Data Structures B
FAST-NU, Lahore, Springs 2016

Homework 2

Storing Text as a Linked Structure

Due on Thursday January 29 11:55PM — Marked out of 50 points.

A simple text editor requires a data structure to store text (a sequence of ASCII characters). This data structure should provide certain simple functions to enable the basic functionalities of the editor, such as, inserting a new character anywhere in the text, deleting a part of the text, cutting a part of the text and pasting it elsewhere, finding sub-texts from a larger piece of text and replacing it with other text, and so on.

The text is stored by the data structure as a doubly linked list (this is case in such editors as notepad), where each node in the list contains a single character. In the following, I define a very basic skeleton for this data structure.

```
struct Node{
        char ch;
        Node * next, *prev;
};

class TextStructure{
    Node * head, * tail;
    int textSize;//num of characters in text
public:
    TextStructure();
    ~TextStructure();
};
```

You have to add the following functions to this class. In all these functions where a Node* type variable is passed as parameter you can assume that it has been passed by the text-editor application which keeps track of these pointers depending on which area of the text is currently being edited.

1) addCharAtEnd(char ch)
2) addCharAtFront(char ch)

3) addCharAfter(char, Node*ptr): add character in a new node after the node pointed to by ptr. If ptr is null, add it at the end of the list.
4) cutKCharsFrom(Node*ptr, int k, string & subtext): remove the ‘k’ characters starting from the character in the node ptr (this means actually removing the nodes from the list) and store these character in the string subtext.
5) findText(const string& str, bool direction, Node*& start, Node*&end): find the starting and ending nodes of the text stored in str in the list, in the specified direction (forward=true or backward=false).
6) findandReplaceText(const string & str1, const string & str2, bool direction): find all instances of the text stored in str1 in the specified direction (forward or backward) and replace each with nodes containing the characters of str2, again in the specified direction (forward or backward).
7) readTextFromFile(const string & filename): read the text from the file into the list.
8) operator << : print the text on the console
9) Default constructor, Copy Constructor and Desctructor
10) Assignment operator

That’s all.