Roll No and Name - 20BCE526 Jainil Solanki

Project Definition - Text Editor using Circular Linked List

Project Explanation - This project is to implement a simple text editor using circular version of doubly linked list. We can Insert text in a particular line and delete as well. We can swap lines between each other replace text in a particular line. We can print all the lines and save all this data in a simple text file(.txt file). We can open a file created and edit it as well. We can print the data in the form of pages as well.

Data Structure - Circular Linked List

Code -

```
#include <iostream>
#include <string>
#include <fstream>
#include <stack>
#include <thread>
#include <chrono>
using namespace std;
using namespace std::this_thread;
using namespace std::chrono;

struct undoCmd{
        int lineNumber;
        string text;
        int commandNumber;
        int mLine;
        int nLine;
};

struct node{
        string data;
        struct node *next;
};

class linked_list
{
private:
        node *head;
        node *tail;
        int numOfLines = 0;
        int next = 1;
        stack<undoCmd> undoStack;
public:
	std::ofstream outfile;
	linked_list(){
		int choice = -1;
		head = NULL;
		tail = NULL;
		int prevPagePrinted = 1;
		while(choice != 0){
			cout<<"====TEXT EDITOR====\n"<<endl;
			cout<<"Please choose what you want to do\n1. Insert text into Line N"<<endl;
			cout<<"2. Delete line N\n3. Move line N into line M\n4. Replace text in Line
N"<<endl;
			cout<<"5. Print all\n6. Save into a .txt file\n7. Undo\n8. Open a .txt file\n9. Print the
next page\n10. Print the previous page"<<endl;
			cin>>choice;
			cout<<endl;
			if (choice == 1)			//Insertion of a line, any line, works fine
			{
				int lineGiven;
				string dataGiven;
				cout<<"Enter line you want the text to be placed into : ";
				cin >> lineGiven;
				cout<<"Enter text : ";
				cin.ignore(1);
				getline(cin,dataGiven);
				dataGiven+="\n";
				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);
                }
            }
        }
        else if (choice == 2)               //Deletion of a line, any line, works fine
        {
            int lineGiven;
            cout<<"Enter the line you want to delete : ";
            cin>>lineGiven;
            deleteLine(lineGiven);
        }
        else if (choice == 3)               //Interchanging two lines, any two line, works fine
        {
            int lineGiven1;
            int lineGiven2;
            cout<<"Enter line 1 you want to swap : ";
            cin>>lineGiven1;
            cout<<"Enter line 2 you want to swap : ";
            cin>>lineGiven2;
            moveNtoM(lineGiven1, lineGiven2);
        }
        else if (choice == 4)
        {
            int lineGiven;
            string dataGiven;
            cout<<"Enter line you want to change the content of : ";
            cin>>lineGiven;
            if (lineGiven > numOfLines)
            {
                cout<<"The line you entered exceeds the existing number of
lines..."<<endl;
            }
            else{
                cout<<"Enter the new text : ";
                cin>>dataGiven;
                dataGiven+="\n";
                replaceTextInLine(dataGiven, lineGiven);
            }
        }
        else if (choice == 5)               //Printing the whole list, works fine
        {
            printall();
            sleep_for(nanoseconds(1000));
            sleep_until(system_clock::now() + 1s);
```

```
}
else if (choice == 6)                   //Saving the list into a txt file, works fine
{
        saveAll();
}
else if (choice == 7)
{
        if (undoStack.empty())
        {
                cout<<"No command."<<endl;
                sleep_for(nanoseconds(1000));
                sleep_until(system_clock::now() + 1s);
        }
        else{
                //cout<<"under construction..."<<endl;
                undo();
                sleep_for(nanoseconds(1000));
                sleep_until(system_clock::now() + 1s);
        }
}
else if (choice == 8)
{
        node* current = head;
        node* next;
        while (current != NULL)
        {
           next = current->next;
                 free(current);
           current = next;
        }
        head = NULL;
        openFile();
}
else if (choice == 9)       //Printing the next page
{
        if (prevPagePrinted*10 > numOfLines)
        {
                // cout<<"No more page left to print."<<endl;
                printOnePage(prevPagePrinted);
                sleep_for(nanoseconds(1000));
                sleep_until(system_clock::now() + 1s);
        }
        else if (prevPagePrinted == 1)
        {
                printOnePage(1);
                prevPagePrinted++;
```

```
                                        sleep_for(nanoseconds(1000));
                                        sleep_until(system_clock::now() + 1s);
                                }
                                else{
                                        printOnePage(prevPagePrinted);
                                        prevPagePrinted++;
                                        sleep_for(nanoseconds(1000));
                                        sleep_until(system_clock::now() + 1s);
                                }
                        }
                        else if (choice == 10)                  //Printing the previous page
                        {
                                if (prevPagePrinted <= 0)
                                {
                                        prevPagePrinted = 1;
                                        printOnePage(1);
                                        sleep_for(nanoseconds(1000));
                                        sleep_until(system_clock::now() + 1s);
                                }
                                else if (prevPagePrinted == 1)
                                {
                                        prevPagePrinted--;
                                        printOnePage(1);
                                        sleep_for(nanoseconds(1000));
                                        sleep_until(system_clock::now() + 1s);
                                }
                                else{
                                        prevPagePrinted--;
                                        printOnePage(prevPagePrinted);
                                        sleep_for(nanoseconds(1000));
                                        sleep_until(system_clock::now() + 1s);
                                }
                        }
                }
        }
        void addToHead(string dataGiven){               //this function will add to Head
                if (head == NULL)                                       //no node, empty linked list
                {
                        node *temp;
                        temp = new node;
                        temp->data = dataGiven;
                        temp->next = NULL;
                        head = temp;
                        tail = head;
                        numOfLines++;
                }
                else                                                            //one or more than one
node
                {
                        node *temp;
                        temp = new node;
                        temp->data = dataGiven;
                        temp->next = NULL;
                        temp->next = head;
                        head = temp;
                        numOfLines++;
                }
                undoCmd adddedToHead;
                adddedToHead.lineNumber = 1;
                adddedToHead.commandNumber = 1;
                undoStack.push(adddedToHead);
        }

        void whateverAddToTail(string dataGiven){               //an extra function used to add to tail, had
to implement to make Undo function work, ignore this one please
                if (head == NULL)                                       //no node, empty linked list
                {
                        node *temp;
                        temp = new node;
                        temp->data = dataGiven;
                        temp->next = NULL;
                        head = temp;
                        tail = head;
                        numOfLines++;
                }
                else                                                            //one or more than one
node
                {
                        node *temp;
                        temp = new node;
                        temp->data = dataGiven;
                        temp->next = NULL;
                        tail->next = temp;
                        tail = temp;
                        numOfLines++;
                }
        }

        void whateverDeleteTail(){                                      //an extra function used to
delete from tail, had to implement to make Undo function work,ignore this one please
                node *temp = head;
                if (head == NULL)
```

```
		{
			cout<<"Nothing to be deleted."<<endl;
		}
		else if (head == tail)
		{
			temp = head;
			string backup = temp->data;
			delete(temp);
			head = NULL;
			tail = NULL;
			numOfLines--;
		}
		else
		{
			while (temp->next != NULL && temp->next->next != NULL)
			{
				temp = temp->next;
			}
			tail = temp;
			delete temp->next;
			temp->next = NULL;
			numOfLines--;
		}
	}

	void addToTail(string dataGiven){			//this function will add to Tail
		if (head == NULL)					//no node, empty linked list
		{
			node *temp;
			temp = new node;
			temp->data = dataGiven;
			temp->next = NULL;
			head = temp;
			tail = head;
			numOfLines++;
		}
		else							//one or more than one
node
		{
			node *temp;
			temp = new node;
			temp->data = dataGiven;
			temp->next = NULL;
			tail->next = temp;
			tail = temp;
			numOfLines++;
```

```
                }
                undoCmd addedToTail;
                addedToTail.lineNumber = 1;
                addedToTail.commandNumber = 8;
                undoStack.push(addedToTail);
        }

        void deleteHead(){                                              //function used to delete the very
first element, and update the head
                string backup = head->data;
                node *temp = head;
                node *nextNode = head->next;
                head = nextNode;
                delete(temp);
                numOfLines--;
                undoCmd deletedHead;
                deletedHead.text = backup;
                deletedHead.lineNumber = 1;
                deletedHead.commandNumber = 3;
                undoStack.push(deletedHead);
        }

        void deleteTail(){                                              //function used to delete the very
last element, and update the tail
                node *temp = head;
                if (head == NULL)
                {
                        cout<<"Nothing to be deleted."<<endl;
                }
                else if (head == tail)
                {
                        temp = head;
                        string backup = temp->data;
                        delete(temp);
                        head = NULL;
                        tail = NULL;
                        numOfLines--;
                        undoCmd deletedTail;
                        deletedTail.text = backup;
                        deletedTail.lineNumber = 1;
                        deletedTail.commandNumber = 7;
                        undoStack.push(deletedTail);
                }
                else
                {
                        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--;
				undoCmd deletedTail;
				deletedTail.text = backup;
				deletedTail.lineNumber = 1;
				deletedTail.commandNumber = 7;
				undoStack.push(deletedTail);
			}
		}

		void insertTextInBetween(string dataGiven, int lineGiven){		//this function will insert text in
the given line, and will push all the other lines
			if (lineGiven == 0)
			{
				cout<<"There's no line 0, did you mean 1 (cough...Google
suggestions...cough)"<<endl;
			}
			else if (lineGiven == 1)
			{
				if (head == NULL)						//no node, empty linked
list
				{
					node *temp;
					temp = new node;
					temp->data = dataGiven;
					temp->next = NULL;
					head = temp;
					tail = head;
					numOfLines++;
				}
				else								//one or more
than one node
				{
					node *temp;
					temp = new node;
					temp->data = dataGiven;
					temp->next = NULL;
					temp->next = head;
					head = temp;
					numOfLines++;
```

```
                }
                //May be unnecessary, dunno
                undoCmd insertedToHead;
                insertedToHead.lineNumber = 1;
                insertedToHead.commandNumber = 5;
                undoStack.push(insertedToHead);
                // addToHead(dataGiven);
                // numOfLines++;
            }
            else{
                node *prevNode = head;
                node *nextNode = head;
                node *temp = new node();
                temp->data = dataGiven;
                temp->next = NULL;
                int iterator = 2;
                while(iterator < lineGiven)
                {
                    prevNode = prevNode->next;
                    nextNode = nextNode->next;
                    iterator++;
                }
                nextNode = nextNode->next;
                prevNode->next = temp;
                temp->next = nextNode;
                numOfLines++;
                undoCmd insertedInBetween;
                insertedInBetween.lineNumber = lineGiven;
                insertedInBetween.commandNumber = 6;
                undoStack.push(insertedInBetween);
            }
        }

        void replaceTextInLine(string dataGiven,int lineGiven){          //this function will overwrite
anything written in the given line
            undoCmd replacedLine;
            if (numOfLines < lineGiven)
            {
                cout<<"The line you entered exceeds the existing number of lines..."<<endl;
            }
            else if (lineGiven == 0)
            {
                cout<<"There's no line 0, did you mean 1 (cough...Google
suggestions...cough)"<<endl;
            }
            else if (numOfLines >= lineGiven )
            {
                node *temp = head;
                int goToLine = 1;
                while(goToLine < lineGiven)
                {
                    temp = temp->next;
                    goToLine++;
                }
                string backup = temp->data;
                temp->data = dataGiven;        //change what is inside the node number that has
been given as line parameter
                replacedLine.lineNumber = lineGiven;
                replacedLine.text = backup;
                replacedLine.commandNumber = 4;
                undoStack.push(replacedLine);
            }
        }

        void deleteLine(int lineGiven){                                    //this    function
should delete anything in the given line, also decreases the numOfLines
            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 == 0){
                cout<<"There's   no   line   0,   did   you   mean   1   (cough...Google
suggestions...cough)"<<endl;
            }
            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;
                string backup = temp->data;
                delete(temp);
                numOfLines--;
                deletedLine.text = backup;
```

