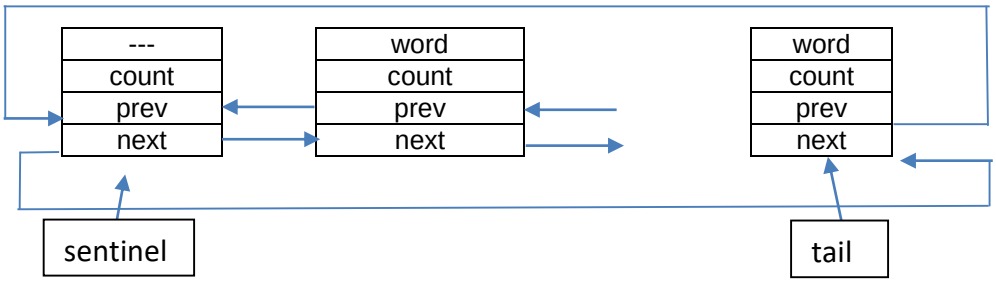
MA 251 Data Structures
Laboratory Assignment 4
21-08-2019

Note: Upload your programs to the server (deadline: 4:30 pm)

1. With doubly linked list, checking boundary conditions can be cumbersome. The solution is to use sentinels.

   In this assignment, first create *a doubly linked list* using sentinel.
   Each node has four fields – **word:** string, **count:** integer, **prev:** pointer to previous node and **next:** pointer to next node. Also, **a tail pointer** that points to the last node in the list.

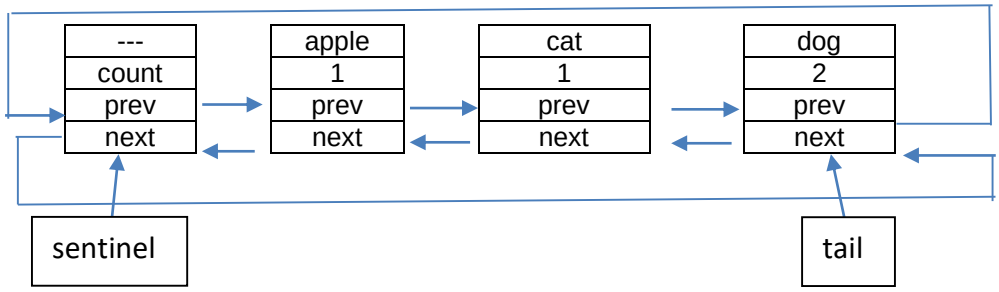


   Implement the following functions on the list:
   - *initialize():* Create an empty list with only sentinel node.
   - *pushFront(char *key):* Push a word (key) at the front of the list and set ***count=1***;
   - *char * popFront():* Remove a node from the front of the list
     char * *popBack*(): Remove a node from the tail of the list
     For both pop functions
     - print the key with count;
     - It removes the node **if *count==1***, otherwise decrement the count;
   - *int search(char *key):*
     - return *1*, if *key* == *word* (if key is matched with the existing word in a node);
     - return *2*, if *key* > *word* (check alphabetical order, e.g., dog>cat);
     - return *0,* otherwise.

2. (a) Create a header file **dlist.h**, to initialize, create and search a doubly linked list using the above functions.
   (b) Write a program to insert words into a double linked list in alphabetical order. You can use a function add(char *key). *(Use the header file and necessary functions of assignment 1)*

   When a *key* is inserted, the *key* value is compared with the existing words in the list (i.e. nodes).
   a. If the *word* in the node is matched with the *key*, increment *count* by *1*;
   b. If the *word* in the node is less than *key*, move to next node in the list, else, create a new node and insert it before the last visited node.

For example, consider the words – apple, dog, cat and dog.



The first node (after sentinel) will be apple then dog. When the word cat is inserted, the search will terminate before the node containing word dog. Thus cat will be inserted before dog. When the fourth key dog will be added, it will simply increment the count.

Deliverables:

In order to test Assignment 1, you will need to write main() which calls all the procedures. A typical Input/Ouput of the assignments are shown below. Take the screenshot of the output of your program and upload it to the server along with the source code

Input/Output for Question 1:

Expected output:

$./a.out

Enter 1: to push at the front, 2 to pop from front, 3 to pop from back and 4 to search a key

Input: 1, apple

Input: 1, banana

Input: 1, guava

Input: 3

Output: apple

Input: 2

Output: guava (and so on)

Input/output for question 2(b)

Expected output:

$./a.out

Enter 1: to add key, 2 to pop from front, 3 to pop from back and 4 to search a key

Input: 1, guava
Input: 1, apple
Input: 1, banana
Input: 1, guava
Input: 1, banana
Input: 3
Output:
guava - 1 (The output also specifies the remaining count)
Input: 1, apple
Input: 1, guava
Input: 2
Output:
apple - 1
Input: 2
Output:
apple – 0 (node is deleted)
and so on ……