COMP 322
Winter Semester 2023

INSTRUCTOR: DR. CHAD ZAMMAR
chad.zammar@mcgill.ca

Assignment 2: Exploring Dynamic Memory Management

Due date: 12 March 2023, 11:59 PM.

Before you start:

- Research for similar problems is tolerated. However, your submission should reflect individual work and personal effort.
- Some of the topics may not be covered in class due to our limited time. You are encouraged to find answers online. You can also reach out to the TAs for guidance.
- Please submit your assignment before the due date to avoid penalties or worse risking your assignment being rejected.
- Submit one file called **assignment2.cpp** containing all the functions together with their implementations. It will also contain the main() function that runs everything.

Make sure your code is clear and readable. **Readability of your code as well as the quality of your comments will be graded**.

- No submission by email. Submit your work to mycourse.
- If your code does not compile it will not be graded.
- Be happy when working on your assignment, because a happy software developer has more inspiration than a sad one :).

We saw in assignment 1 some ways of creating a diff tool to check whether 2 files have the same content. In this assignment we will push the development a bit further in order to create a git-like system.

Git is a code versioning system that keeps track of all previous states of a file. When a file gets modified git will keep both versions of the file, the old one and the new modified one. Git will offer you some utilities to retrieve previous versions of the same file and to compare them with newer versions.

We will be using **dynamic memory management** in order to implement our Comp322 version of git. Let's call our program git322.

git322 will track one file and will keep all versions of this file in memory. The user decides when to take a snapshot of his/her modification. The user should be able to load a specific version to make it the current version or can also decide to delete any version of the file. Here are the allowed operations while running git322 program:

1. Adding a version of the file when the user finds suitable
2. Removing a version (any version that was being tracked)
3. Loading a specific version and making it the current active one
4. Printing to the screen the list of all versions of the file
5. Comparing any 2 versions of the file
6. Searching and displaying all the versions having a specific keyword
7. Exiting the program

git322 has to keep all versions of the file in memory in a linked list using dynamic memory allocation. Every element in the list holds information about one version of the file.

The following example shows the interaction between the user and git322 when you execute your program:

```
Welcome to the Comp322 file versioning system!

To add the content of your file to version control press 'a'
To remove a version press 'r'
```

```
To load a version press 'l'
To print to the screen the detailed list of all versions press 'p'
To compare any 2 versions press 'c'
To search versions for a keyword press 's'
To exit press 'e'
```

Create a file called file.txt and open it in your text editor and add the following content to it:

```
Dear Comp 322 students.
```

Save your file and go back to your command prompt window where you program is running and choose option 'a' to add the content of your file to the tracking system that you have obviously implemented by now.

By pressing 'a' the following output should be displayed to the screen:

```
Your content has been added successfully.
```

If you try to add the same content twice or if you don't make any modification to your file and you try to add the content after it was being added previously the following message should be displayed to the screen:

```
git322 did not detect any change to your file and will not create a new
version.
```

Which means that your code has to verify first whether the file has changed before attempting to add a new unnecessary version.

The logic of adding the content of the file to the tracking system should reside in a function called add and having the following signature:

void add(string content);

Now let's print all the available versions (we only have one so far) to the screen. This function is handy for debugging. From the menu press 'p'. This should print the following to the screen:

```
Number of versions: 1
Version number: 1
```

```
Hash value: 35345342435
Content: Dear Comp 322 students.
```

The logic of printing should reside in a function called print and having the following signature:

void print(void);

Now let's modify the file by adding a line to it, so go to your text editor and add the following to your file:

```
Dear Comp 322 students. C++ is a complicated language.
```

Save your file and go back to your command prompt window where you program is running and choose option 'a' to add the content of your file again.

By pressing 'a' the following output should be displayed to the screen:

```
Your content has been added successfully.
```

Now let's verify by printing. Choose 'p', the following should be printed out to the screen:

```
Number of versions: 2
Version number: 1
Hash value: 35345342435
Content: Dear Comp 322 students.

Version number: 2
Hash value: 98735230844
Content: Dear Comp 322 students. C++ is a complicated language.
```

If the user wants to work on the first version of the file, he/she has to load it to his/her editor first. This is what option 'l' is for. Go back to your command prompt where your program is still running and press 'l'. The following should be printed to the screen:

```
Which version would you like to load?
```

Now the user has to enter the number that corresponds to the version he/she wants. In our case it is version 1. Enter 1 as a response to the question. The following should be printed to the screen:

```
Version 1 loaded successfully. Please refresh your text editor to see
the changes.
```