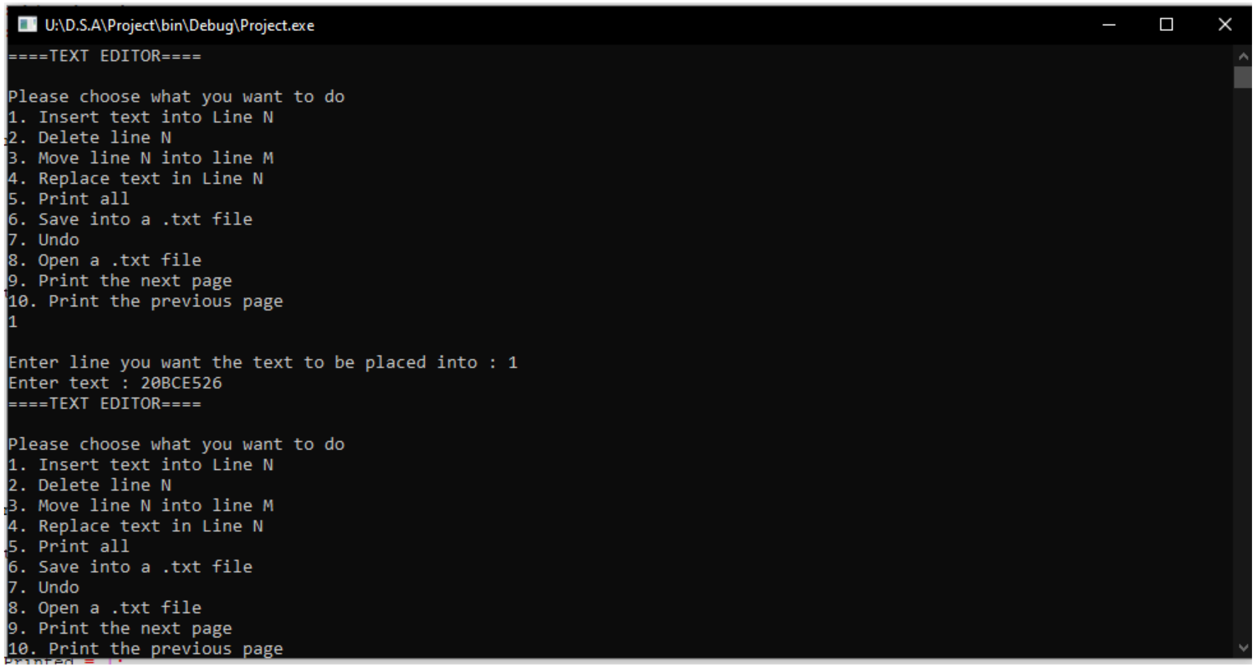
```
                        deletedLine.lineNumber = lineGiven;
                        undoStack.push(deletedLine);
                }
        }

        void insertFurtherAway(string dataGiven, int lineGiven){               //will print /n lines if given line is
larger than numOfLines
                undoCmd insertedFurtherAway;
                insertedFurtherAway.lineNumber = 0;
                insertedFurtherAway.commandNumber = 9;
                if (head == NULL)
                {
                        while(numOfLines < lineGiven-1)
                        {
                                whateverAddToTail("\n");
                                insertedFurtherAway.lineNumber++;
                        }
                        // insertedFurtherAway.lineNumber++;
                        whateverAddToTail(dataGiven);
                }
                else{
                        while(numOfLines < lineGiven-1)
                        {
                                whateverAddToTail("\n");
                                insertedFurtherAway.lineNumber++;
                        }
                        whateverAddToTail(dataGiven);
                }
                undoStack.push(insertedFurtherAway);
        }

        void moveNtoM(int nLineGiven, int mLineGiven){                         //function used to Move line N
into line M
                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);
    }

    void printOnePage(int pageGiven){                                   //function    used
to print only one page, only 10 or if there are less than 10 lines, it'll print only those lines
        node *temp = head;
        if (numOfLines < pageGiven*10)
        {
            int iterator = 1;
            while(iterator < (pageGiven*10)-9){
                temp = temp->next;
                iterator++;
            }
            for (int start = (pageGiven*10)-9 ; start <= numOfLines; start++)
            {
                cout<<start<<") "<<temp->data<<endl;
                temp = temp->next;
            }
            cout<<"-------------------Page "<<pageGiven<<"-------------------\n";
        }
        else if (numOfLines >= pageGiven * 10)
        {
            int iterator = 1;
            while(iterator < (pageGiven*10)-9){
                temp = temp->next;
                iterator++;
            }
            for (int start = (pageGiven*10)-9 ; start <= pageGiven*10; start++)
            {
                cout<<start<<") "<<temp->data<<endl;
                temp = temp->next;
            }
            cout<<"-------------------Page "<<pageGiven<<"-------------------\n";
        }
        else if (pageGiven * 10 > numOfLines)
        {
            cout<<"WHOOSH, you want to print an inexisting page, collect yourself!"<<endl;
        }
    }

    void openFile(){                                //function used to open a file from the same folder
this cpp file is in
        string fileName;
        cout<<"Enter the file name : ";
        cin>>fileName;
        fileName+=".txt";
        ifstream myfile;
        myfile.open(fileName);
        string s;
        while(getline(myfile,s))
        {
            addToTail(s);
        }
        myfile.close();
    }

    void undo(){                                        //function used to undo the last action taken
        undoCmd temp = undoStack.top();
        if (temp.commandNumber == 1)
        {
            cout<<"Added To head, removing from head..."<<endl;
            deleteHead();
            undoStack.pop();
        }
        else if (temp.commandNumber == 2)
        {
            cout<<"Moved M to N, moving N to M"<<endl;
            moveNtoM(temp.mLine, temp.nLine);
            undoStack.pop();
        }
        else if (temp.commandNumber == 3)
        {
            cout<<"Deleted head, replacing head..."<<endl;;
            addToHead(temp.text);
            undoStack.pop();
        }
        else if (temp.commandNumber == 4)
        {
            cout<<"Replaced line, replacing again..."<<endl;
            replaceTextInLine(temp.text,temp.lineNumber);
            undoStack.pop();
        }
```

```
else if (temp.commandNumber == 5)
{
        cout<<"Inserted to Head, removing from head..."<<endl;
        deleteHead();
        undoStack.pop();
}
else if (temp.commandNumber == 6)
{
        cout<<"Inserted in between, removing that line..."<<endl;
        deleteLine(temp.lineNumber);
        undoStack.pop();
}
else if (temp.commandNumber == 7)
{
        cout<<"Deleted Tail, inserting again..."<<endl;
        addToTail(temp.text);
        undoStack.pop();
}
else if (temp.commandNumber == 8)
{
        cout<<"Added to tail, removing from tail..."<<endl;
        deleteTail();
        undoStack.pop();
}
else if (temp.commandNumber == 9)
{
        int whatever = temp.lineNumber;
        while(whatever >= 0){
                whateverDeleteTail();
                whatever--;
        }
        undoStack.pop();
}
else if (temp.commandNumber == 10)
{
        cout<<"Line deleted, inserting again..."<<endl;
        insertTextInBetween(temp.text, temp.lineNumber);
        undoStack.pop();
}
else if (temp.commandNumber == 11)
{
        cout<<"Last line deleted, inserting again..."<<endl;
        addToTail(temp.text);
        undoStack.pop();
}
else if (temp.commandNumber == 12)
```

