/news)
```

The Latest News from the BBC



That's all there is to writing inline links.

The other link type is called a *reference* link. As the name implies, the link is actually a reference to another place in the document. Here's an example of what we mean:

```
Here's [a link to something else][another place].
Here's [yet another link][another-link].
And now back to [the first link][another place].

[another place]: www.github.com
[another-link]: www.google.com
```

The "references" above are the second set of brackets: and . At the bottom of a Markdown document, these brackets are defined as proper links to outside websites. An advantage of the reference link style is that multiple links to the same place only need to be updated once. For example, if we decide to make all of the links go somewhere else, we only have to change the single reference link. `[another place]` `[another-link]` `[another place]`

Reference links don't appear in the rendered Markdown. You define them by providing the same tag name wrapped in brackets, followed by a colon, followed by the link.

In the box below, we've started writing out some reference links. You'll need to finish them up! Call the first reference tag "a fun place", and make it link to www.zombo.com; make the second link out to www.stumbleupon.com.

```
Do you want to [see something fun][a fun place]?

Well, do I have [the website for you][another fun place]!

[a fun place]:www.zombo.com
[another fun place]:www.stumbleupon.com
```

Do you want to see something fun?

Well, do I have the website for you!

Images

If you know how to create links in Markdown, you can create images, too. The syntax is nearly the same.

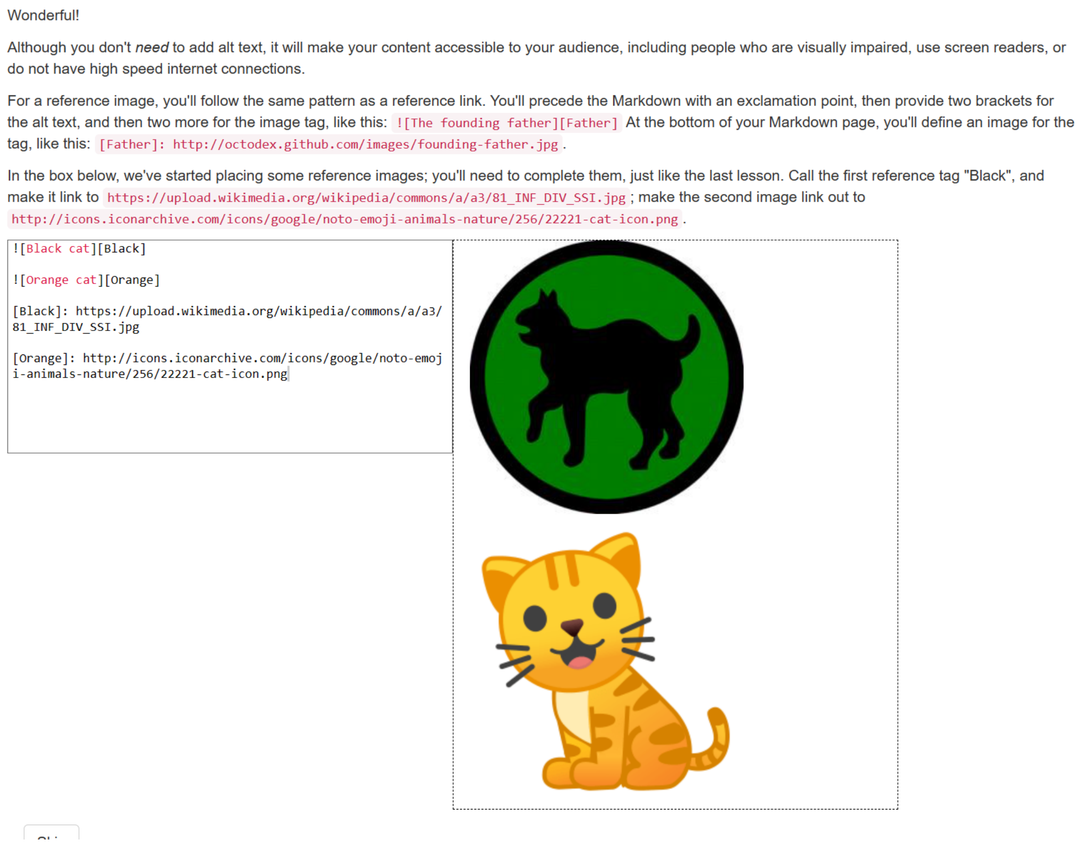
Images also have two styles, just like links, and both of them render the exact same way. The difference between links and images is that images are prefaced with an exclamation point ( ). `!`

The first image style is called an *inline image link*. To create an inline image link, enter an exclamation point ( ), wrap the alt text in brackets ( ), and then wrap the link in parentheses ( ). (Alt text is a phrase or sentence that describes the image for the visually impaired.) `!` `[ ]` `( )`

For example, to create an inline image link to https://octodex.github.com/images/bannekat.png, with an alt text that says, Benjamin Bannekat, you'd write this in Markdown: . `![Benjamin Bannekat](https://octodex.github.com/images/bannekat.png)`

In the box below, turn the link to an image, and fill out the alt text brackets to say "A pretty tiger":

```
![A pretty tiger](https://upload.wikimedia.org/wikipedia/comm
ons/5/56/Tiger.50.jpg)
```



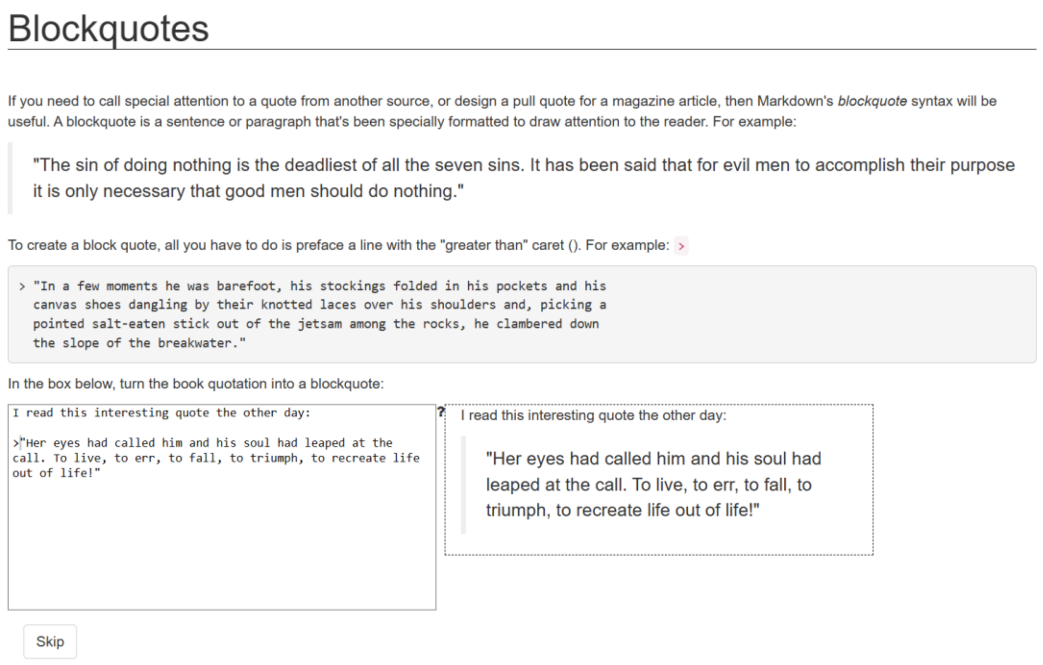
Wonderful!

Although you don't *need* to add alt text, it will make your content accessible to your audience, including people who are visually impaired, use screen readers, or do not have high speed internet connections.

For a reference image, you'll follow the same pattern as a reference link. You'll precede the Markdown with an exclamation point, then provide two brackets for the alt text, and then two more for the image tag, like this: `![The founding father][Father]` At the bottom of your Markdown page, you'll define an image for the tag, like this: `[Father]: http://octodex.github.com/images/founding-father.jpg` .

In the box below, we've started placing some reference images; you'll need to complete them, just like the last lesson. Call the first reference tag "Black", and make it link to `https://upload.wikimedia.org/wikipedia/commons/a/a3/81_INF_DIV_SSI.jpg` ; make the second image link out to `http://icons.iconarchive.com/icons/google/noto-emoji-animals-nature/256/22221-cat-icon.png` .

```
![Black cat][Black]

![Orange cat][Orange]

[Black]: https://upload.wikimedia.org/wikipedia/commons/a/a3/
81_INF_DIV_SSI.jpg

[Orange]: http://icons.iconarchive.com/icons/google/noto-emoj
i-animals-nature/256/22221-cat-icon.png
```



Blockquotes

If you need to call special attention to a quote from another source, or design a pull quote for a magazine article, then Markdown's *blockquote* syntax will be useful. A blockquote is a sentence or paragraph that's been specially formatted to draw attention to the reader. For example:

> "The sin of doing nothing is the deadliest of all the seven sins. It has been said that for evil men to accomplish their purpose it is only necessary that good men should do nothing."

To create a block quote, all you have to do is preface a line with the "greater than" caret (). For example: `>`

```
> "In a few moments he was barefoot, his stockings folded in his pockets and his
  canvas shoes dangling by their knotted laces over his shoulders and, picking a
  pointed salt-eaten stick out of the jetsam among the rocks, he clambered down
  the slope of the breakwater."
```

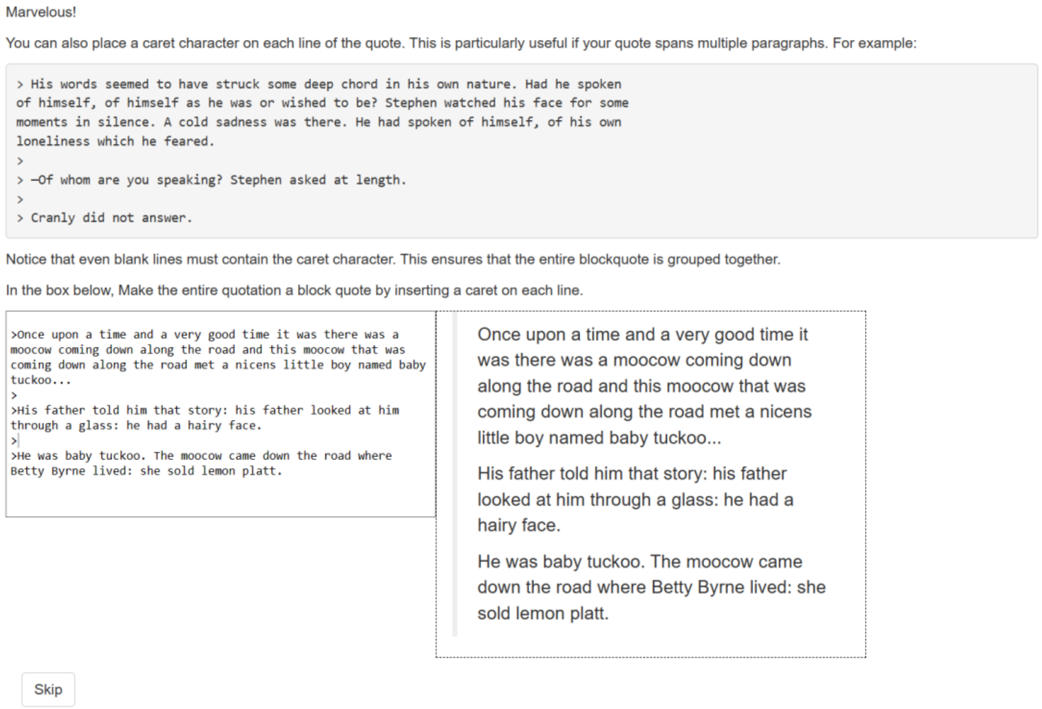
In the box below, turn the book quotation into a blockquote:

