DEPARTMENT OF SOFTWARE ENGINEERING



**TOPIC :--TEXT EDITOR** SUBMITTED TO-RAHUL SIR SUBMITTED BY-PRINCE

KUMAR(2K20/SE/100)

#### DELHI TECHNOLOGICAL UNIVERSITY, DELHI

**DELHI-110042**

#### SOFTWARE ENGINEERING DEPARTMENT

**DELHI TECHNOLOGICAL UNIVERSITY (FORMERLY DELHI COLLEGE OF ENGINEERING)**

## CERTIFICATE

I,PRINCE KUMAR(2K20/SE/100)**,** hereby declare that the project Dissertation titled “**TEXT EDITOR**” which is submitted by us to the Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is original and not opened and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi PRINCE KUMAR(2K20/SE/100) Date: 12/11/2021

## Software Engg. Deptt.

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering) Shahbad Daulatpur, Bawana Road-Delhi-42

Certificate

I hereby certify that the Project Dissertation titled “***TEXT EDITOR***” which is submitted by Prince Kumar(2K20/SE/100) and ,Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology is a record of the project work carried out by the students under my supervision. To do the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi PRINCE KUMAR(2K20/SE/100) Date: 12/11/2021

# ACKNOWLEDGEMENT

### The success and final outcome of this project required a lot of guidance and assistance from many people and We are extremely privileged to have got this all along the completion of our project. All that We have done is only due to such supervision and assistance and We would not forget to thank them.

We respect and thank **RAHUL SIR,** for providing me an opportunity to do the project work and giving us all support and guidance, which made me complete the project duly. We are extremely thankful to her for providing us such a nice support and guidance.

INTRODUCTION

We all have used text editor from the beginning of our education. A text editor is simply a type of computer program that edits plain text. So, in this project we will be making the text editor which is a must need program in every device, say laptops, desktops, mobiles, PC’s.

The project implements a text editor using Linked List and Stack data structures.

Each line of text is saved into a linked list node, which consists of a data portion holding the text and a pointer to the next node of the linked list, which is the next line of the text. The program makes use of Stack data structure to implement the Undo function.

#### Features:

* Open an existing file and parse every line to a linked list.
* Save into a file
* Insert text into line n (user given line number and text)
* Delete line n (user given line number)
* Move line n to line m (interchanging two user given lines)
* Replace text in line n (user given line number and text)
* Next page
* Previous page
* Undo (Implemented using Stack data structure)

# Prerequisites:

**LINKED LIST**

A linked list is a common data structure made of a chain of nodes in which each node contains a value and a pointer to the next node in the chain.

The head pointer points to the first node, and the last element of the list points to null. When the list is empty, the head pointer points to null.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

* **Link** − Each link of a linked list can store a data called an

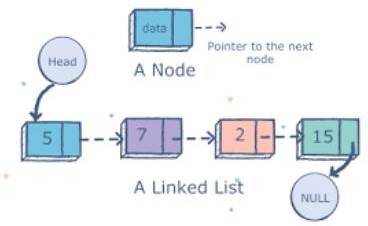
element.

* **Next** − Each link of a linked list contains a link to the next link

called Next.

* **LinkedList** − A Linked List contains the connection link to the

first link called First.



#### Basic Operations

Following are the basic operations supported by a list.

