Project No.1: A Simple Line Editor

Name	Student No.	Email	Responsibility
付新宇	14346005	fuxy203@163.com	Delete, Print, etc; Report
陈嘉民	14346002	Chenjm69@mail2.sysu.edu.cn	Insert, Line, etc; Report

1. Introduction (1 point)

What are the problems? Briefly describe how you solve them. (Font: Calibri, 12pt)

(1) How does the program distinguish between commands and line texts?

We add a special prefix '$' to every commands. So we just need to let the program check the first element of the entered line to determine whether it's command or line text.

(2) How does the program check if the commands entered are correct?

Firstly, we use nested if – else statement in the command_check function to determine which command it really is. If there's no match, the program would return a string indicating an error has occurred.
Secondly, in each command function, the program further checks the validity of parameters, like the range and the type (float or integer).

(3) What member functions of list can we use to implement insert command and delete command?

To implement insert command, we can use "insert()" or "push_back()".
To delete command, we can use "erase()" or "remove()" or "pop_back()".

2. Analysis and Design (3 points)

Data structures you use, the decision behind selecting them;
Describe the most important algorithms, again with a justification of your decision;
Time complexity of your algorithms.

We use list to store the line text entered. Since we usually enter several text lines continuously, the data structure we use may not need to be accessed randomly. So we choose list (instead of vector), a simple data structure with high efficiency.
Most important algorithm: operation on list using iterator (as we choose list that can't access randomly)
Time complexity (of 3 main operation inner function):

Insert_command(): O(1)

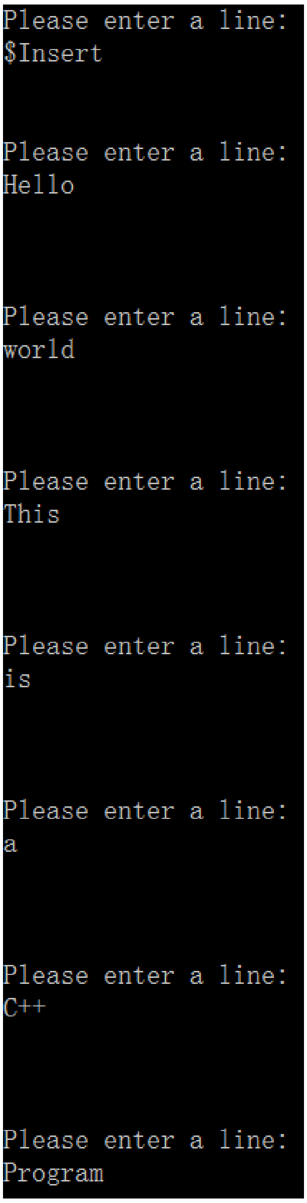
Delete_command(): O(n)

Line_command(): O(n)

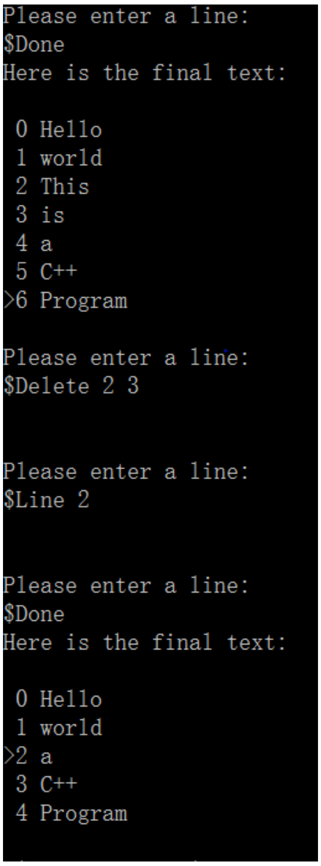
3. Test (1 points)

Describe the test plan -- how the program/system was verified. Show your test cases and results within screenshots.