```
I read this interesting quote the other day:

>"Her eyes had called him and his soul had leaped at the
call. To live, to err, to fall, to triumph, to recreate life
out of life!"
```

?

I read this interesting quote the other day:

> "Her eyes had called him and his soul had leaped at the call. To live, to err, to fall, to triumph, to recreate life out of life!"



Marvelous!

You can also place a caret character on each line of the quote. This is particularly useful if your quote spans multiple paragraphs. For example:

```
> His words seemed to have struck some deep chord in his own nature. Had he spoken
of himself, of himself as he was or wished to be? Stephen watched his face for some
moments in silence. A cold sadness was there. He had spoken of himself, of his own
loneliness which he feared.
>
> —Of whom are you speaking? Stephen asked at length.
>
> Cranly did not answer.
```

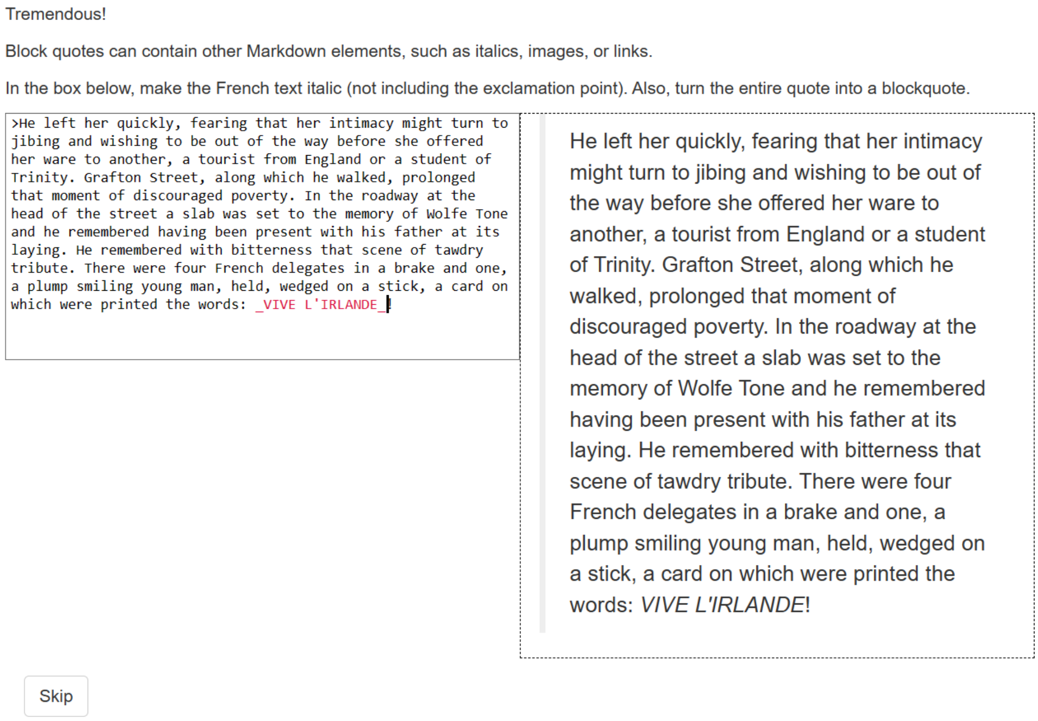
Notice that even blank lines must contain the caret character. This ensures that the entire blockquote is grouped together.

In the box below, Make the entire quotation a block quote by inserting a caret on each line.

```
>Once upon a time and a very good time it was there was a
moocow coming down along the road and this moocow that was
coming down along the road met a nicens little boy named baby
tuckoo...
>
>His father told him that story: his father looked at him
through a glass: he had a hairy face.
>
>He was baby tuckoo. The moocow came down the road where
Betty Byrne lived: she sold lemon platt.
```

> Once upon a time and a very good time it was there was a moocow coming down along the road and this moocow that was coming down along the road met a nicens little boy named baby tuckoo...
>
> His father told him that story: his father looked at him through a glass: he had a hairy face.
>
> He was baby tuckoo. The moocow came down the road where Betty Byrne lived: she sold lemon platt.

Skip



Tremendous!

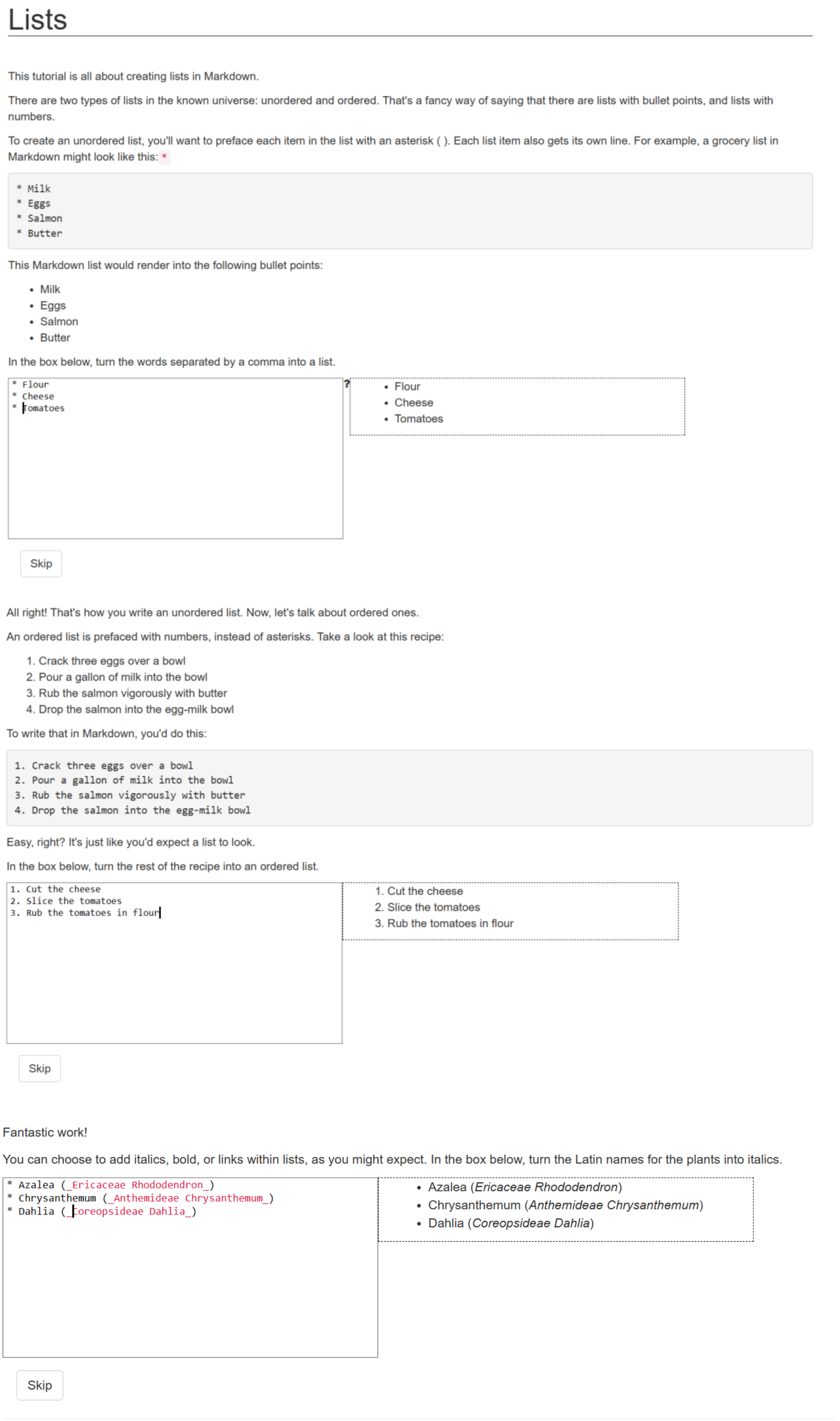
Block quotes can contain other Markdown elements, such as italics, images, or links.

In the box below, make the French text italic (not including the exclamation point). Also, turn the entire quote into a blockquote.

```
>He left her quickly, fearing that her intimacy might turn to
jibing and wishing to be out of the way before she offered
her ware to another, a tourist from England or a student of
Trinity. Grafton Street, along which he walked, prolonged
that moment of discouraged poverty. In the roadway at the
head of the street a slab was set to the memory of Wolfe Tone
and he remembered having been present with his father at its
laying. He remembered with bitterness that scene of tawdry
tribute. There were four French delegates in a brake and one,
a plump smiling young man, held, wedged on a stick, a card on
which were printed the words: _VIVE L'IRLANDE_!
```

> He left her quickly, fearing that her intimacy might turn to jibing and wishing to be out of the way before she offered her ware to another, a tourist from England or a student of Trinity. Grafton Street, along which he walked, prolonged that moment of discouraged poverty. In the roadway at the head of the street a slab was set to the memory of Wolfe Tone and he remembered having been present with his father at its laying. He remembered with bitterness that scene of tawdry tribute. There were four French delegates in a brake and one, a plump smiling young man, held, wedged on a stick, a card on which were printed the words: *VIVE L'IRLANDE*!

Skip



Lists

This tutorial is all about creating lists in Markdown.

There are two types of lists in the known universe: unordered and ordered. That's a fancy way of saying that there are lists with bullet points, and lists with numbers.

To create an unordered list, you'll want to preface each item in the list with an asterisk ( ). Each list item also gets its own line. For example, a grocery list in Markdown might look like this: *

```
* Milk
* Eggs
* Salmon
* Butter
```

This Markdown list would render into the following bullet points:

- Milk
- Eggs
- Salmon
- Butter

In the box below, turn the words separated by a comma into a list.

```
* Flour
* Cheese
* Tomatoes
```

- Flour
- Cheese
- Tomatoes

All right! That's how you write an unordered list. Now, let's talk about ordered ones.

An ordered list is prefaced with numbers, instead of asterisks. Take a look at this recipe:

1. Crack three eggs over a bowl
2. Pour a gallon of milk into the bowl
3. Rub the salmon vigorously with butter
4. Drop the salmon into the egg-milk bowl

To write that in Markdown, you'd do this:

```
1. Crack three eggs over a bowl
2. Pour a gallon of milk into the bowl
3. Rub the salmon vigorously with butter
4. Drop the salmon into the egg-milk bowl
```

Easy, right? It's just like you'd expect a list to look.

In the box below, turn the rest of the recipe into an ordered list.