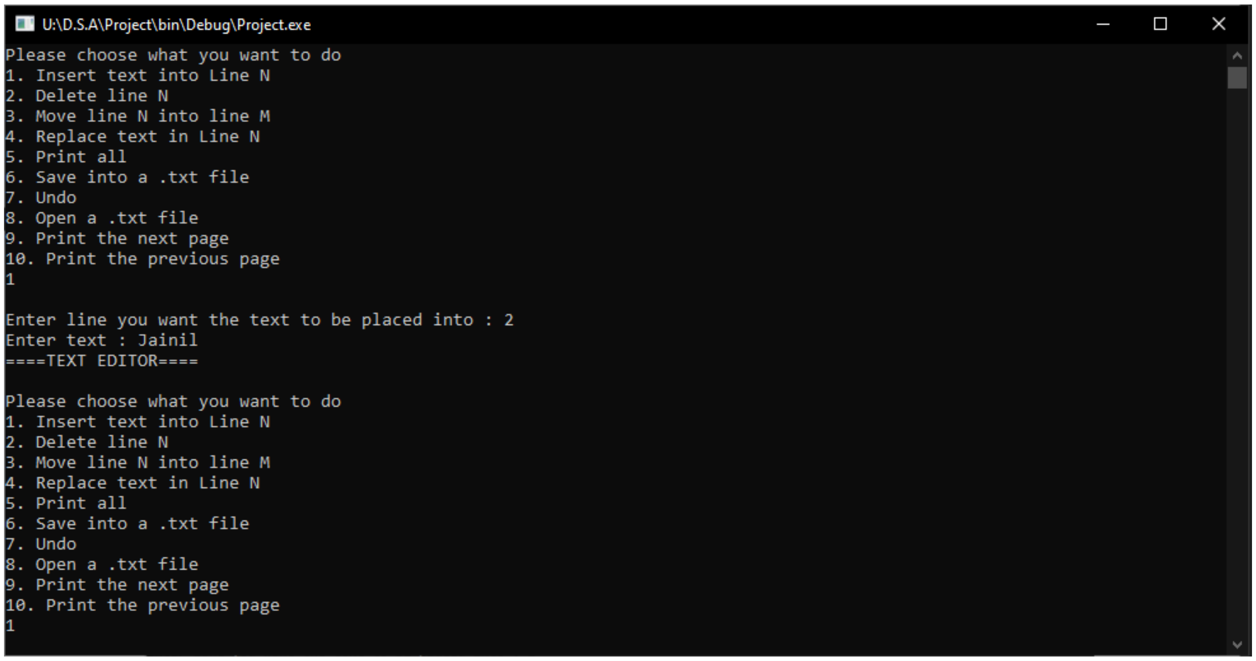
```
		{
			cout<<"First line deleted, inserting again..."<<endl;
			addToHead(temp.text);
			undoStack.pop();
		}
	}

	void printall(){						//function used to print the whole linked list
		node *temp = head;
		int linePrinted = 1;
		int pagePrinted = 2;
		int choice;
		if (head == NULL)
		{
			cout<<"no elements here, yay!"<<endl;
		}
		else{
			while(temp!=NULL)
			{
				if (linePrinted == 1)
				{
					cout<<"-------------------Page "<<"1"<<"-------------------\n";
				}
				else if ((linePrinted-1) % 10 == 0)
				{
					cout<<"-------------------Page "<<pagePrinted<<"-------------------\n";
					pagePrinted++;

				}
				cout<<linePrinted<<") "<<temp->data<<endl;
				temp = temp->next;
				linePrinted++;
			}
		}
	}

	void saveAll(){
		node *temp = head;
		int linePrinted = 1;
		int pagePrinted = 2;
		string fileName;
		cout<<"Enter the file name : ";
		cin>>fileName;
		fileName+=".txt";
		outfile.open(fileName, ios_base::app);
		while(temp!=NULL)
```

```
				{
					outfile<<temp->data;
					temp = temp->next;
					linePrinted++;
				}
				outfile.flush();
				outfile.close();
			}

			// void numOfLinesp(){				//Will print the numOfLines, used for debugging
			//		cout<<numOfLines<<endl;
			// }
};

int main(int argc, char const *argv[])
{
			linked_list ourList;
			return 0;
}
```