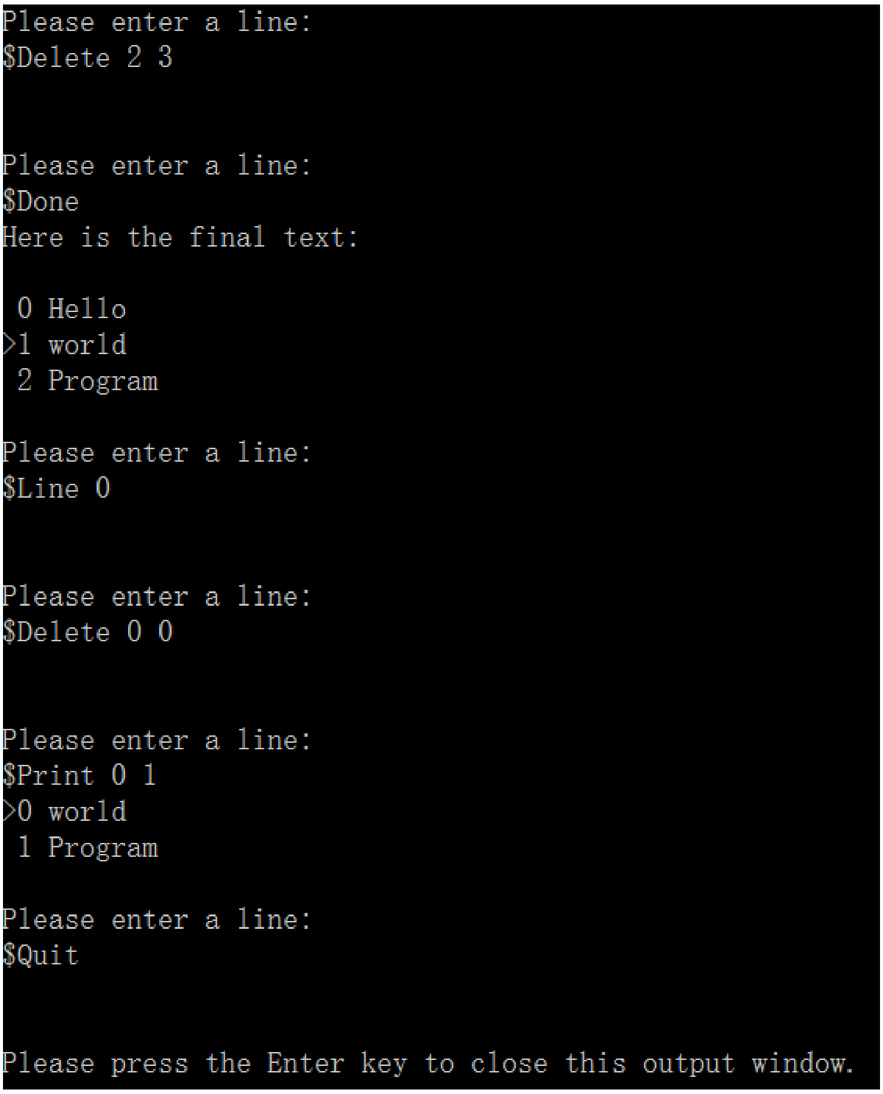
1. List all the normal situation to test different functions we want.



```
Please enter a line:
$Insert


Please enter a line:
Hello



Please enter a line:
world



Please enter a line:
This



Please enter a line:
is



Please enter a line:
a



Please enter a line:
C++



Please enter a line:
Program
```