```
1. Cut the cheese
2. Slice the tomatoes
3. Rub the tomatoes in flour
```

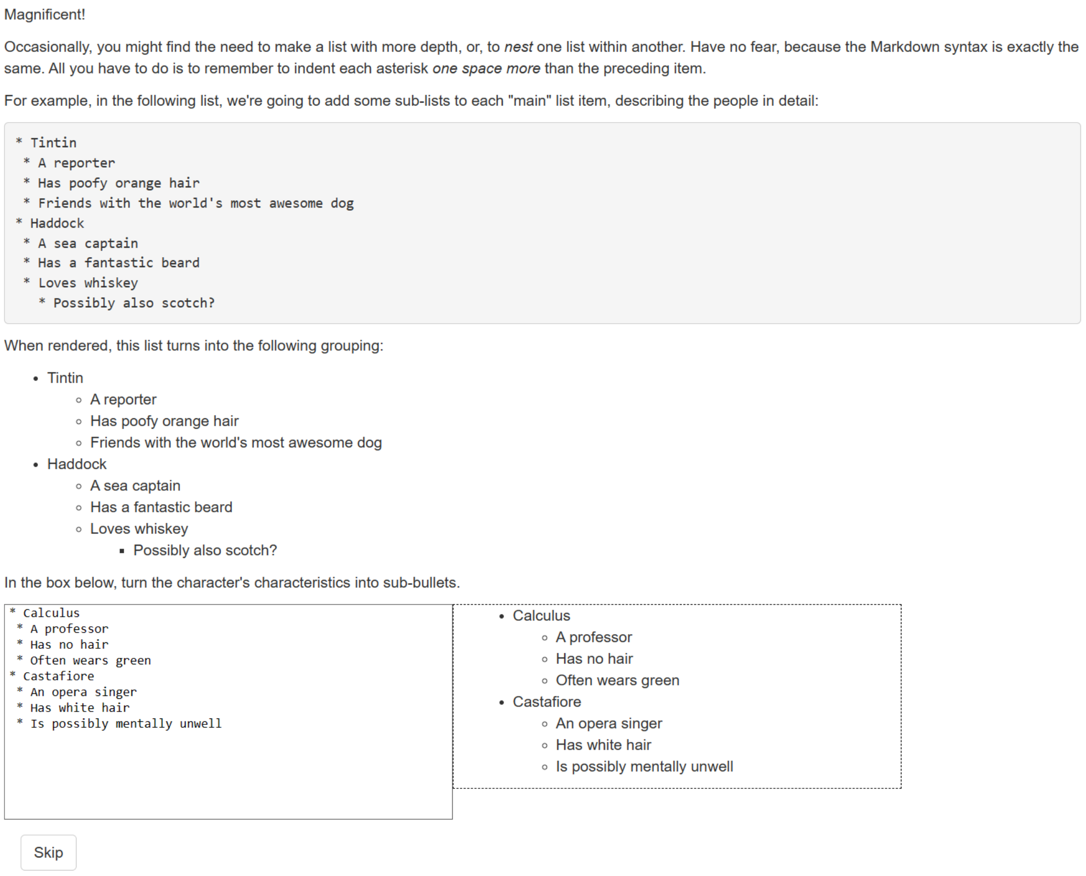
1. Cut the cheese
2. Slice the tomatoes
3. Rub the tomatoes in flour

Fantastic work!

You can choose to add italics, bold, or links within lists, as you might expect. In the box below, turn the Latin names for the plants into italics.

```
* Azalea (_Ericaceae Rhododendron_)
* Chrysanthemum (_Anthemideae Chrysanthemum_)
* Dahlia (_Coreopsideae Dahlia_)
```

- Azalea (*Ericaceae Rhododendron*)
- Chrysanthemum (*Anthemideae Chrysanthemum*)
- Dahlia (*Coreopsideae Dahlia*)



Magnificent!

Occasionally, you might find the need to make a list with more depth, or, to *nest* one list within another. Have no fear, because the Markdown syntax is exactly the same. All you have to do is to remember to indent each asterisk *one space more* than the preceding item.

For example, in the following list, we're going to add some sub-lists to each "main" list item, describing the people in detail:

```
* Tintin
 * A reporter
 * Has poofy orange hair
 * Friends with the world's most awesome dog
* Haddock
 * A sea captain
 * Has a fantastic beard
 * Loves whiskey
   * Possibly also scotch?
```

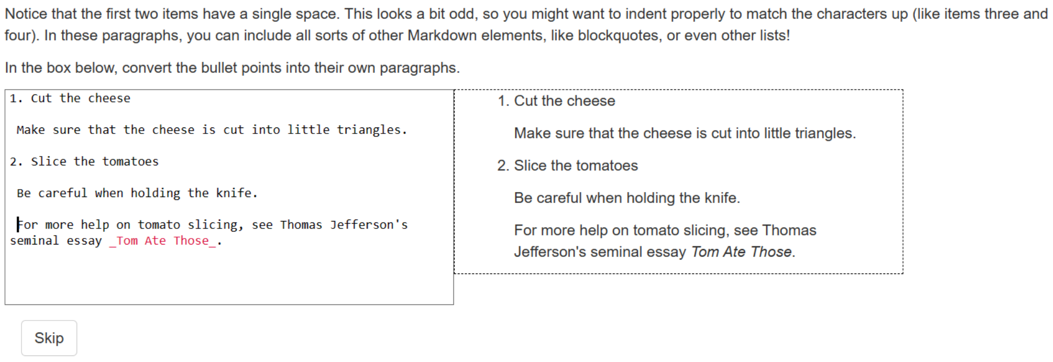
When rendered, this list turns into the following grouping:

- Tintin
  - A reporter
  - Has poofy orange hair
  - Friends with the world's most awesome dog
- Haddock
  - A sea captain
  - Has a fantastic beard
  - Loves whiskey
    - Possibly also scotch?

In the box below, turn the character's characteristics into sub-bullets.

```
* Calculus
 * A professor
 * Has no hair
 * Often wears green
* Castafiore
 * An opera singer
 * Has white hair
 * Is possibly mentally unwell
```

- Calculus
  - A professor
  - Has no hair
  - Often wears green
- Castafiore
  - An opera singer
  - Has white hair
  - Is possibly mentally unwell



Notice that the first two items have a single space. This looks a bit odd, so you might want to indent properly to match the characters up (like items three and four). In these paragraphs, you can include all sorts of other Markdown elements, like blockquotes, or even other lists!

In the box below, convert the bullet points into their own paragraphs.

```
1. Cut the cheese

 Make sure that the cheese is cut into little triangles.

2. Slice the tomatoes

 Be careful when holding the knife.

 For more help on tomato slicing, see Thomas Jefferson's
seminal essay _Tom Ate Those_.
```

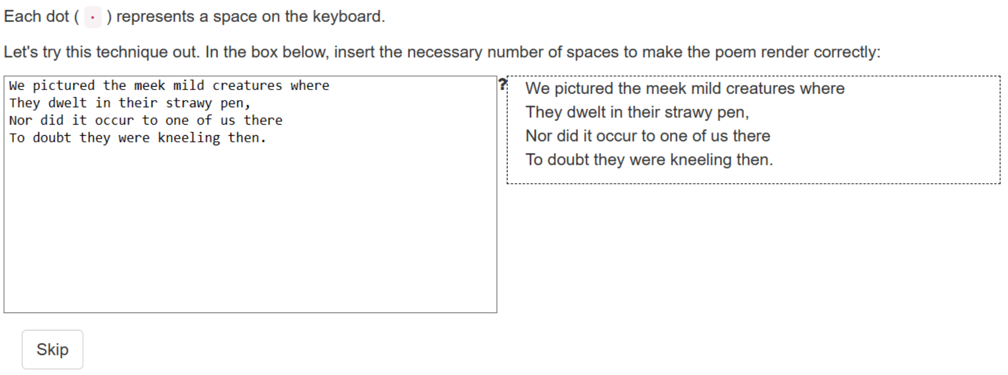
1. Cut the cheese

   Make sure that the cheese is cut into little triangles.

2. Slice the tomatoes

   Be careful when holding the knife.

   For more help on tomato slicing, see Thomas Jefferson's seminal essay *Tom Ate Those*.

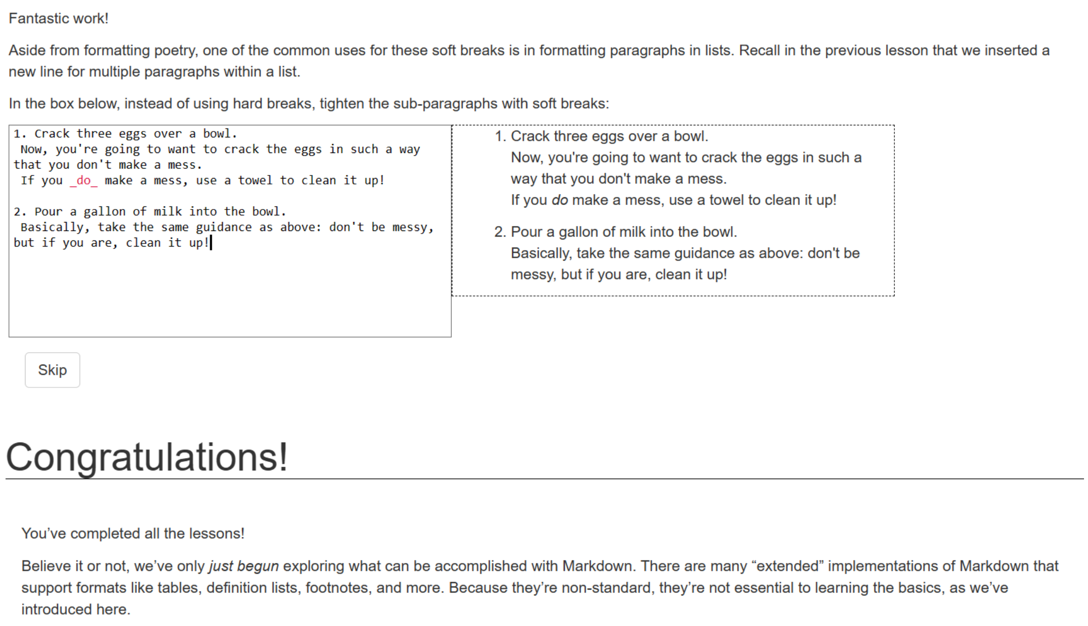


Each dot (·) represents a space on the keyboard.

Let's try this technique out. In the box below, insert the necessary number of spaces to make the poem render correctly:

```
We pictured the meek mild creatures where
They dwelt in their strawy pen,
Nor did it occur to one of us there
To doubt they were kneeling then.
```

We pictured the meek mild creatures where
They dwelt in their strawy pen,
Nor did it occur to one of us there
To doubt they were kneeling then.



Fantastic work!

Aside from formatting poetry, one of the common uses for these soft breaks is in formatting paragraphs in lists. Recall in the previous lesson that we inserted a new line for multiple paragraphs within a list.

In the box below, instead of using hard breaks, tighten the sub-paragraphs with soft breaks:

```
1. Crack three eggs over a bowl.
 Now, you're going to want to crack the eggs in such a way
that you don't make a mess.
 If you _do_ make a mess, use a towel to clean it up!