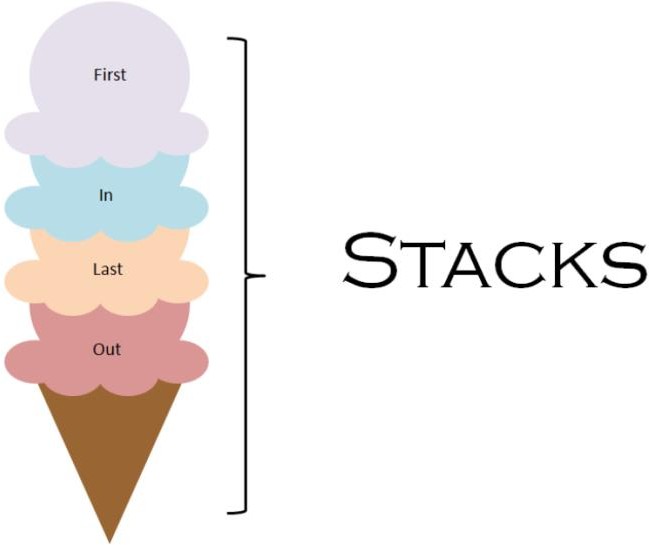
* **Insertion** − Adds an element at the beginning of the list.
* **Deletion** − Deletes an element at the beginning of the list.
* **Display** − Displays the complete list.
* **Search** − Searches an element using the given key.
* **Delete** − Deletes an element using the given key.

**STACK**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). It is an Abstract Data Type (ADT).

A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of the stack only. Likewise, Stack ADT allows all data operations at one end only.

A stack can be implemented by means of Array, Structure, Pointer, and Linked List. Stack can either be a fixed size one or it may have a sense of dynamic resizing.



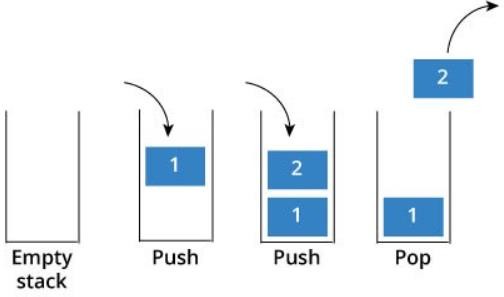
#### Basic Operations

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations −

* **push()** − Pushing (storing) an element on the stack.
* **pop()** − Removing (accessing) an element from the stack.
* **peek()** − get the top data element of the stack, without

removing it.

* **isFull()** − check if stack is full.
* **isEmpty()** − check if stack is empty.



IMPLEMENTATION

#### Insert text into line n (user given line number and text)

A doubly linked list is used to implement the insert text into line N feature of the text editor. In this feature, user is first asked the line number in which he wants to insert text and then the text he wants to be inserted.

If the line number entered already exists, the text is added into the previously saved text. Or else a new line is made and text is added. Special functions are created to implement all these options.

Created a counter to keep track of number of lines till now.

Pseudo code

if (lineGiven == 1) addToHead(dataGiven);

else if (lineGiven > numOfLines) insertFurtherAway(dataGiven,lineGiven);

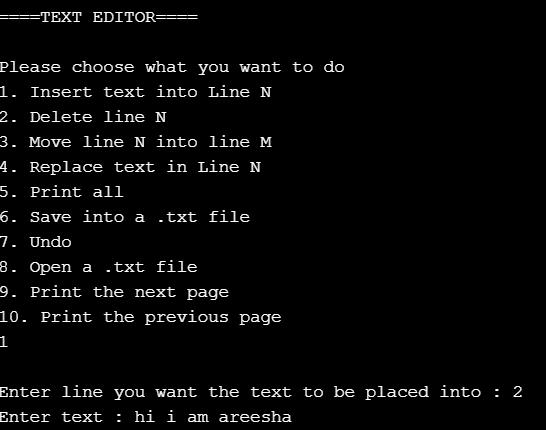
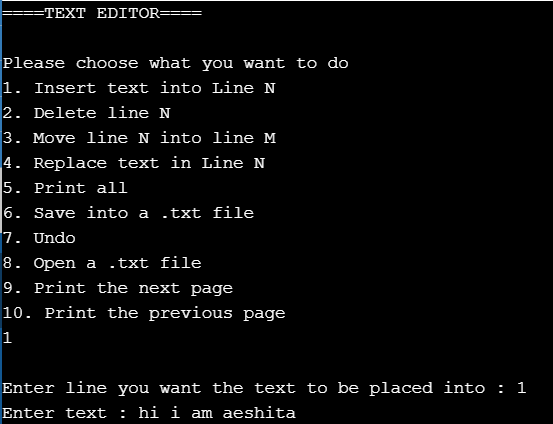
else if (lineGiven < numOfLines) insertTextInBetween(dataGiven,lineGiven);