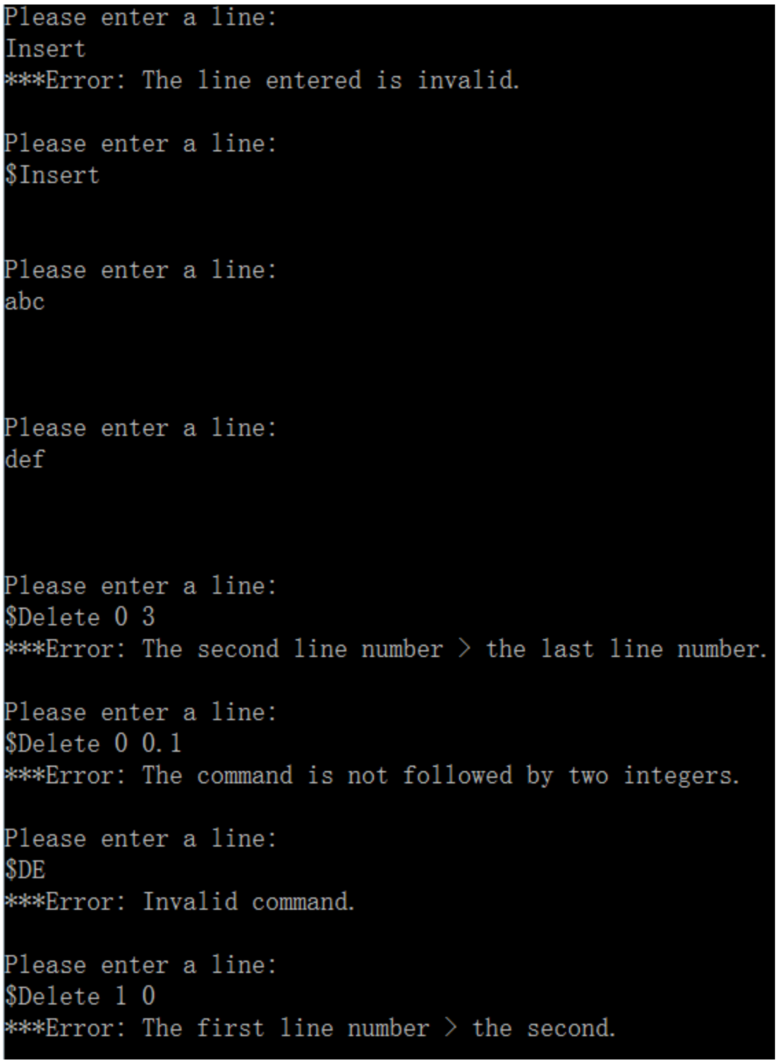


```
Please enter a line:
$Done
Here is the final text:

 0 Hello
 1 world
 2 This
 3 is
 4 a
 5 C++
>6 Program

Please enter a line:
$Delete 2 3


Please enter a line:
$Line 2


Please enter a line:
$Done
Here is the final text:

 0 Hello
 1 world
>2 a
 3 C++
 4 Program
```



```
Please enter a line:
$Delete 2 3


Please enter a line:
$Done
Here is the final text:

 0 Hello
>1 world
 2 Program

Please enter a line:
$Line 0


Please enter a line:
$Delete 0 0


Please enter a line:
$Print 0 1
>0 world
 1 Program

Please enter a line:
$Quit


Please press the Enter key to close this output window.
```

2. Try some illegal input and see how the error detector works.



```
Please enter a line:
Insert
***Error: The line entered is invalid.

Please enter a line:
$Insert


Please enter a line:
abc



Please enter a line:
def



Please enter a line:
$Delete 0 3
***Error: The second line number > the last line number.

Please enter a line:
$Delete 0 0.1
***Error: The command is not followed by two integers.

Please enter a line:
$DE
***Error: Invalid command.

Please enter a line:
$Delete 1 0
***Error: The first line number > the second.
```

4. Conclusion and Discussion (1 point)

What have you achieved and have not achieved?
Any other highlights about your design and implementation (you may get some bonus credits)

We have achieved the three basic functions of the Editor and yet not achieved the expanded functions.
Highlights:

1. We learn to make use of the return value of two member functions of list to solve the problem we met moving the iterator after inserting and erasing. [iterator insert(); iterator erase()]
2. We use the conception of package to separate the header file (Editor.h), the implement file (Editor.cpp) and the test file (源.cpp)
   .

5. Appendices (4 points)

The code listing (Your code should be well commented.)

Attachment:

Editor.h
Editor.cpp
source.cpp

6. References

Books and websites

[1] Y.Daniel Liang. Introduction to Programming with C++ (Third Edition) [M］. 机械工业出版社

[2] unknown. “Class Template <list> | std::list” [DB/OL] Available: http://www.cplusplus.com/reference/list/list/（ 2000-2016 ）

[2] unknown. “容器 list 使用之 erase”: http://blog.sina.com.cn/s/blog_66f74d9f0100om0f.html