2. Pour a gallon of milk into the bowl.
 Basically, take the same guidance as above: don't be messy,
but if you are, clean it up!
```

1. Crack three eggs over a bowl.
   Now, you're going to want to crack the eggs in such a way that you don't make a mess.
   If you *do* make a mess, use a towel to clean it up!

2. Pour a gallon of milk into the bowl.
   Basically, take the same guidance as above: don't be messy, but if you are, clean it up!

Congratulations!

You've completed all the lessons!

Believe it or not, we've only *just begun* exploring what can be accomplished with Markdown. There are many “extended” implementations of Markdown that support formats like tables, definition lists, footnotes, and more. Because they're non-standard, they're not essential to learning the basics, as we've introduced here.

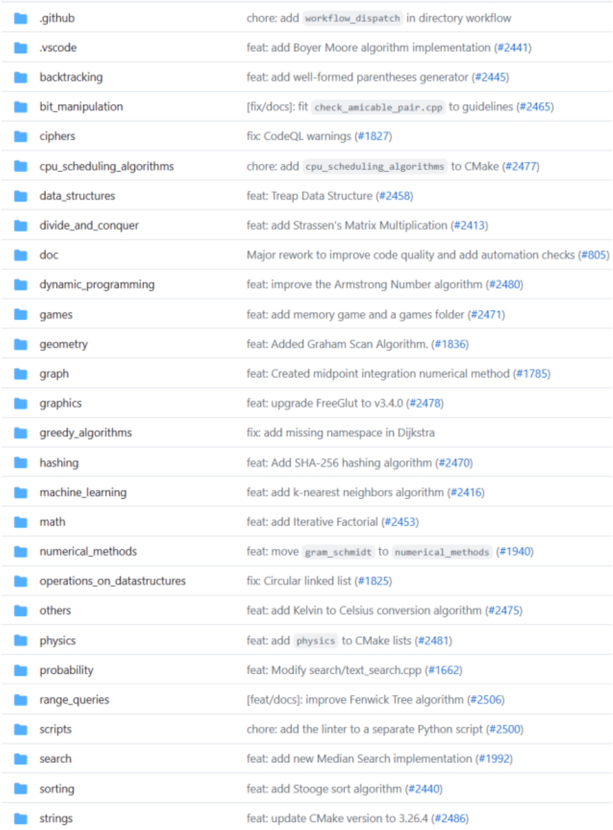
3 实验内容 2 参与开源项目

3.1 分析所选项目

项目地址：https://github.com/TheAlgorithms/C-Plus-Plus

该项目的开发目标是：该存储库是用 C++ 实现并在 MIT 许可证下许可的各种算法的开源实现的集合。这些算法涵盖了计算机科学、数学和统计学、数据科学、机器学习、工程等各种主题。这些实现和相关文档旨在为教师和学生提供学习资源。因此，人们可能会发现针对同一目标但使用不同算法策略和优化的多个实现。

项目结构如下图



.github	chore: add `workflow_dispatch` in directory workflow
.vscode	feat: add Boyer Moore algorithm implementation (#2441)
backtracking	feat: add well-formed parentheses generator (#2445)
bit_manipulation	[fix/docs]: fit `check_amicable_pair.cpp` to guidelines (#2465)
ciphers	fix: CodeQL warnings (#1827)
cpu_scheduling_algorithms	chore: add `cpu_scheduling_algorithms` to CMake (#2477)
data_structures	feat: Treap Data Structure (#2458)
divide_and_conquer	feat: add Strassen's Matrix Multiplication (#2413)
doc	Major rework to improve code quality and add automation checks (#805)
dynamic_programming	feat: improve the Armstrong Number algorithm (#2480)
games	feat: add memory game and a games folder (#2471)
geometry	feat: Added Graham Scan Algorithm. (#1836)
graph	feat: Created midpoint integration numerical method (#1785)
graphics	feat: upgrade FreeGlut to v3.4.0 (#2478)
greedy_algorithms	fix: add missing namespace in Dijkstra
hashing	feat: Add SHA-256 hashing algorithm (#2470)
machine_learning	feat: add k-nearest neighbors algorithm (#2416)
