Practical 5: Rolodex

Task

A Rolodex is a rotating file that is usually used to record business contact information. Cards are inserted in alphabetic order by name. You operate the Rolodex by rolling it forwards or backwards until you find the information you need (or find the place where a new card should be inserted).

Your task is to create a class that simulates a Rolodex and use it to implement a word-sorting program. The program starts with an empty Rolodex, then reads words from standard input and adds a card for each word into the appropriate position to maintain overall sorted order. To find the correct position, the Rolodex is rolled either backwards or forwards from the current card. When all input has been processed, the program moves the Rolodex to the beginning and then outputs each word in order, one per line.

SVN check out

Before you begin, you will need to check out the practical directory from the topic repository. This can be achieved by selecting Get from Version Control on the CLion welcome screen. In the window that appears, select Subversion from the dropdown list. If you have not previously connected CLion to the repository, click the add button (**+**) and paste the following URL into the text field (replacing **`FAN`** with your FAN):

```
https://topicsvn.flinders.edu.au/svn-repos/COMP2711/FAN
```

If you are prompted to login, use your University FAN and password. Expand the repository listing and select the **rolodex** directory. Click Check Out then select a location to store your project (preferably in a practicals directory) and click Open. From the destination list, choose the second option to create a project directory named **rolodex** and then click OK to complete the check out. A dialogue may appear confirming the subversion working format; if so, 1.8 is fine. Finally, when prompted if you would like to open the project, click Yes.

Automated marking

This practical is available for automatic marking via the quiz **Practical 5**, which is located in the Assessment module on FLO. To assess your solution, first commit your changes to the topic repository. This can be completed within CLion via the Commit option under the VCS menu (or alternatively with ⌘+K on macOS or Ctrl+K on Windows).

You should adhere to good development habits and enter a commit message that describes what you have changed since your last commit. When ready, click Commit to upload your changes and receive a revision number. Enter this revision number into the answer box for the relevant question (level).

There are no penalties for incorrect solutions, so if you do not pass all test cases, check the report output, modify your solution, commit, and try again. Remember to finish and submit the quiz when you are ready to hand in. You may complete the quiz as many times as you like—your final mark for the practical will be the highest quiz mark achieved. If you do start a new quiz attempt, ensure you reassess any levels you have previously completed.

Background

A basic version of the program that includes a skeleton **`Rolodex`** class is provided. The class has methods for rotating the Rolodex forwards and backwards, for returning the value of the card at the current position, and for inserting a new card either before or after the current card.

The initial version of the program implements the basic sorting algorithm, which is a modified version of insertion sort (it is consequently not very efficient). In addition, it supports a command line processing mechanism that allows you to run the program with options. The initial version recognises two options: **`-v`** (verbose) and **`-c`** (current). The *verbose* option causes the actions of the Rolodex to be reported as they occur, which might be useful when debugging, whilst the *current* option causes the program to print only the current card's value rather than all card values.

Since you do not know how many entries the Rolodex will hold, you will need to use dynamically allocated memory to store the information. An easy way to implement the Rolodex file is using a doubly linked list. Each item in the list will store the information for a single card and will need an instance variable to store the card value as well as pointers to the next and previous items. For example, you could represent the items like this:

```
struct RolodexItem {
    std::string value_;
    RolodexItem* next_;
    RolodexItem* prev_;
};
```

You will also need some way of knowing when you have reached the beginning or end of the Rolodex. One strategy is to use a circular list with a **sentinel** value that marks both the beginning and the end. This approach will considerably simplify the management of the links when items are inserted or removed.

Level 1: Basic operation

Complete the implementation of **`Rolodex`** so that all the methods work correctly. The insert methods should leave the Rolodex positioned so that the current item is the one just inserted.

Level 2: No duplicates

Modify the program so that the command line option **`-d`** (no duplicates) causes a new item to be inserted only if the item is *not* already present in the Rolodex. For example, if the input consisted of the text

```
far far far away
```

then the output would contain only two lines

```
away
far
```

Level 3: Deletion

Modify the program such that an input string that begins with a **`-`** causes that word (without the **`-`**) to be *deleted* from the Rolodex instead of inserted. For example, if the input is

```
the quick black fox jumps -black brown
over the lazy dog
```

then the output would contain the word **`brown`** but not **`black`**.

Deleting a word should be implemented by moving the Rolodex to the position where the word would normally be inserted, but then deleting the current item if it matches the word (if it does not match, no action is performed). You will need to add a method to the Rolodex class that deletes the current item and leaves the Rolodex positioned so that the current item is the one *following* the deleted item if there is a following item, or the item *preceding* the deleted item otherwise.

Hint: as part of your solution, consider the case where the input word is a **`-`** (i.e., a dash).

Level 4: Report

Modify the program so that if it is run with the command line option **`-r`** (report), it outputs a report of the Rolodex operations rather than the words themselves. The report should consist of the following integer values separated by spaces: the number of items inserted, the number of duplicate items omitted (expected to be 0 unless the **`-d`** option is in effect), the number of items deleted, the number of times the **`rotateForward`** operation was performed, and the number of times the **`rotateBackward`** operation was performed. For example, if the input consisted of the text

```
the quick brown fox jumps over the lazy dog
```

then the output should read

```
9 0 0 5 8
```

The report indicates that there were 9 words inserted, 0 duplicate words omitted, 0 words deleted, and that a total of 5 forward rotations and 8 backward rotations of the Rolodex were performed.

However, if the **`-d`** option is in effect for the same input, the output should instead read

```
8 1 0 5 8
```

In this case, the report indicates that there were 8 words inserted, 1 duplicate word omitted (the second occurrence of **`the`**), 0 words deleted, 5 forward rotations, and 8 backward rotations.