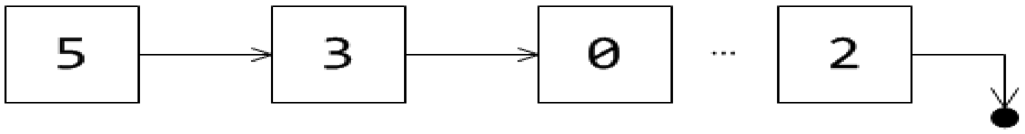
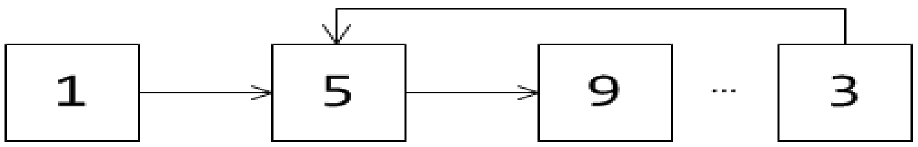
DSA Assignment # 3

Question 1: Implement a function "**hasCycle**" to find whether the given **singly linked list** has a cycle or not. The linked list's head will be passed as parameter. If there is a cycle, return true. Otherwise return false. (Use your implemented singly link list with head pointer)



Normal linked list



Linked list with a cycle

bool hasCycle(Node<T> *head)

Question 2: Implement a Node which has three data members, a character variable named "**data**" and two node pointers named "**next**" and "**prev**".

struct Node {
char data;
Node* next;
Node* prev;
};

Now using the above Node implementation, implement a doubly linked list named "**cString**" which can store a string. Each character of the string is stored in a node. For example, Pakistan would be stored as: P↔a↔k↔i↔s↔t↔a↔n→**NULL.** The nodes which contain the **first** character and the **last** character of the string, respectively, are data members of your cString class.

You must take care of **dangling pointers** and **memory leaks** in your functions. You must use **recursion** wherever specified. You are not allowed to change the function signature if it is already given. You have to implement the following member functions for cString class**:**

1. A default constructor which sets the string to null.

2. A copy constructor which deep copies another cString object.

3. A function named “assign” which uses **recursion** to copy a string of type char*, passed as parameter, into the linked list. **void assign (char * source)**

4. A function named “isPalindrome” which uses **recursion** to check whether the string is a palindrome or not. **bool isPalindrome()**

5. A function named “copy” which is passed as parameter a character pointer. It then uses **recursion** to copy the contents of the linked list into the memory pointed to by the pointer. Assume that the required memory to the pointer is already assigned. **void copy (char* destination)**

6. A function named “removeAll” which is passed as parameter another cString object. Your task is to remove all those characters from your cString object which are also present in the object passed as parameter.
   void removeAll (cString const& obj)
   for example, if removeAll is called on a cString object containing Pakistan, and the cString object passed as parameter contains tias, then word Pakistan will be changed to Pkn.

7. Destructor.

Question 3: You are required to develop a text editor. Implement these features using **Doubly linked list** that you have already studied.

Text editor like Notepad in Windows, is a very handy tool to work with text files. Usually text editors provide the following facilities:

1. Insert an alphabet/number
2. Delete next alphabet/number
3. Backspace:
   (delete previous alphabet/number)
4. Append at the end of line / file
5. Multiline text:
   (This means that you have to check if the input is ‘**\n**’. If it is **\n** then move to new line)
6. Movement of cursor **left** & **right**:
   (This means you have to get the **keystroke ascii** of left and right arrow. If the keystroke ascii matches that of left and right, then point to the required node accordingly. Attached below is the ascii code file for left & right arrows).
7. Save files as txt
8. Save current file
9. Open txt files

(Hint: Doubly linked list)
ASCII Code: https://www.alt-codes.net/

Question 4: Implement a **global function** “addStrings” which is passed three cString objects as parameters, named list1, list2, and result in the same order. The function first copies the contents of list1 into result and then appends the contents of list2 into result. Take care of dangling pointers and memory leaks.

Now run the following main program. **(For Question 2 & Question 3)**

```
int main() {
        cString str;

        char pakistan[100] = "pakistan",
                substr[100] = "ntias",
                ror[10] = "ror",
                rol[10] = "rol";

        str.assign(pakistan);
        cString str1(str);
        cString another;
        another.assign(substr);
        str1.removeAll(another);
        char destination[20];
        str1.copy(destination);
        cout << destination << endl;
        cString pal;
        pal.assign(ror);
        cout << pal.isPalindrome() << endl;
        pal.assign(rol);
        cout << pal.isPalindrome() << endl;
        cString result;
        addStrings(pal, str, result);
        result.copy(destination);
        cout << destination << endl;
}
```

Edited by: Tariq Aman