math	feat: add Iterative Factorial (#2453)
numerical_methods	feat: move `gram_schmidt` to `numerical_methods` (#1940)
operations_on_datastructures	fix: Circular linked list (#1825)
others	feat: add Kelvin to Celsius conversion algorithm (#2475)
physics	feat: add `physics` to CMake lists (#2481)
probability	feat: Modify search/text_search.cpp (#1662)
range_queries	[feat/docs]: improve Fenwick Tree algorithm (#2506)
scripts	chore: add the linter to a separate Python script (#2500)
search	feat: add new Median Search implementation (#1992)
sorting	feat: add Stooge sort algorithm (#2440)
strings	feat: update CMake version to 3.26.4 (#2486)

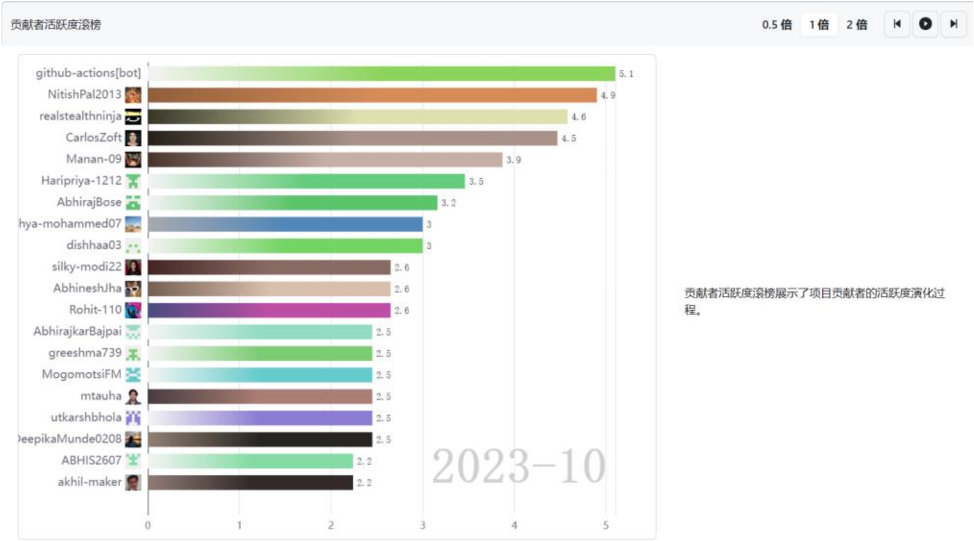
项目遵守的开源协议为：MIT 协议

项目特征为：该存储库以最基本的通用语言之一 C++ 提供了各种算法的实现。

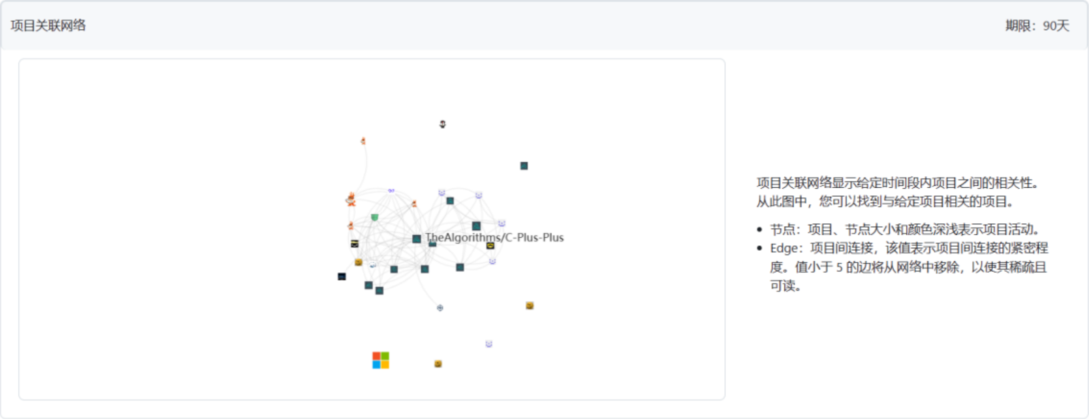
有据可查的源代码和详细的解释为教育工作者和学生提供了宝贵的资源。

每个源代码都是使用 STL 类的原子代码，编译和执行不需要*外部库*。因此，可以深入研究算法的基本原理。分别使用 MSVC 19 2022、AppleClang 14.0.0 和 GNU 11.3.0 在三个主要操作系统（即 Windows、MacOS 和 Ubuntu （Linux））的最新版本上编译和测试源代码。严格遵守 C++11 标准，确保代码可移植到嵌入式系统以及 ESP32、ARM Cortex 等，几乎没有变化。程序中的自检功能可确保自信地正确实施。模块化实现和开源许可使这些功能能够在其他应用程序中方便地使用。

项目参与者人数和活跃度为：



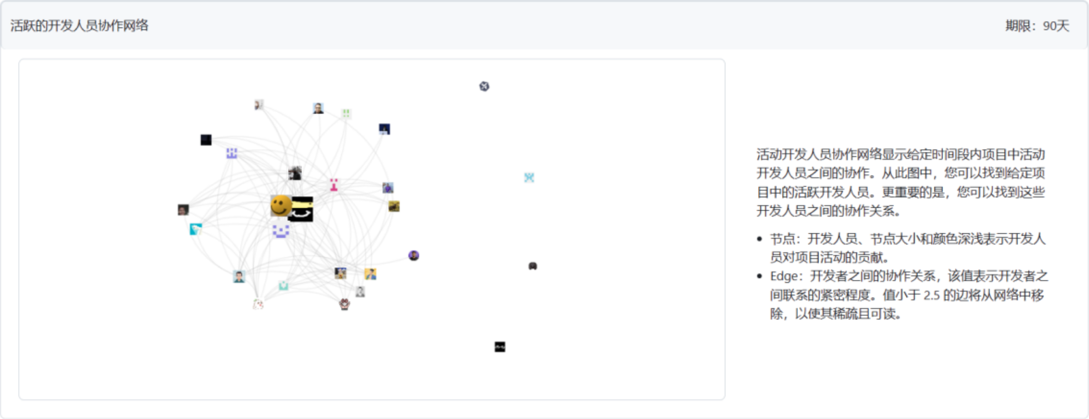
贡献者活跃度滚榜展示了项目贡献者的活跃度演化过程。


项目关联网络

期限：90天

项目关联网络显示给定时间段内项目之间的相关性。从此图中，您可以找到与给定项目相关的项目。

- 节点：项目、节点大小和颜色深浅表示项目活动。
- Edge：项目间连接，该值表示项目间连接的紧密程度。值小于 5 的边将从网络中移除，以使其稀疏且可读。

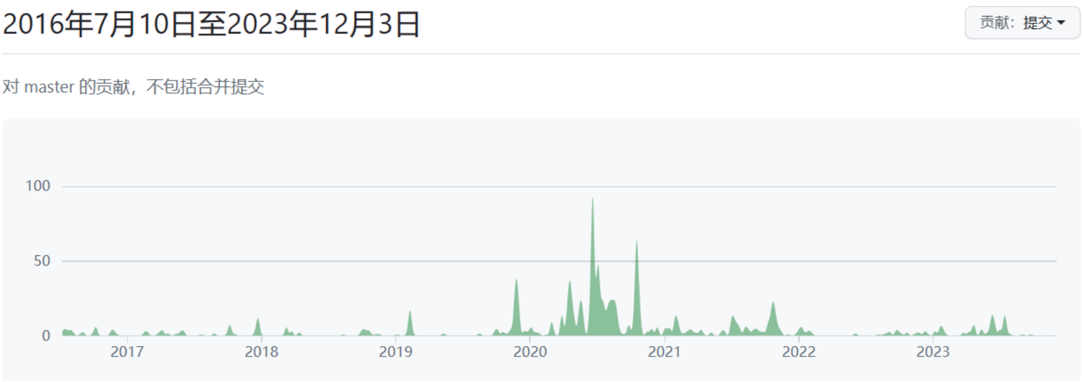

活跃的开发人员协作网络

期限：90天

活动开发人员协作网络显示给定时间段内项目中活动开发人员之间的协作。从此图中，您可以找到给定项目中的活跃开发人员。更重要的是，您可以找到这些开发人员之间的协作关系。

- 节点：开发人员、节点大小和颜色深浅表示开发人员对项目活动的贡献。
- Edge：开发者之间的协作关系，该值表示开发者之间联系的紧密程度。值小于 2.5 的边将从网络中移除，以使其稀疏且可读。

项目的活跃度为：


2016年7月10日至2023年12月3日

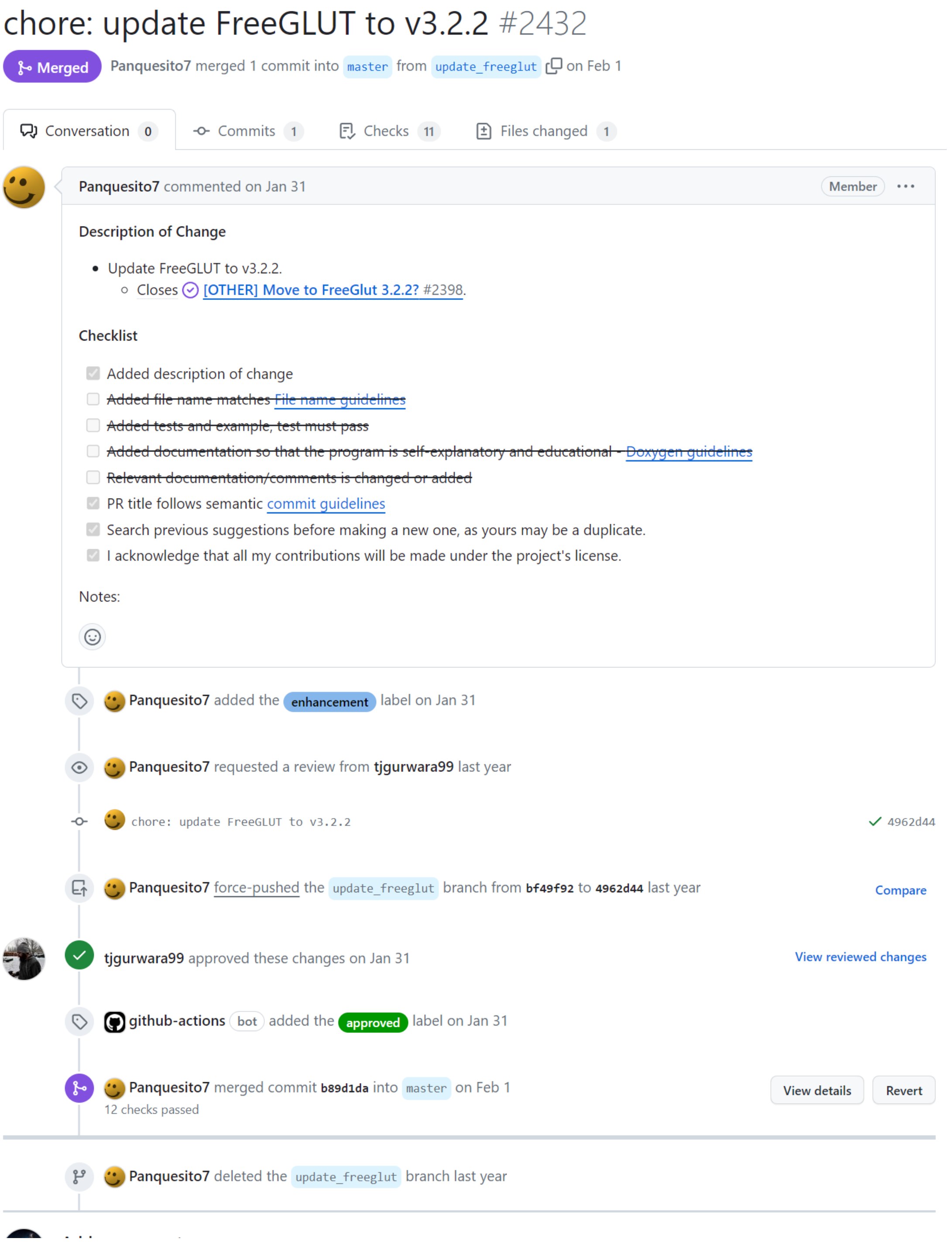
对 master 的贡献，不包括合并提交

3.2 项目参与者贡献过程分析

项目参与贡献的要求：该仓库实现和相关文档旨在为教育工作者和学生提供学习资源。因此，人们可能会发现针对同一目标的不止一种实现，但使用不同的算法策略和优化。要求每个源代码都是使用 STL 类的原子代码，编译和执行不需要外部库。严格遵守 C++11 标准可确保代码无需任何更改即可移植到嵌入式系统等。程序内的自检可确保正确实施并充满信心。模块化实现和开源许可使这些功能可以在其他应用程序中方便地使用。同时相关

文档/注释已更改或添加。

如图是一个已经完成的 issue 中用户 Panquesito7 进行的一个 Pull Request，他希望可以更新 FreeGlut 3.2.1 到 3.2.1，仓库管理者 tjgurwara99 批准了这个请求并合并了 Pull Request



chore: update FreeGLUT to v3.2.2 #2432

Merged　Panquesito7 merged 1 commit into `master` from `update_freeglut` on Feb 1