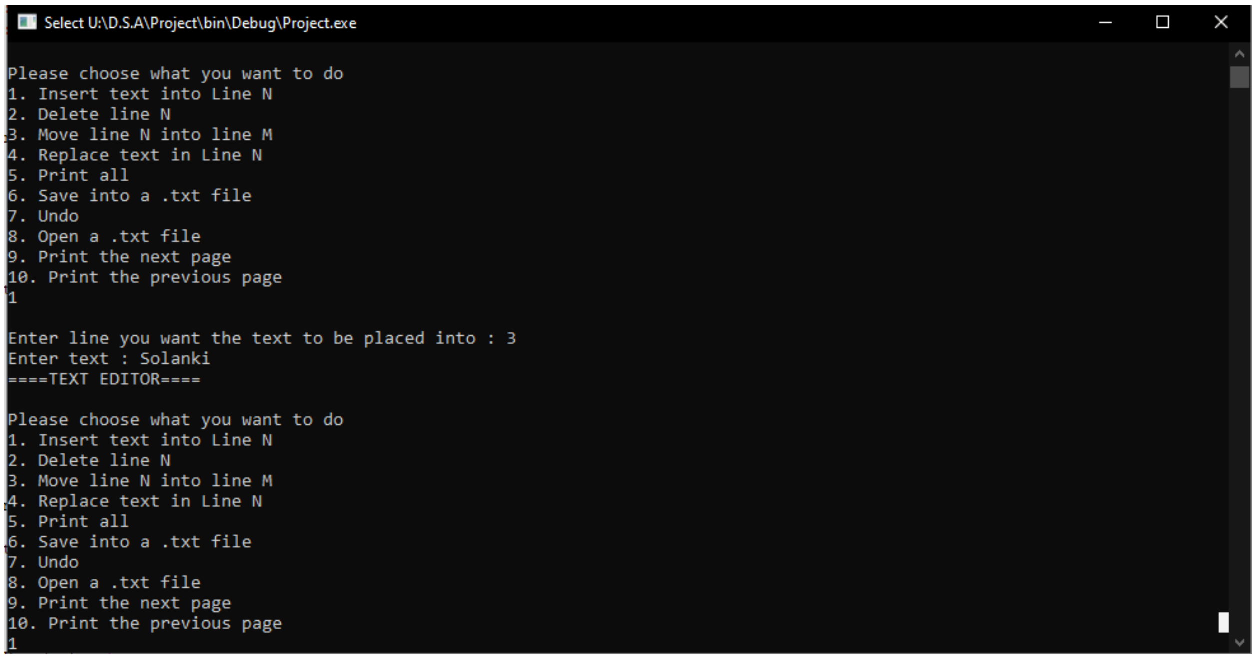
Input:



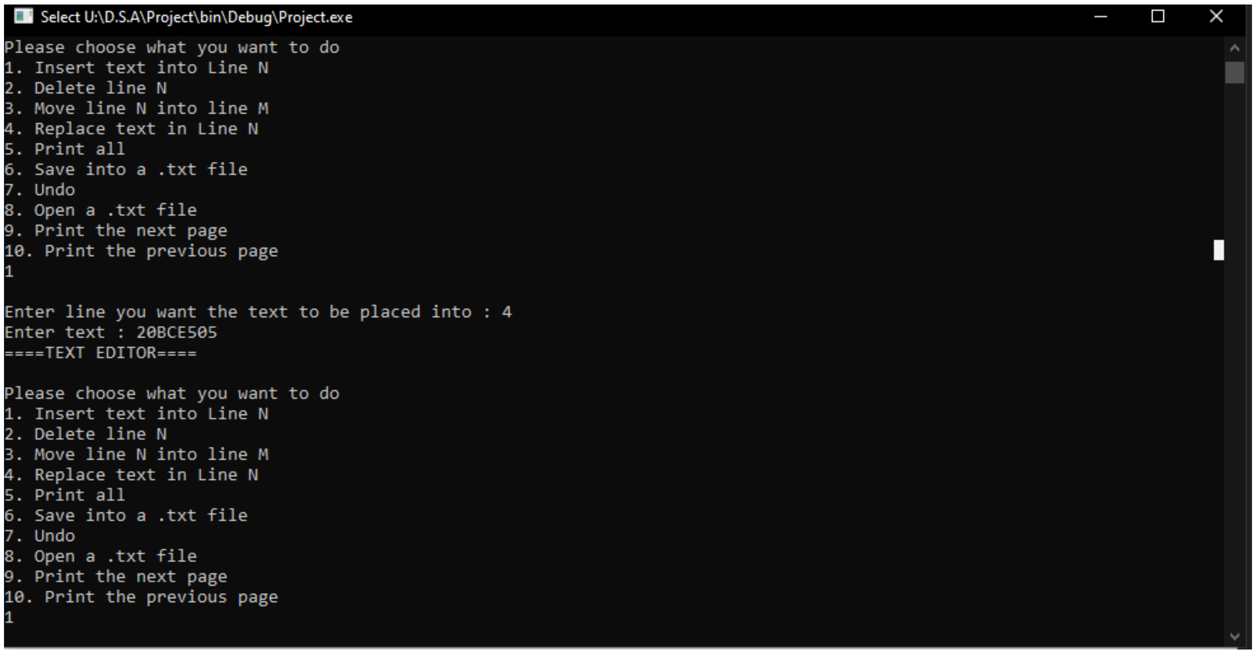
```
U:\D.S.A\Project\bin\Debug\Project.exe
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 1
Enter text : 20BCE526
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
```



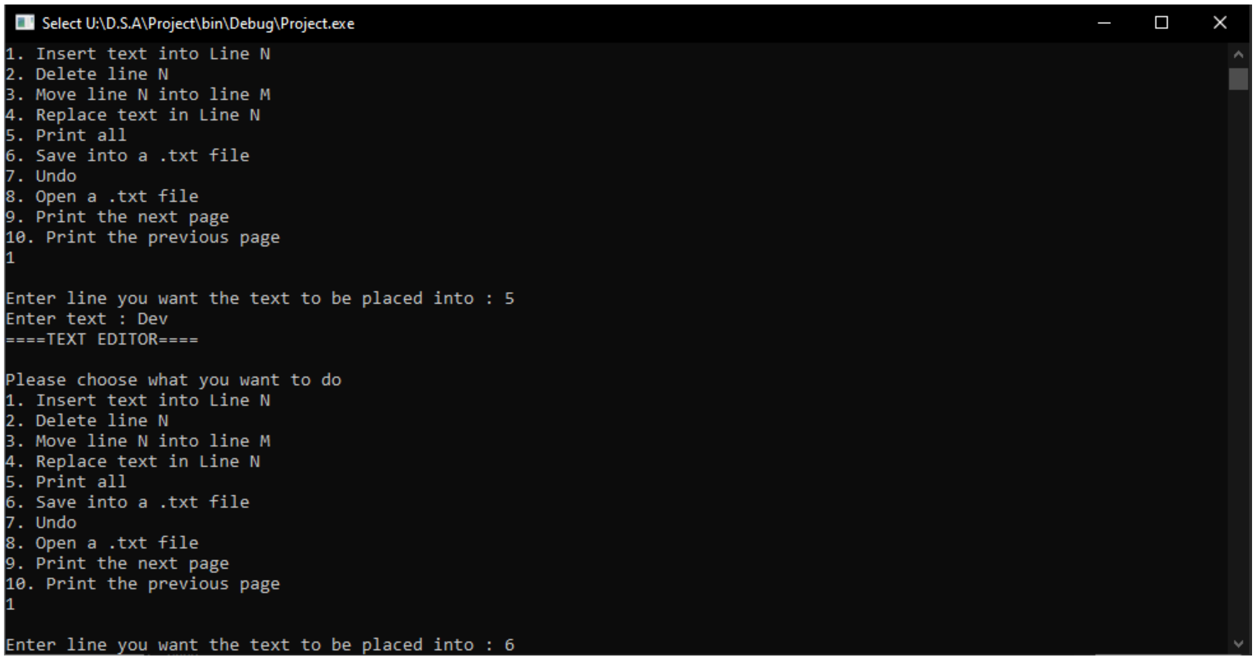
```
U:\D.S.A\Project\bin\Debug\Project.exe
Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 2
Enter text : Jainil
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1
```