If the user enters an invalid number (other than 1 or 2) an error message should be displayed:

```
Please enter a valid version number. If you are not sure please press
'p' to list all valid version numbers.
```

Verify that your text editor is indeed reflecting the newly loaded version. This is what you are supposed to have in your text editor now (you may need to refresh or to close and reopen the file):

```
Dear Comp 322 students.
```

The logic of loading the content of a revision into the file should reside in a function called load and having the following signature:

void load(int version);

Let's now add a new line to the file so that the text in the editor will look like this:

```
Dear Comp 322 students.
Coding in C++ is like going to the gym: No pain no gain!
```

Press 'a' in the command prompt of your program to add this 3rd version and then press 'p' to check the result. If all is good you must see the following as as result of 'p' command:

```
Number of versions: 3
Version number: 1
Hash value: 35345342435
Content: Dear Comp 322 students.

Version number: 2
Hash value: 98735230844
Content: Dear Comp 322 students. C++ is a complicated language.

Version number: 3
```

```
Hash value: 88345342411
Content: Dear Comp 322 students.
Coding in C++ is like going to the gym: No pain no gain!
```

Now let's reuse old wisdom from the first assignment. Press 'c' in the command prompt of your running program to compare 2 versions of your file. Let's assume that the user wants to compare the first version with the second one. By pressing 'c' you must see the following:

```
Please enter the number of the first version to compare:
```

The user inputs 1. The following will be printed to the screen:

```
Please enter the number of the second version to compare:
```

The user inputs 2.

The following result will be printed to the screen:

```
Line 1: Dear Comp 322 students. <<>> Dear Comp 322 students. C++ is a
complicated language.
```

Let's enter 'c' again to compare the first and the 3rd versions

```
Please enter the number of the first version to compare:
```

The user inputs 1. The following will be printed to the screen:

```
Please enter the number of the second version to compare:
```

The user inputs 3.

The following result will be printed to the screen:

```
Line 1: <Identical>
Line 2: <Empty line> <<>> Coding in C++ is like going to the gym: No
pain no gain!
```

The logic of comparing the content of 2 versions should reside in a function called compare and having the following signature:

void compare(int version1, int version2);

Now let's search for the version of the file containing the word "gym". Press 's' in the command prompt where your program is still hopefully running. The following should be printed to the screen:

```
Please enter the keyword that you are looking for:
```

The user types: gym

The program should output the following:

```
The keyword 'gym' has been found in the following versions:
Version number: 3
Hash value: 88345342411
Content: Dear Comp 322 students.
Coding in C++ is like going to the gym: No pain no gain!
```

If the keyword is found in multiple versions, all of them should be listed.

If the keyword is not found, display to the screen the following message:

```
Your keyword 'keyword' was not found in any version.
```

The logic of searching the tracking system should reside in a function called search and having the following signature:

void search(string keyword);

Finally we want now to be able to remove (delete) a version from the tracking history. Input 'r' in the command prompt of your running program. The following should be displayed to the screen:

```
Enter the number of the version that you want to delete:
```

Assuming that the user inputs 2 to delete the second version.

The following should be displayed to the screen if the operation was successful:

```
Version 2 deleted successfully .
```

If the user inputs a number other than the existing valid version numbers. The following should be printed to the screen:

```
Please enter a valid version number.
```

The logic of removing a version from the tracking system should reside in a function called remove and having the following signature:

void remove(int version);

Press ‘p’ to verify that version 2 is no more available. The following should appear on the screen:

```
Number of versions: 2
Version number: 1
Hash value: 35345342435
Content: Dear Comp 322 students.

Version number: 3
Hash value: 88345342411
Content: Dear Comp 322 students.
Coding in C++ is like going to the gym: No pain no gain!
```

Your mission is complete. Press ‘e’ to exit the program.

Please provide the needed main() function. main() should only implement the active program loop asking the user to input options and keeping the session interactive. Everything else should be coded in its appropriate function.

You will need to declare a pointer to your linked list as a global variable to be able to access it from within your functions.

Don’t forget to delete any memory that you allocate dynamically.

Grading Scheme:

- Adding a version of the file when the user finds suitable **(10 pts)**
- Removing a version (any version that was being tracked) **(10 pts)**
- Loading a specific version and making it the current active one **(15 pts)**

- Printing to the screen the list of all versions of the file **(10 pts)**
- Comparing any 2 versions of the file **(20 pts)**
- Searching and displaying all the versions having a specific keyword **(20 pts)**
- Exiting the program **(5 pts)**
- Providing main function implementing the interactive loop **(10 pts)**