Panquesito7 commented on Jan 31　Member

Description of Change

- Update FreeGLUT to v3.2.2.
  - Closes [OTHER] Move to FreeGlut 3.2.2? #2398.

Checklist

☑ Added description of change
□ ~~Added file name matches File name guidelines~~
□ ~~Added tests and example, test must pass~~
□ ~~Added documentation so that the program is self-explanatory and educational - Doxygen guidelines~~
□ ~~Relevant documentation/comments is changed or added~~
☑ PR title follows semantic commit guidelines
☑ Search previous suggestions before making a new one, as yours may be a duplicate.
☑ I acknowledge that all my contributions will be made under the project's license.

Notes:

Panquesito7 added the enhancement label on Jan 31

Panquesito7 requested a review from tjgurwara99 last year

`chore: update FreeGLUT to v3.2.2`　✓ 4962d44

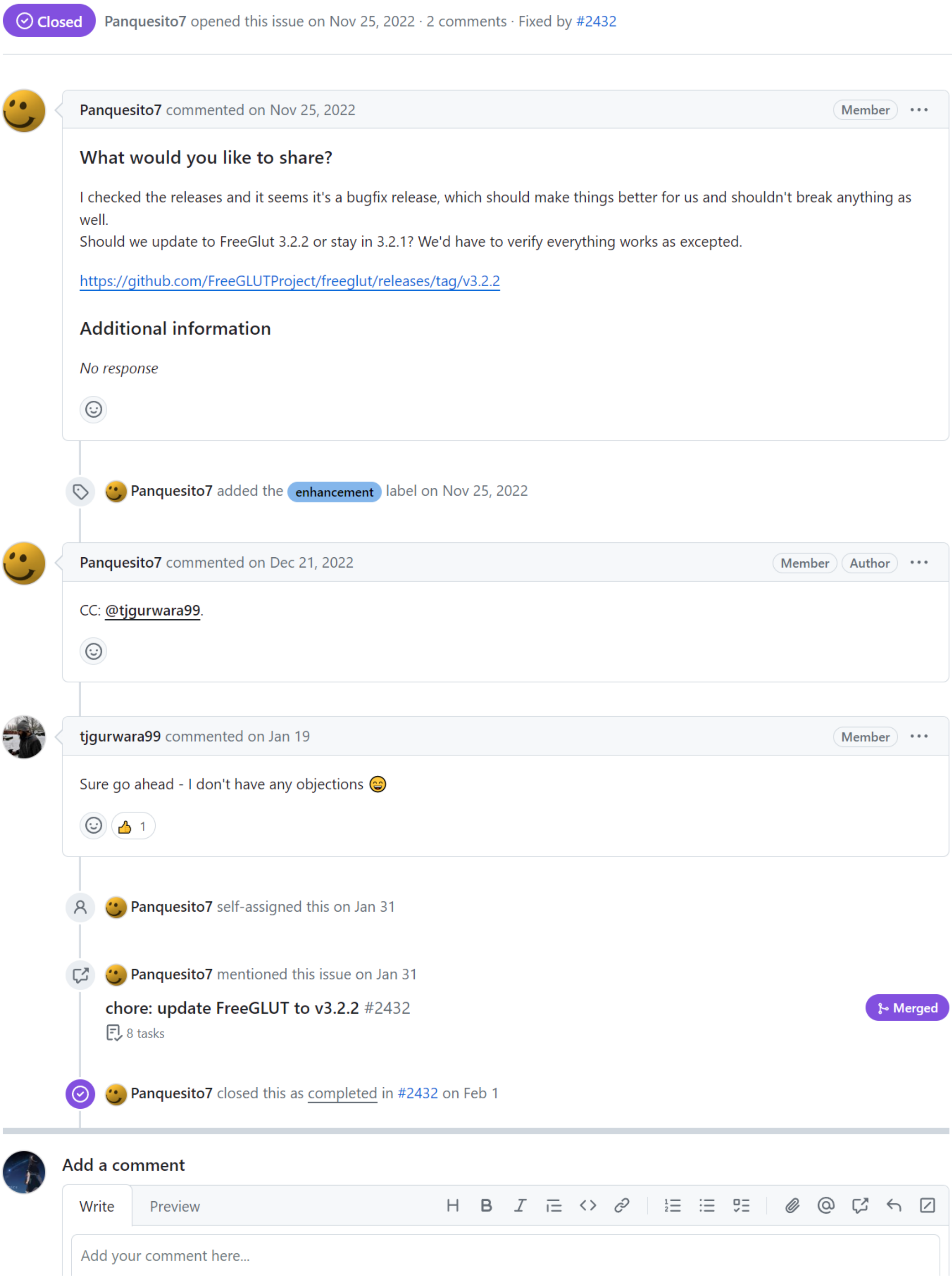
Panquesito7 force-pushed the `update_freeglut` branch from **bf49f92** to **4962d44** last year　Compare

tjgurwara99 approved these changes on Jan 31　View reviewed changes

github-actions bot added the approved label on Jan 31

Panquesito7 merged commit **b89d1da** into `master` on Feb 1　View details　Revert
12 checks passed

Panquesito7 deleted the `update_freeglut` branch last year



Closed Panquesito7 opened this issue on Nov 25, 2022 · 2 comments · Fixed by #2432

Panquesito7 commented on Nov 25, 2022 Member

What would you like to share?

I checked the releases and it seems it's a bugfix release, which should make things better for us and shouldn't break anything as well.
Should we update to FreeGlut 3.2.2 or stay in 3.2.1? We'd have to verify everything works as excepted.

https://github.com/FreeGLUTProject/freeglut/releases/tag/v3.2.2

Additional information

No response

Panquesito7 added the enhancement label on Nov 25, 2022

Panquesito7 commented on Dec 21, 2022 Member Author

CC: @tjgurwara99.

tjgurwara99 commented on Jan 19 Member

Sure go ahead - I don't have any objections 😄

👍 1

Panquesito7 self-assigned this on Jan 31

Panquesito7 mentioned this issue on Jan 31

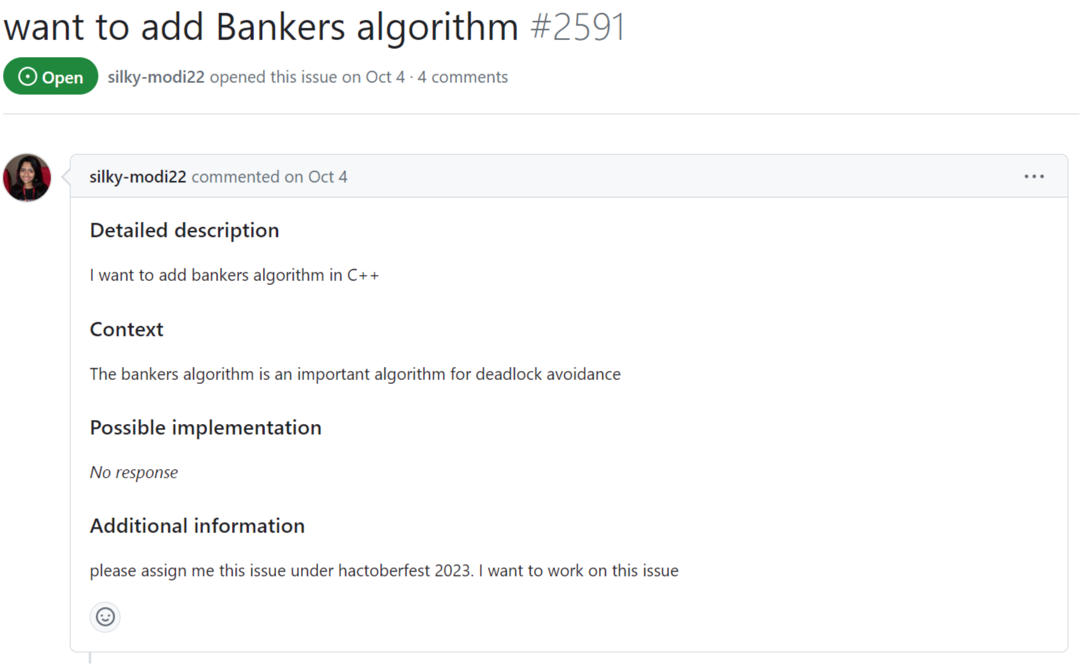
chore: update FreeGLUT to v3.2.2 #2432 Merged
8 tasks