else if (lineGiven == numOfLines) int selection;

cout<<"Enter 1 to replace the last line, enter 2 to insert a new line"; cin>>selection;

if (selection == 1) replaceTextInLine(dataGiven,lineGiven); else if (selection == 2) addToTail(dataGiven);

# Output



#### Delete line n (user given line number)

Next feature of our text editor is the deletion of line. User is asked the line number he wants to delete and according to that the line is deleted from the file.

To implement this 5 test cases are encountered

* When no line exists
* 1st line to be deleted
* When line given== numoflines(total lines)
* When line given is greater than total lines entered (does not exist)
* Line given in range Pseudo code

void deleteLine(int lineGiven) if (head == NULL)

cout<<"There is no line to be deleted/removed."<<endl; else if(head == tail)

node \*temp = head; delete(temp);

head = NULL; tail = NULL; numOfLines--;

else if(lineGiven == 1)

string backup = head->data; node \*temp = head;

node \*nextNode = head->next; head = nextNode; delete(temp);

numOfLines--;

undoCmd headRemoved; headRemoved.text = backup; headRemoved.lineNumber = 1;

headRemoved.commandNumber = 12; undoStack.push(headRemoved);

else if(lineGiven == numOfLines) node \*temp = head; undoCmd deletedLine;

deletedLine.commandNumber = 11;

while (temp->next != NULL && temp->next->next != NULL) temp = temp->next;

tail = temp;

string backup = temp->next->data; delete temp->next;

temp->next = NULL; numOfLines--; deletedLine.text = backup;

deletedLine.lineNumber = lineGiven; undoStack.push(deletedLine);

else if (lineGiven > numOfLines)

cout<<"Entered line is larger than existing lines..."<<endl;

else if (lineGiven < numOfLines) undoCmd deletedLine; deletedLine.commandNumber = 10; node \*prevNode = head;

node \*nextNode = head; node \*temp = head;

int iterator = 2; while(iterator < lineGiven)

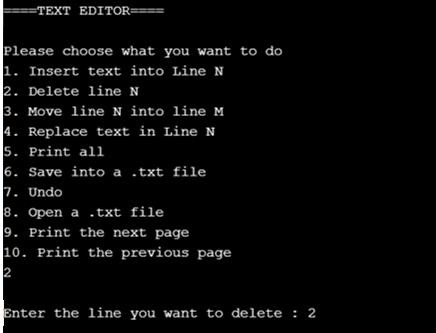
prevNode = prevNode->next;

nextNode = nextNode->next; iterator++;

nextNode = nextNode->next; temp = nextNode;

nextNode = nextNode->next; prevNode->next = nextNode;

Output



#### Move line n to line m (interchanging two user given lines)

This feature swaps two lines of the code. User is asked to enter line numbers for the lines swapping wants to be done.

Pseudo code

void moveNtoM(int nLineGiven, int mLineGiven) if (nLineGiven == 1)

string headText = head->data; deleteHead();

insertTextInBetween(headText,mLineGiven);

else

node \*temp = head; int iterator = 1;