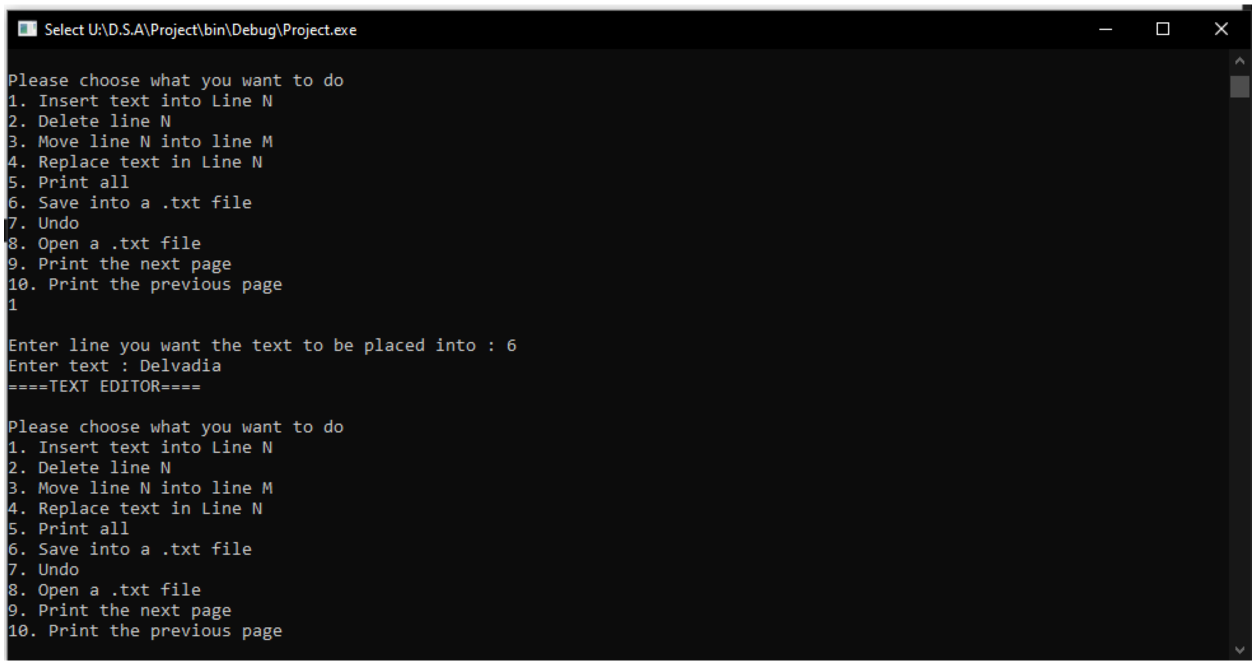
```
Select U:\D.S.A\Project\bin\Debug\Project.exe

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 3
Enter text : Solanki
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1
```



```
Select U:\D.S.A\Project\bin\Debug\Project.exe
Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 4
Enter text : 20BCE505
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1
```



```
Select U:\D.S.A\Project\bin\Debug\Project.exe
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 5
Enter text : Dev
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 6
```

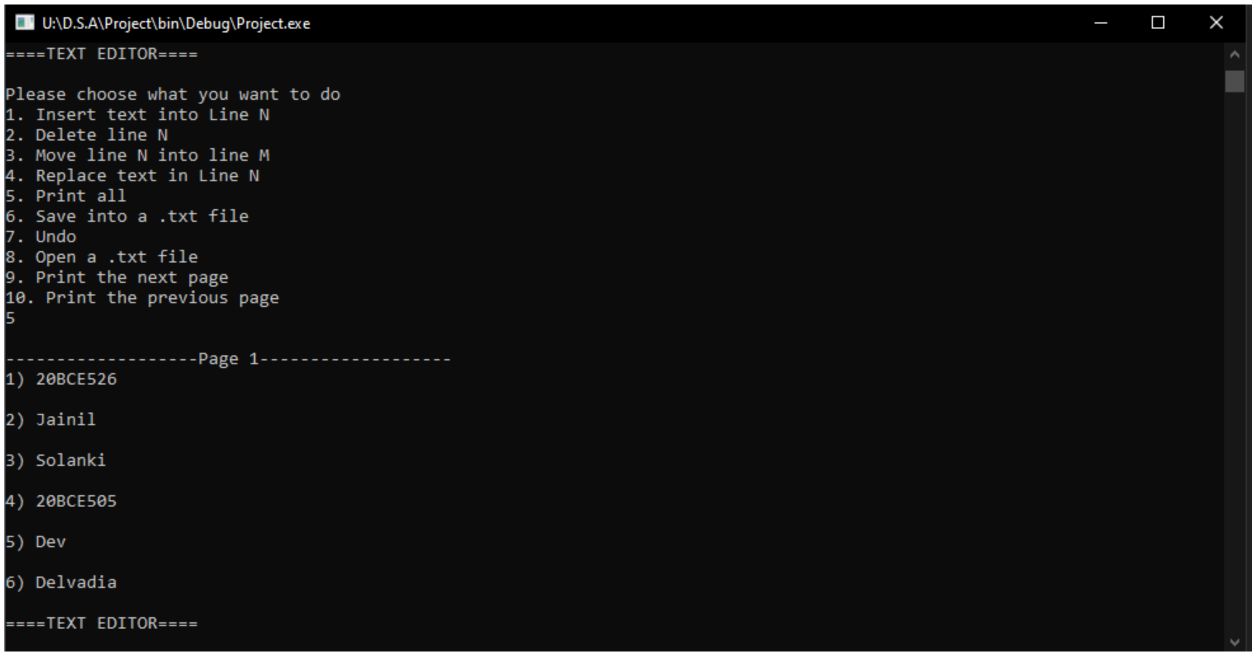


Select U:\D.S.A\Project\bin\Debug\Project.exe

```
Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
1

Enter line you want the text to be placed into : 6
Enter text : Delvadia
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
```

Output:

