BIRZEIT UNIVERSITY

Department of Computer Science

COMP2421 - Data Structures and Algorithms
(Second Semester – Spring 2023/2024)

Project#2 Due Date: April 25th 2024 (by 11:59 PM)

In this project, you will implement an application of stack: **Text editor with differential Undo/Redo**.

Your application should be able to load a string from the text file to the computer memory, manipulate the loaded text by inserting or removing strings, and perform the undo/redo operations on the text. Then, the application writes the modified text to the new output file based on the user's request. When the program prints the content of the stack, it shall keep its content without any change. Clear the stack's content when writing the modified text for the output.

You shall implement two stacks (using linked lists) and one queue:

- **Undo Stack**: stores the differential changes (insert/remove) made to the text, including the word, operation type, and position (starting index).
- **Redo Stack:** stores changes undone by the user, allowing for redo operations (involves the text involved, the operation type, and the index.
- **Add Queue:** Stores the words for the added string, then these words are added from the queue to both text and the undo stack.

Undo/Redo: Undo pops the word from the undo stack, pushes it to the redo stack, and reflects these changes to the text). Redo pops the entries from the Redo stack, reapplying changes to the text. The Undo/Redo can be done while the stacks are not empty, while the stacks are cleared when the user saves the text to the output file. Insert/remove operation can be done anywhere in the text; the text is updated up to each insert/remove or Undo/Redo operation.

Your application should be able to show the following information through a proper menu of the application:

1. Load the input file which contains the initial text.
2. Print the loaded text.
3. Insert strings to the text.
4. Remove strings from the text.
5. Perform Undo operation
6. Perform Redo operation

7. Print the Undo Stack and the Redo stack
8. Save the updated text to the output file.
9. Exit

Explanation Examples: (The initial test is "Welcome to Data Structures and Algorithms.")

1. Insert: "**Course code is COMP 2421.**". First, we add it to the **queue,** then to the **Undo**.

rear front

~~front~~ **2421.**	**COMP**	**Is**	**code**	**Course** ~~rear~~

Undo Stack: (-1 means the addition happens at the end of the original text)

Token	Operation	index
2421.	Insert	-1
COMP	Insert	-1
Is	Insert	-1
Code	Insert	-1
Course	Insert	-1

The updated text:

Welcome to Data Structures and Algorithms. **Course code is COMP 2421.**

2. Insert: "**This is an easy course.**"

Queue:

~~front~~ **course**	**easy**	**an**	**is**	**This** ~~rear~~

rear front

Undo Stack:

Course	Insert	-1
Easy	Insert	-1
An	Insert	-1
Is	Insert	-1
This	Insert	-1
2421.	Insert	-1
COMP	Insert	-1
Is	Insert	-1
Code	Insert	-1
Course	Insert	-1

Original text becomes:

Welcome to Data Structures and Algorithms. Course code is COMP 2421**. This is an easy course**

3. **Remove:** delete "**Algorithms**" the program searches for the first occurrence that matches the string to be deleted, deletes it, and adds it with its starting index to the undo stack (mark its status as delete, and its location to the starting index of "algorithm")

Undo Stack: (-1 means the addition happens at the end of the original text)

Algorithms	remove	32
Course	insert	-1
Easy	Insert	-1
An	Insert	-1
Is	Insert	-1
This	Insert	-1
2421.	Insert	-1
COMP	Insert	-1
Is	Insert	-1
Code	Insert	-1
Course	insert	-1

Text becomes:

"Welcome to Data Structures and. Course code is COMP 2421. This is an easy course"

The word "algorithms" disappeared but was logged to the undo stack.

Now, after executing 3 undo operations, the stacks are as follows:

The Undo Stack		
An	Insert	-1
Is	Insert	-1
This	Insert	-1
2421.	Insert	-1
COMP	Insert	-1
Is	Insert	-1
Code	Insert	-1

The Redo Stack		
Easy	Insert	-1
Course	insert	-1
Algorithms	remove	32

Note: the stack nodes contain the word we insert or remove, the starting index of that word, and the operation type (insert/remove).

The deadline for this assignment will be April 25th, 2024 (by 11:59 PM). LATE SUBMISSIONS will not be accepted for any reason. Before the discussion, please ensure your application runs properly on your laptop. Project discussions will be decided later.

Grading policy:

1. Your application should have all functionalities working properly.
2. The following notes will be considered during the grading process:
   a. There has to be adequate documentation and comments in the code (i.e., functions, loops, etc.);
   b. Your code should follow the code convention (i.e., spaces, indentations, etc.); and
   c. Your application should contain a menu to allow the user to select which option (s) he would like to run.
3. Properly handling file I/O errors, invalid user inputs, and edge cases in text editing operations.

Notes and submission instructions:

1. **This is individual work**. It should represent your efforts. It is fine to discuss your work and ask your colleagues, but you are not allowed to copy/paste part of the work of others or give it to anyone else. You are not allowed to post/copy from other websites and/or social media, which will be considered cheating.
2. Any **plagiarized** code will not be marked, resulting in a **zero** grade.
3. You are responsible for the submitted code.
4. **Document format**. Please submit only the code file (**c** file) containing your project's code. Please rename it as follows: "**P1_YourStudentID_FirstNameLastName_SectionNo.c**".
5. **Input/output file names**. Ensure the input/output file names are the same as in the specifications.
6. Include your full name, student ID, and section number at the beginning of your file.
7. Please do not compress the file; only the C-file is needed.
8. Files not following the convention in point 2 will not be marked.
9. You must use C language only.

Good luck!