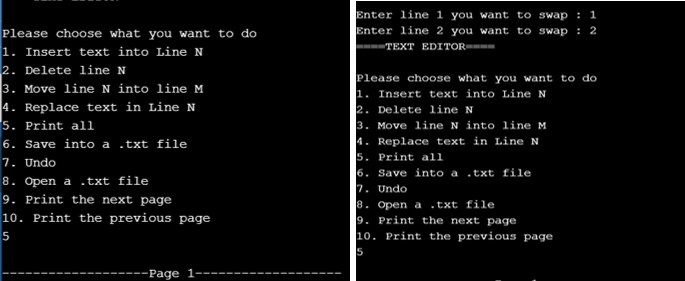
while(iterator < nLineGiven) temp = temp -> next; iterator++;

string dataSaved = temp->data; deleteLine(nLineGiven);

insertTextInBetween(dataSaved,mLineGiven);

undoCmd moveHeadToM; moveHeadToM.commandNumber = 2; moveHeadToM.nLine = nLineGiven; moveHeadToM.mLine= mLineGiven; undoStack.push(moveHeadToM);

**Output**



#### Replace text in line n (user given line number and text)

We make mistakes, in a text editor, replacing feature is a must. In here, user gives line number, and the already existing line context is deleted and the new text is added to it.

Pseudo code

void replaceTextInLine(string dataGiven,int lineGiven) undoCmd replacedLine;

if (numOfLines < lineGiven)

cout<<"The line you entered does not exists”;

else if (numOfLines >= lineGiven ) node \*temp = head;

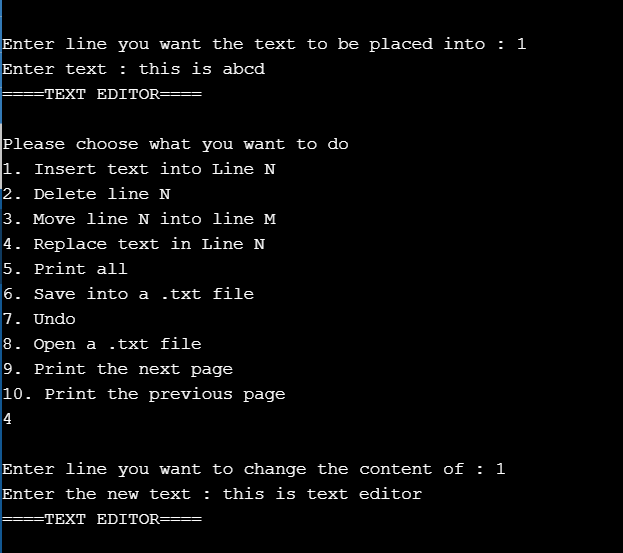
int goToLine = 1; while(goToLine < lineGiven)

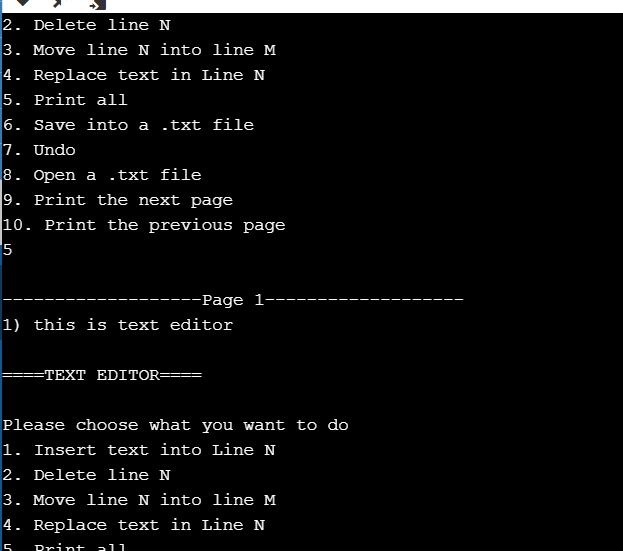
temp = temp->next; goToLine++;

string backup = temp->data; temp->data = dataGiven;

replacedLine.lineNumber = lineGiven; replacedLine.text = backup; replacedLine.commandNumber = 4; undoStack.push(replacedLine);

**Output**





**Undo**

Undo feature is to change the last move made by the user. It uses the stack data structure. Stack stores the choices and inputs done by the user and if undo choice is encountered, it pops the last element out.