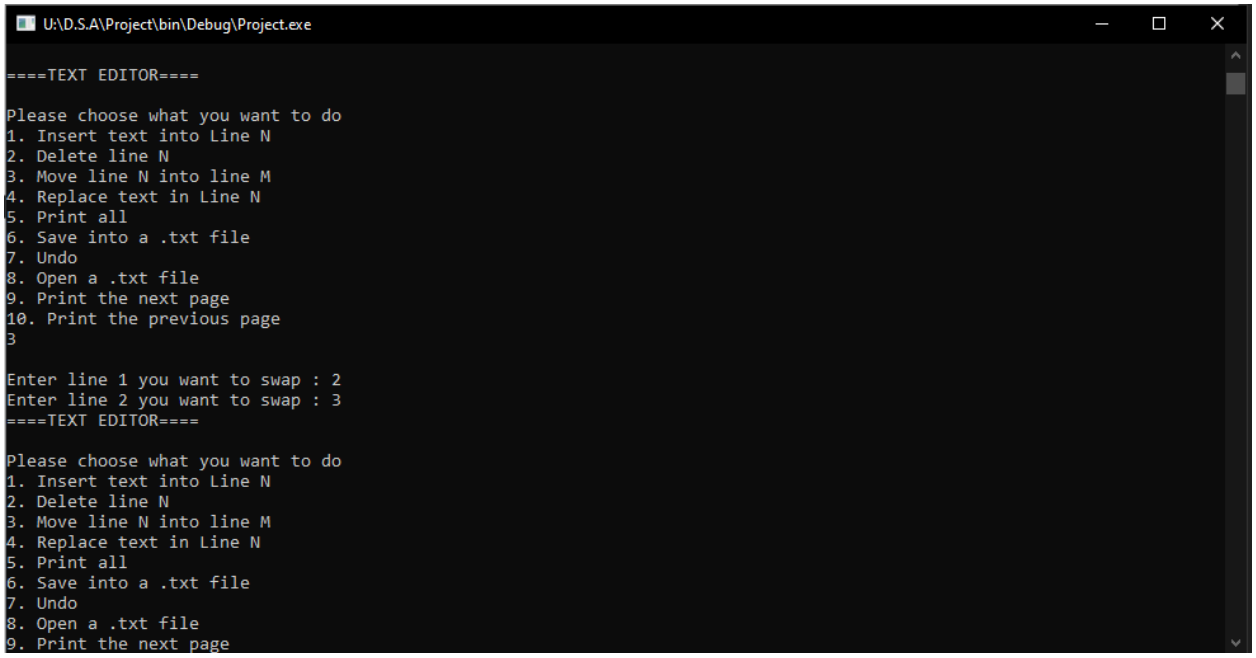
Printing All Data



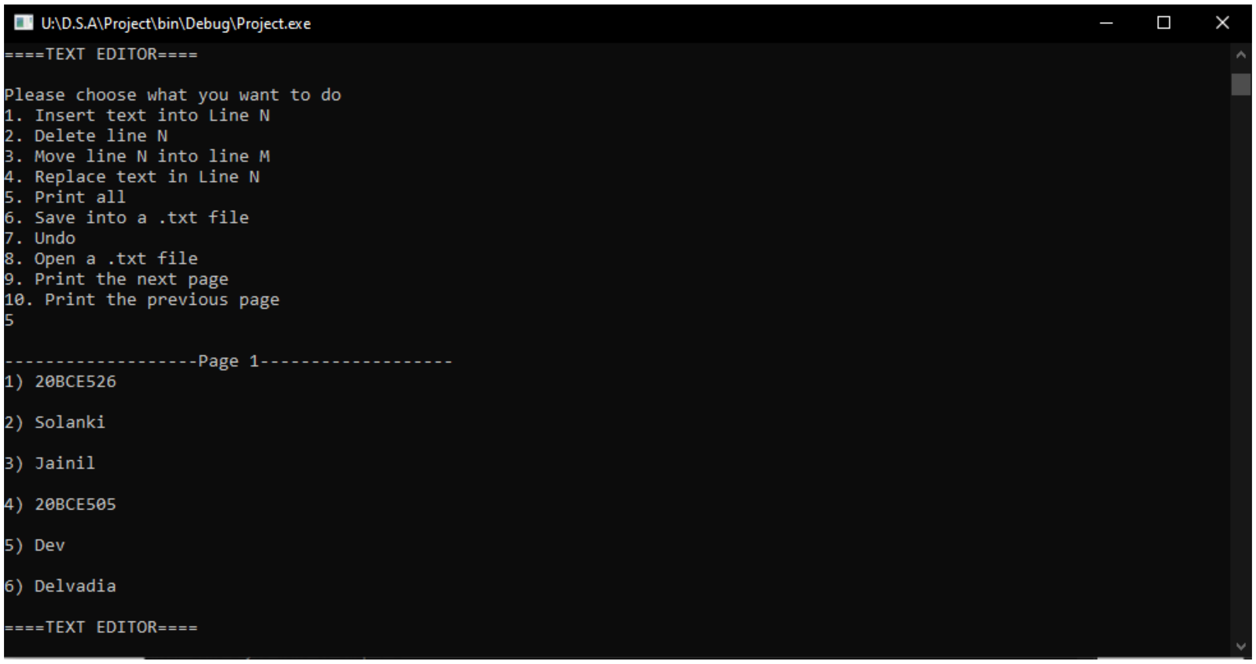
U:\D.S.A\Project\bin\Debug\Project.exe

```
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
5

-------------------Page 1-------------------
1) 20BCE526

2) Jainil

3) Solanki

4) 20BCE505

5) Dev

6) Delvadia

====TEXT EDITOR====
```

Swapping Lines

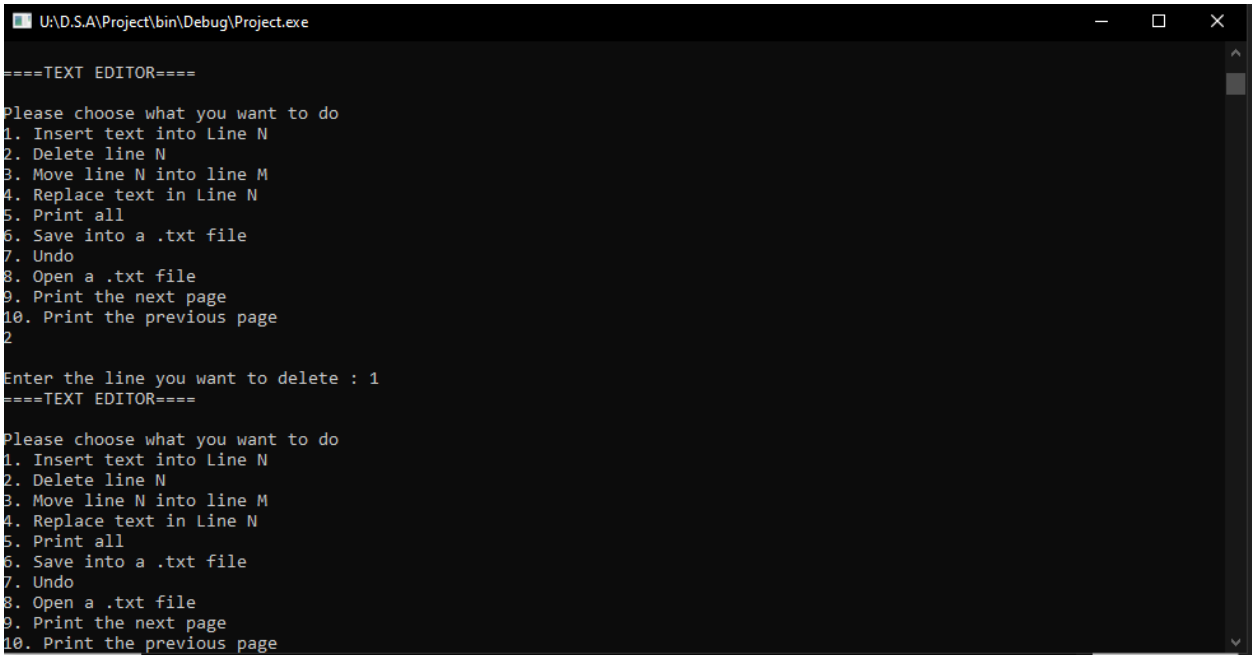


```
U:\D.S.A\Project\bin\Debug\Project.exe

====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
3

Enter line 1 you want to swap : 2
Enter line 2 you want to swap : 3
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
```