Panquesito7 closed this as completed in #2432 on Feb 1

Add a comment

3.3 贡献代码

选择的 issue：https://github.com/TheAlgorithms/C-Plus-Plus/issues/2591



want to add Bankers algorithm #2591

Open silky-modi22 opened this issue on Oct 4 · 4 comments

silky-modi22 commented on Oct 4

Detailed description

I want to add bankers algorithm in C++

Context

The bankers algorithm is an important algorithm for deadlock avoidance

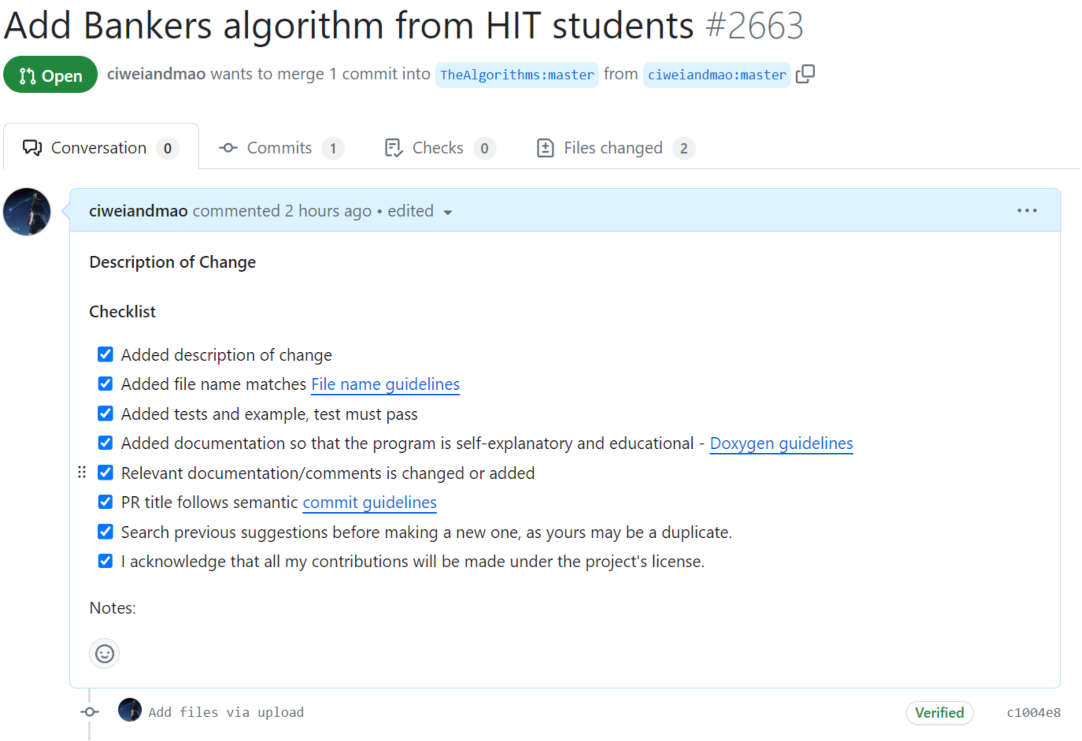
Possible implementation

No response

Additional information

please assign me this issue under hactoberfest 2023. I want to work on this issue

提交 PR 的地址：https://github.com/TheAlgorithms/C-Plus-Plus/pull/2663



Add Bankers algorithm from HIT students #2663

Open ciweiandmao wants to merge 1 commit into `TheAlgorithms:master` from `ciweiandmao:master`

ciweiandmao commented 2 hours ago • edited

Description of Change

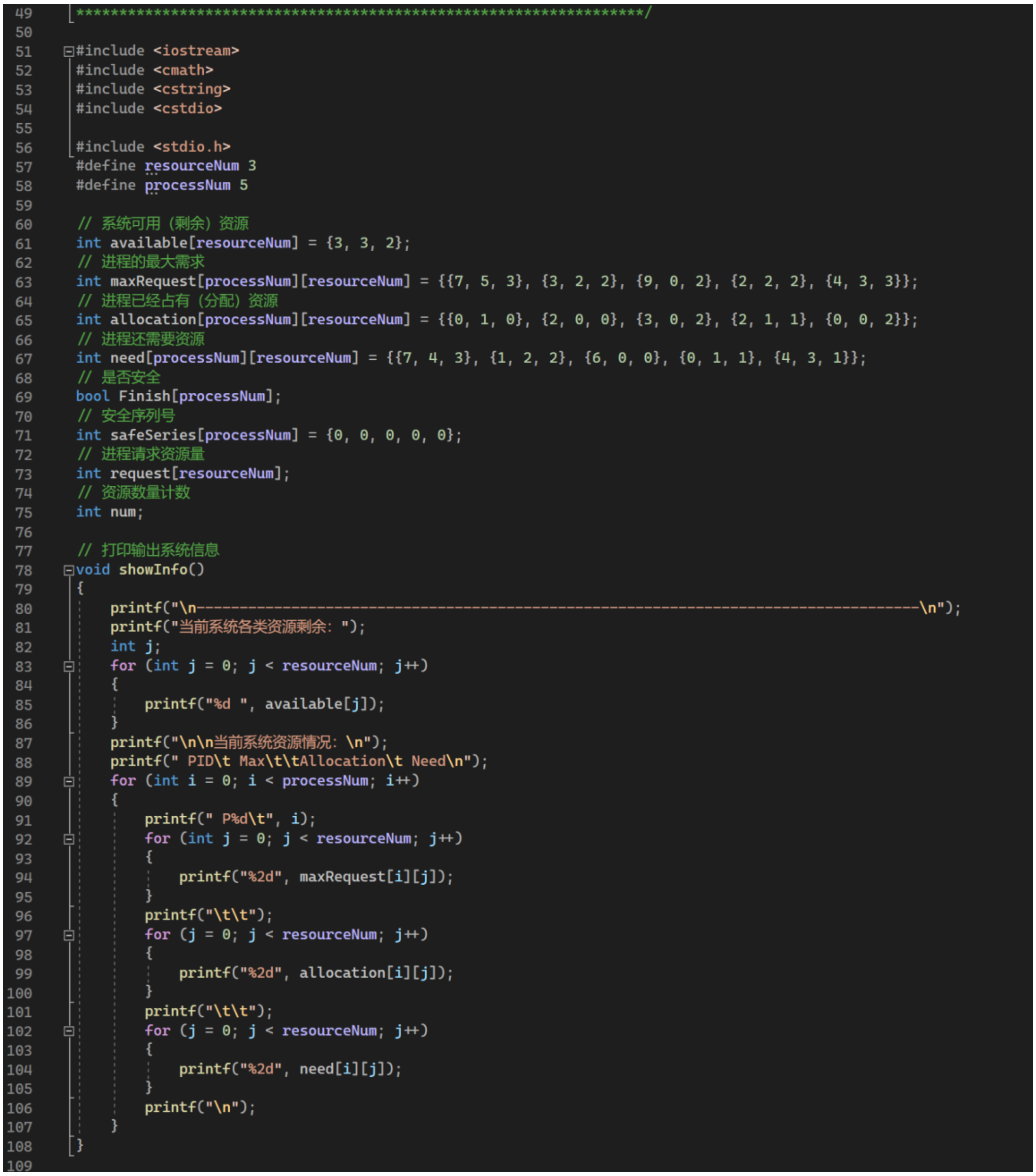
Checklist

- ☑ Added description of change
- ☑ Added file name matches File name guidelines
- ☑ Added tests and example, test must pass
- ☑ Added documentation so that the program is self-explanatory and educational - Doxygen guidelines
- ☑ Relevant documentation/comments is changed or added
- ☑ PR title follows semantic commit guidelines
- ☑ Search previous suggestions before making a new one, as yours may be a duplicate.
- ☑ I acknowledge that all my contributions will be made under the project's license.

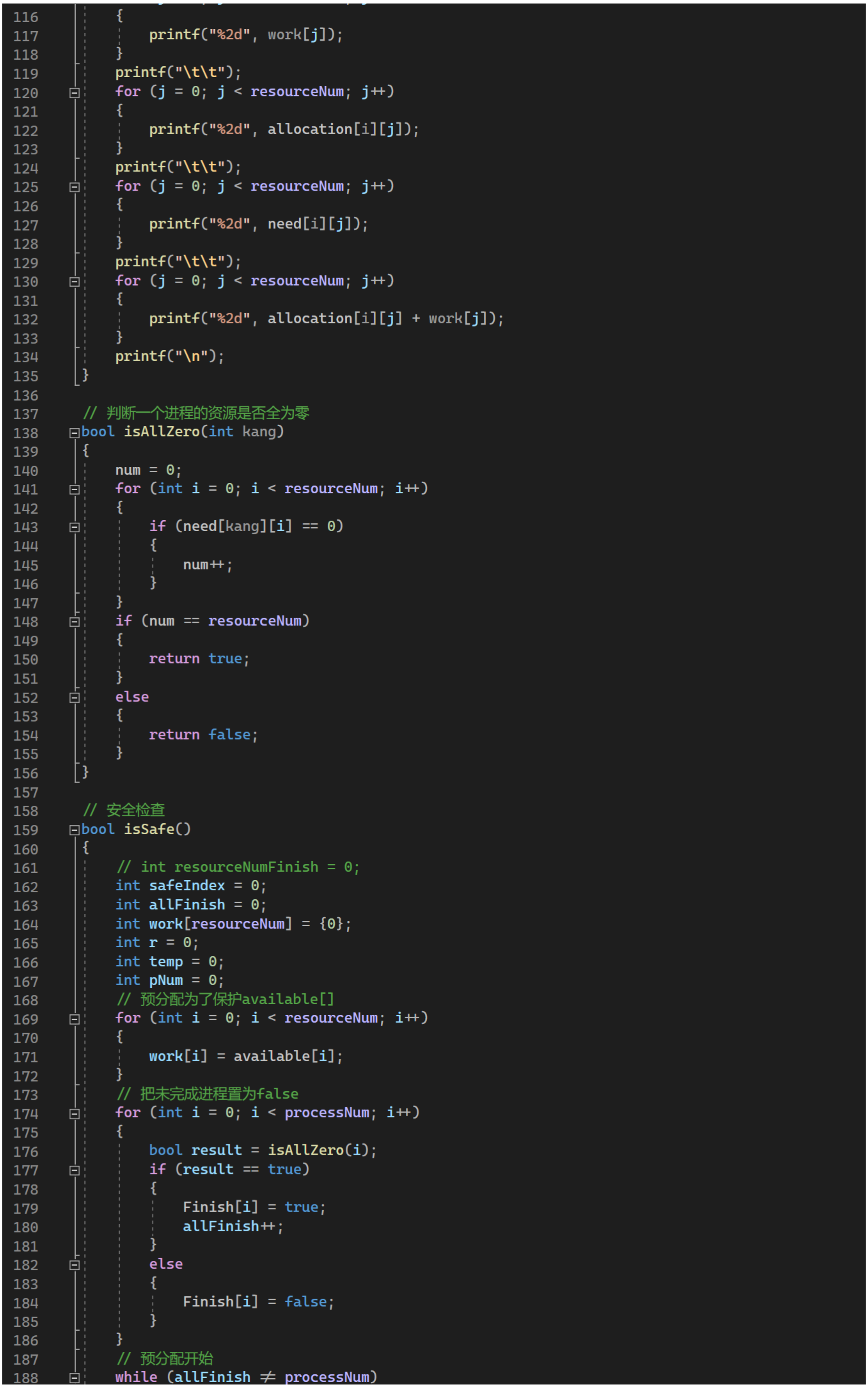
Notes:

`Add files via upload` Verified c1004e8

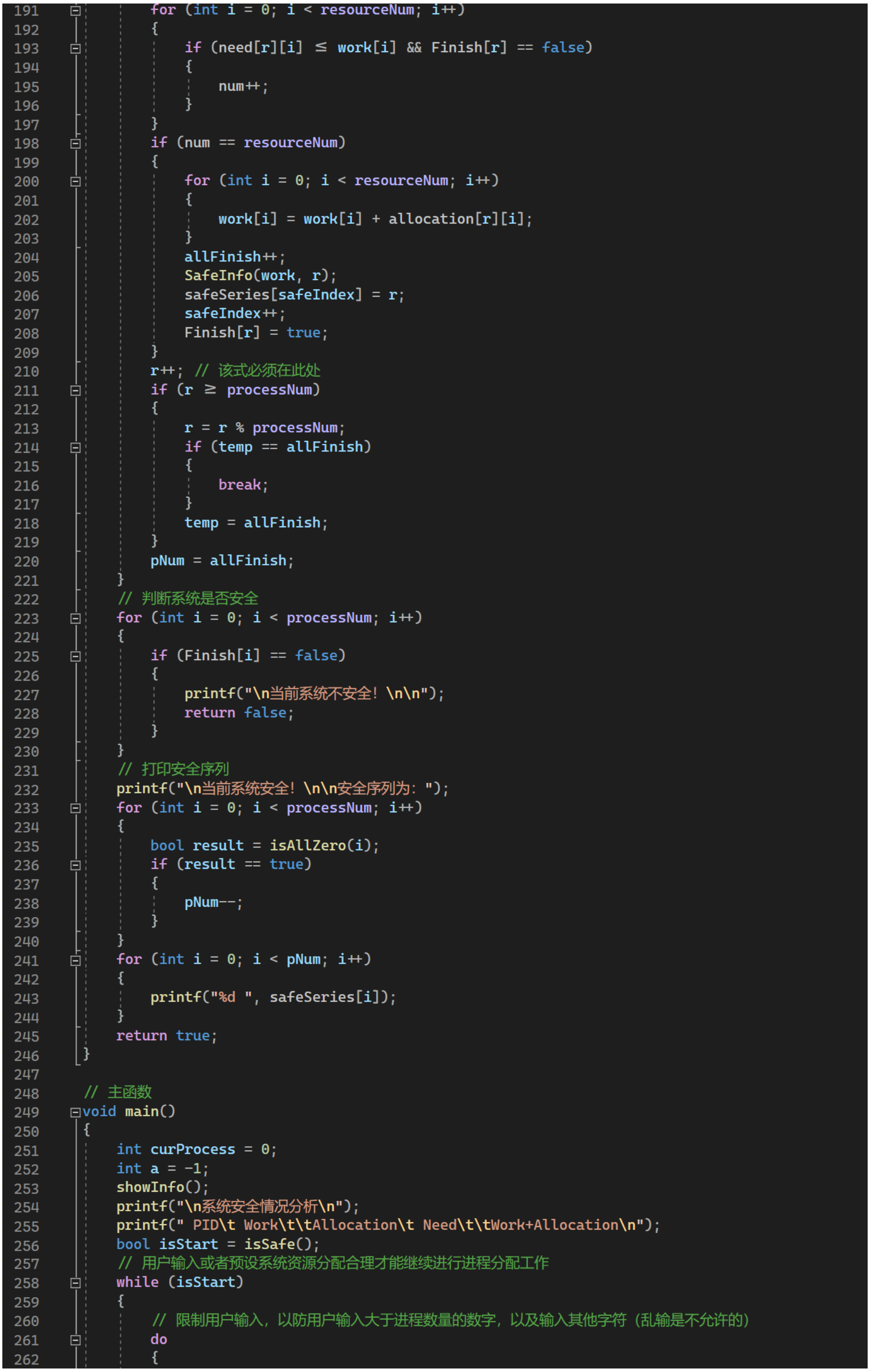
解决该 issue 需要写出一个银行家算法，银行家算法是用于避免死锁的一种资源分配和管理算法，它确保系统在分配资源时不会陷入无限等待的状态。这个算法主要应用于操作系统和并发编程中。它的核心思想是通过判断系统在分配资源后是否处于安全状态来进行资源分配。系统在运行之初，会获得所有可用资源的数量，以及每个进程所需的最大资源数和已分配资源数。这些信息构成了系统的状态。当一个进程请求分配资源时，系统会先检查分配资源后系统是否会陷入不安全状态。判断方法是尝试将资源分配给进程，然后检查系统是否能够找到一种安全序列来完成所有进程。安全序列是指系统能够按照某种顺序满足所有进程对资源的需求，不会发生死锁。如果能找到这样的序列，系统就可以安全地分配资源给请求的进程；否则，就需要等待。当一个进程请求资源时，系统会模拟分配资源并检查是否存在安全序列。如果存在安全序列，系统就会实际分配资源给进程；否则，进程会被阻塞，直到资源变得可用或者系统进入安全状态。当一个进程完成任务后，它会释放已经占用的资源。这些资源可以被其他进程请求和使用。核心代码如下



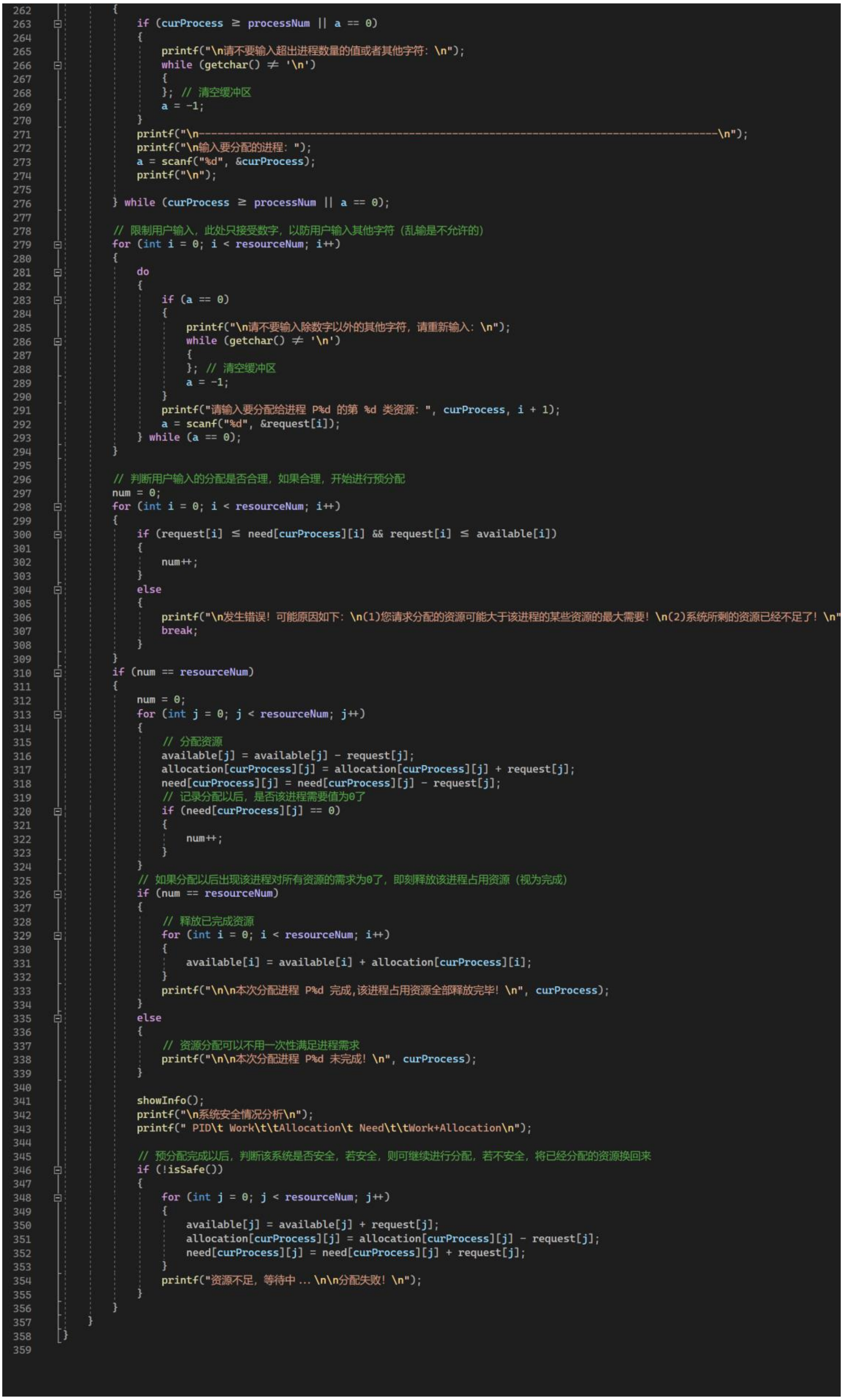
```
*******************************************************************/