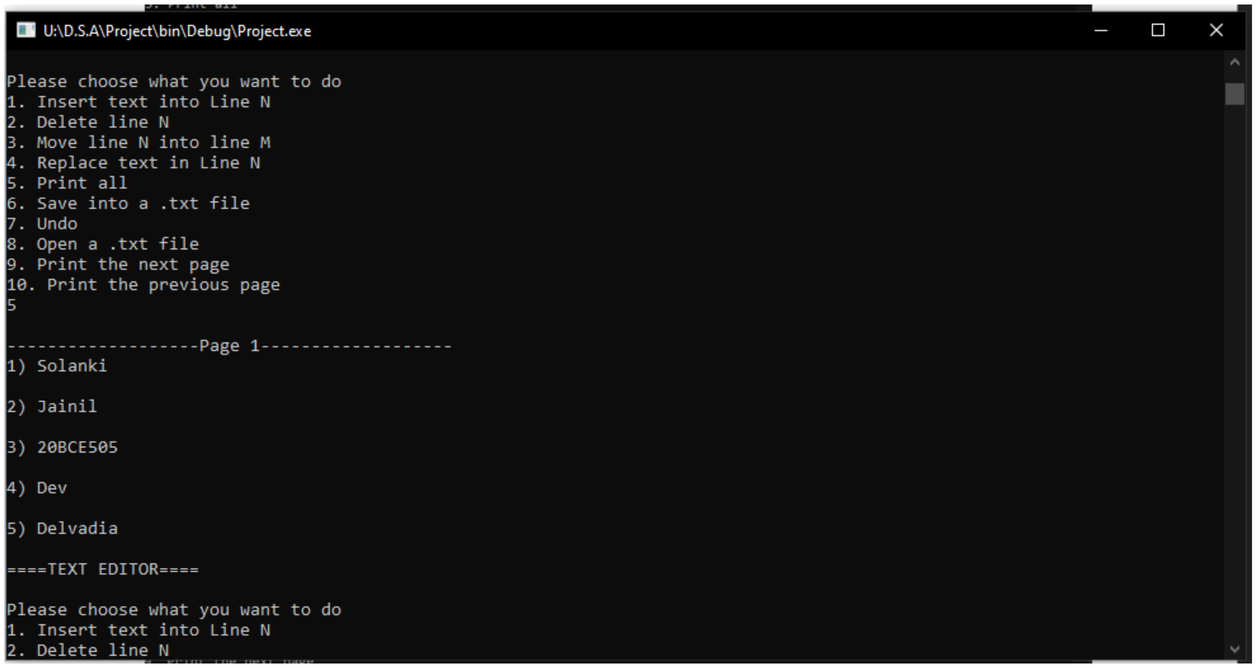


```
U:\D.S.A\Project\bin\Debug\Project.exe
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
5

-------------------Page 1-------------------
1) 20BCE526

2) Solanki

3) Jainil

4) 20BCE505

5) Dev

6) Delvadia

====TEXT EDITOR====
```

Deleting a line

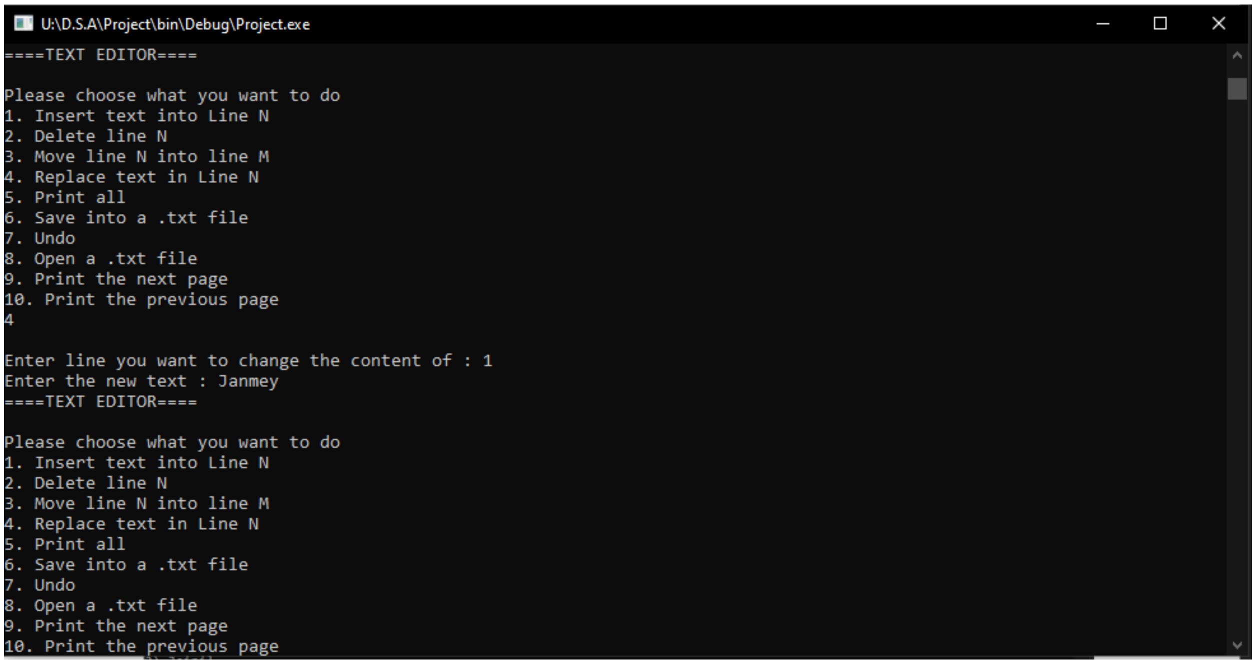


```
U:\D.S.A\Project\bin\Debug\Project.exe

====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
2

Enter the line you want to delete : 1
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
```



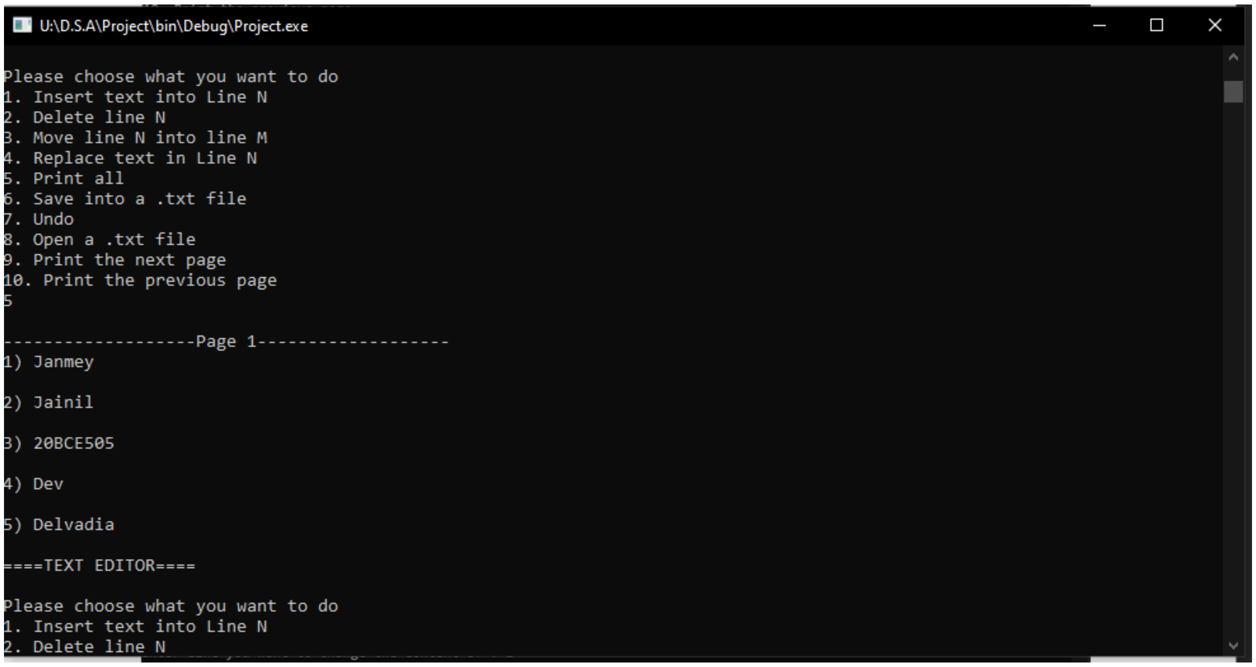
```
U:\D.S.A\Project\bin\Debug\Project.exe

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
5

-------------------Page 1-------------------
1) Solanki

2) Jainil

3) 20BCE505

4) Dev

5) Delvadia

====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
```