#include <iostream>
#include <cmath>
#include <cstring>
#include <cstdio>

#include <stdio.h>
#define resourceNum 3
#define processNum 5

// 系统可用（剩余）资源
int available[resourceNum] = {3, 3, 2};
// 进程的最大需求
int maxRequest[processNum][resourceNum] = {{7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}};
// 进程已经占有（分配）资源
int allocation[processNum][resourceNum] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};
// 进程还需要资源
int need[processNum][resourceNum] = {{7, 4, 3}, {1, 2, 2}, {6, 0, 0}, {0, 1, 1}, {4, 3, 1}};
// 是否安全
bool Finish[processNum];
// 安全序列号
int safeSeries[processNum] = {0, 0, 0, 0, 0};
// 进程请求资源量
int request[resourceNum];
// 资源数量计数
int num;

// 打印输出系统信息
void showInfo()
{
    printf("\n-------------------------------------------------------------------------\n");
    printf("当前系统各类资源剩余: ");
    int j;
    for (int j = 0; j < resourceNum; j++)
    {
        printf("%d ", available[j]);
    }
    printf("\n\n当前系统资源情况: \n");
    printf(" PID\t Max\t\tAllocation\t Need\n");
    for (int i = 0; i < processNum; i++)
    {
        printf(" P%d\t", i);
        for (int j = 0; j < resourceNum; j++)
        {
            printf("%2d", maxRequest[i][j]);
        }
        printf("\t\t");
        for (j = 0; j < resourceNum; j++)
        {
            printf("%2d", allocation[i][j]);
        }
        printf("\t\t");
        for (j = 0; j < resourceNum; j++)
        {
            printf("%2d", need[i][j]);
        }
        printf("\n");
    }
}
```



```
    {
        printf("%2d", work[j]);
    }
    printf("\t\t");
    for (j = 0; j < resourceNum; j++)
    {
        printf("%2d", allocation[i][j]);
    }
    printf("\t\t");
    for (j = 0; j < resourceNum; j++)
    {
        printf("%2d", need[i][j]);
    }
    printf("\t\t");
    for (j = 0; j < resourceNum; j++)
    {
        printf("%2d", allocation[i][j] + work[j]);
    }
    printf("\n");
}

// 判断一个进程的资源是否全为零
bool isAllZero(int kang)
{
    num = 0;
    for (int i = 0; i < resourceNum; i++)
    {
        if (need[kang][i] == 0)
        {
            num++;
        }
    }
    if (num == resourceNum)
    {
        return true;
    }
    else
    {
        return false;
    }
}

// 安全检查
bool isSafe()
{
    // int resourceNumFinish = 0;
    int safeIndex = 0;
    int allFinish = 0;
    int work[resourceNum] = {0};
    int r = 0;
    int temp = 0;
    int pNum = 0;
    // 预分配为了保护available[]
    for (int i = 0; i < resourceNum; i++)
    {
        work[i] = available[i];
    }
    // 把未完成进程置为false
    for (int i = 0; i < processNum; i++)
    {
        bool result = isAllZero(i);
        if (result == true)
        {
            Finish[i] = true;
            allFinish++;
        }
        else
        {
            Finish[i] = false;
        }
    }
    // 预分配开始
    while (allFinish != processNum)
```



```
        for (int i = 0; i < resourceNum; i++)
        {
            if (need[r][i] <= work[i] && Finish[r] == false)
            {
                num++;
            }
        }
        if (num == resourceNum)
        {
            for (int i = 0; i < resourceNum; i++)
            {
                work[i] = work[i] + allocation[r][i];
            }
            allFinish++;
            SafeInfo(work, r);
            safeSeries[safeIndex] = r;
            safeIndex++;
            Finish[r] = true;
        }
        r++; // 该式必须在此处
        if (r >= processNum)
        {
            r = r % processNum;
            if (temp == allFinish)
            {
                break;
            }
            temp = allFinish;
        }
        pNum = allFinish;
    }
    // 判断系统是否安全
    for (int i = 0; i < processNum; i++)
    {
        if (Finish[i] == false)
        {
            printf("\n当前系统不安全！\n\n");
            return false;
        }
    }
    // 打印安全序列
    printf("\n当前系统安全！\n\n安全序列为：");
    for (int i = 0; i < processNum; i++)
    {
        bool result = isAllZero(i);
        if (result == true)
        {
            pNum--;
        }
    }
    for (int i = 0; i < pNum; i++)
    {
        printf("%d ", safeSeries[i]);
    }
    return true;
}

// 主函数
void main()
{
    int curProcess = 0;
    int a = -1;
    showInfo();
    printf("\n系统安全情况分析\n");
    printf(" PID\t Work\t\tAllocation\t Need\t\tWork+Allocation\n");
    bool isStart = isSafe();
    // 用户输入或者预设系统资源分配合理才能继续进行进程分配工作
    while (isStart)
    {
        // 限制用户输入，以防用户输入大于进程数量的数字，以及输入其他字符（乱输是不允许的）
        do
        {
```



```
        {
            if (curProcess ≥ processNum || a == 0)
            {
                printf("\n请不要输入超出进程数量的值或者其他字符: \n");
                while (getchar() ≠ '\n')
                {
                }; // 清空缓冲区
                a = -1;
            }
            printf("\n-----------------------------------------------------------------------------------------\n");
            printf("\n输入要分配的进程: ");
            a = scanf("%d", &curProcess);
            printf("\n");

        } while (curProcess ≥ processNum || a == 0);

        // 限制用户输入，此处只接受数字，以防用户输入其他字符（乱输是不允许的）
        for (int i = 0; i < resourceNum; i++)
        {
            do
            {
                if (a == 0)
                {
                    printf("\n请不要输入除数字以外的其他字符，请重新输入: \n");
                    while (getchar() ≠ '\n')
                    {
                    }; // 清空缓冲区
                    a = -1;
                }
                printf("请输入要分配给进程 P%d 的第 %d 类资源: ", curProcess, i + 1);
                a = scanf("%d", &request[i]);
            } while (a == 0);
        }

        // 判断用户输入的分配是否合理，如果合理，开始进行预分配
        num = 0;
        for (int i = 0; i < resourceNum; i++)
        {
            if (request[i] ≤ need[curProcess][i] && request[i] ≤ available[i])
            {
                num++;
            }
            else
            {
                printf("\n发生错误！ 可能原因如下: \n(1)您请求分配的资源可能大于该进程的某些资源的最大需要！ \n(2)系统所剩的资源已经不足了！ \n"
                break;
            }
        }
        if (num == resourceNum)
        {
            num = 0;
            for (int j = 0; j < resourceNum; j++)
            {
                // 分配资源
                available[j] = available[j] - request[j];
                allocation[curProcess][j] = allocation[curProcess][j] + request[j];
                need[curProcess][j] = need[curProcess][j] - request[j];
                // 记录分配以后，是否该进程需要值为0了
                if (need[curProcess][j] == 0)
                {
                    num++;
                }
            }
            // 如果分配以后出现该进程对所有资源的需求为0了，即刻释放该进程占用资源（视为完成）
            if (num == resourceNum)
            {
                // 释放已完成资源
                for (int i = 0; i < resourceNum; i++)
                {
                    available[i] = available[i] + allocation[curProcess][i];
                }
                printf("\n\n本次分配进程 P%d 完成,该进程占用资源全部释放完毕！ \n", curProcess);
            }
            else
            {
                // 资源分配可以不用一次性满足进程需求
                printf("\n\n本次分配进程 P%d 未完成！ \n", curProcess);
            }

            showInfo();
            printf("\n系统安全情况分析\n");
            printf(" PID\t Work\t\tAllocation\t Need\t\tWork+Allocation\n");

            // 预分配完成以后，判断该系统是否安全，若安全，则可继续进行分配，若不安全，将已经分配的资源换回来
            if (!isSafe())
            {
                for (int j = 0; j < resourceNum; j++)
                {
                    available[j] = available[j] + request[j];
                    allocation[curProcess][j] = allocation[curProcess][j] - request[j];
                    need[curProcess][j] = need[curProcess][j] + request[j];
                }
                printf("资源不足，等待中 ... \n\n分配失败！ \n");
            }
        }
    }
}

```

4 小结

这次实验中我们学习了并掌握基本的 Markdown 语法，并且在实验中获得了能够根据开源项目提供的文档，对项目进行了解和分析的能力。同时我们在实际参与某开源项目，贡献代码，提升了团队协作能力和代码编写，需求分析等能力。这次实验让我们受益匪浅。