Replacing text in a line



U:\D.S.A\Project\bin\Debug\Project.exe

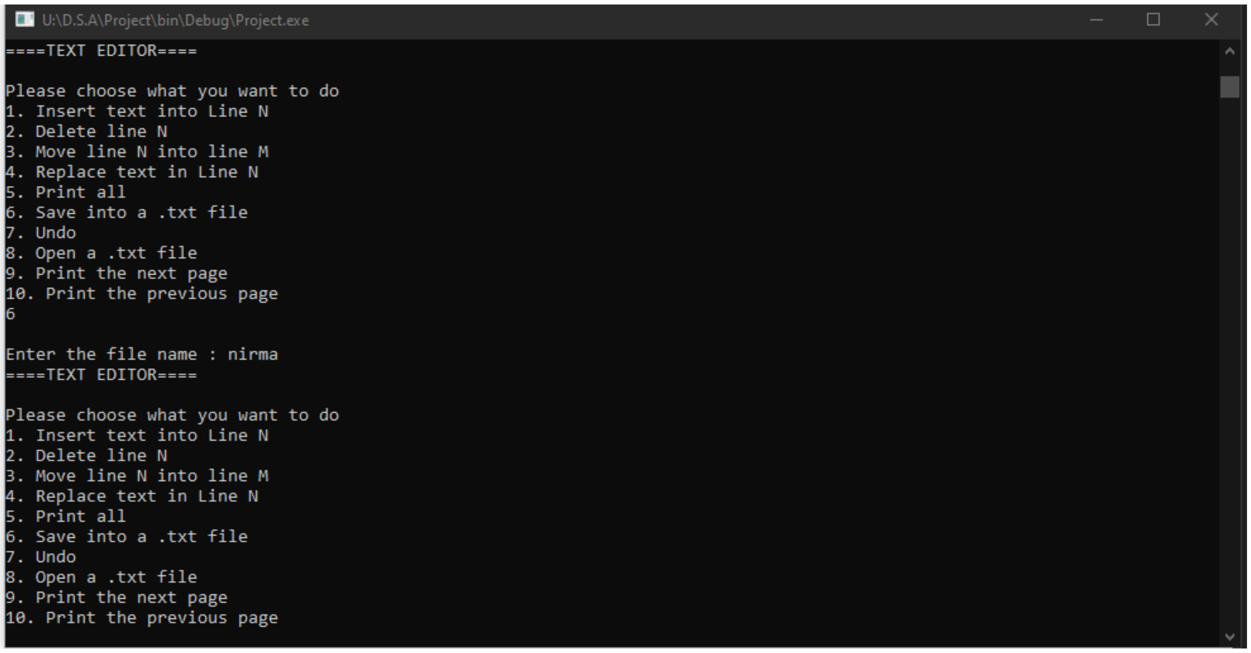
```
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
4

Enter line you want to change the content of : 1
Enter the new text : Janmey
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
```



U:\D.S.A\Project\bin\Debug\Project.exe

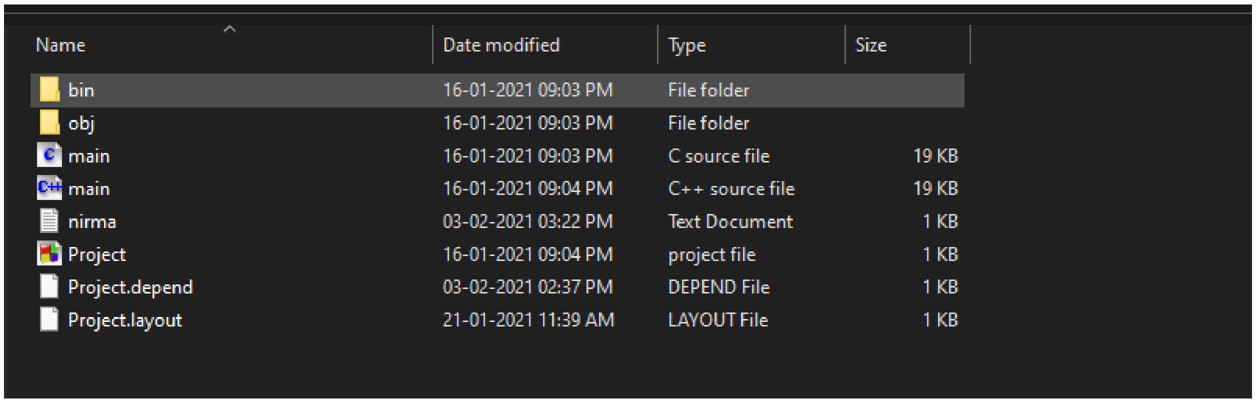
```
Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
5

-------------------Page 1-------------------
1) Janmey

2) Jainil

3) 20BCE505

4) Dev

5) Delvadia

====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
```

Saving in a file

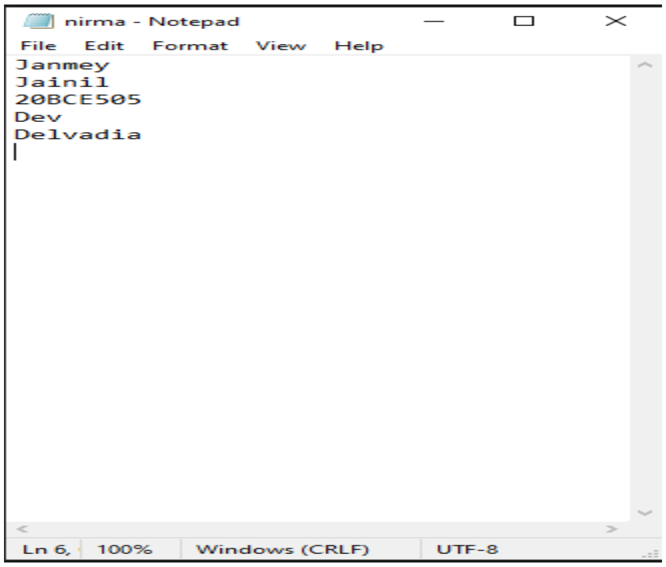


U:\D.S.A\Project\bin\Debug\Project.exe

```
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
6

Enter the file name : nirma
====TEXT EDITOR====

Please choose what you want to do
1. Insert text into Line N
2. Delete line N
3. Move line N into line M
4. Replace text in Line N
5. Print all
6. Save into a .txt file
7. Undo
8. Open a .txt file
9. Print the next page
10. Print the previous page
```



Name	Date modified	Type	Size
bin	16-01-2021 09:03 PM	File folder	
obj	16-01-2021 09:03 PM	File folder	
main	16-01-2021 09:03 PM	C source file	19 KB
main	16-01-2021 09:04 PM	C++ source file	19 KB
nirma	03-02-2021 03:22 PM	Text Document	1 KB
Project	16-01-2021 09:04 PM	project file	1 KB
Project.depend	03-02-2021 02:37 PM	DEPEND File	1 KB
Project.layout	21-01-2021 11:39 AM	LAYOUT File	1 KB



```
Janmey
Jainil
20BCE505
Dev
Delvadia
```

Conclusion: By implementing this project we learned deeply about the circular linked list data structure and how is it helpful in real life applications like this.