CS 2123 Data Structures

Programming Project 3

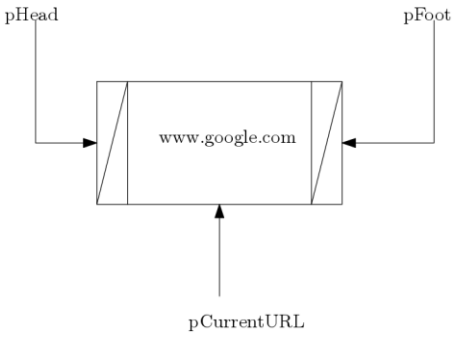
In this project, we will be implementing the functionality of the forward/back buttons of a web browser. For example, if we go to: www.google.com, then to www.yahoo.com, and then to www.utsa.edu, the browser would store these websites in order in memory. If we want to go back, then we would be taken back to yahoo. If we then clicked forward, we would then be taken back to utsa.edu.

We will call the data structure to handle the functionality a **BrowserList.** A browser list will consist of a **DoublyLinkedList**, where each node in the linked list will store a webpage URL and will have a pointer to the node containing the webpage that comes before it in the list and a pointer to the node containing the webpage that comes after it. The BrowserList will also have a pointer the node containing the URL of the webpage we are currently viewing.

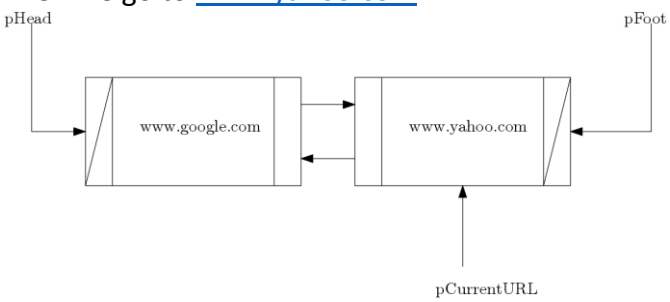
Example:

We first create an empty BrowserList which will contain an empty DoublyLinkedList.

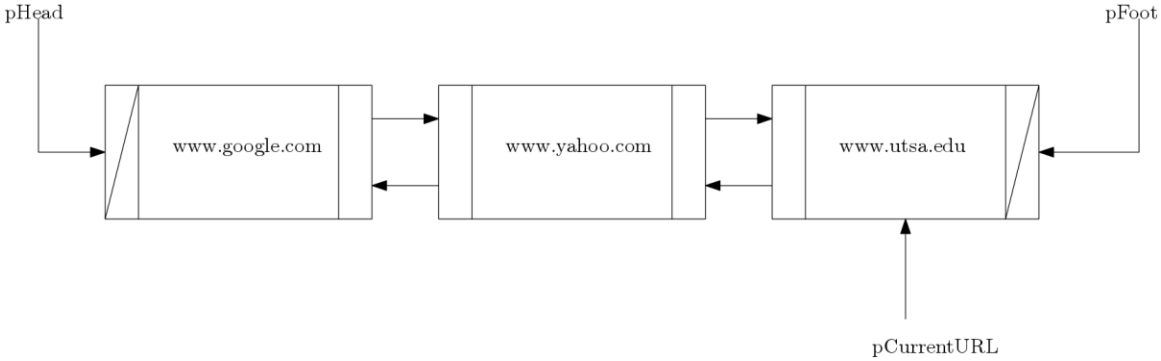
1) Then we go to www.google.com:



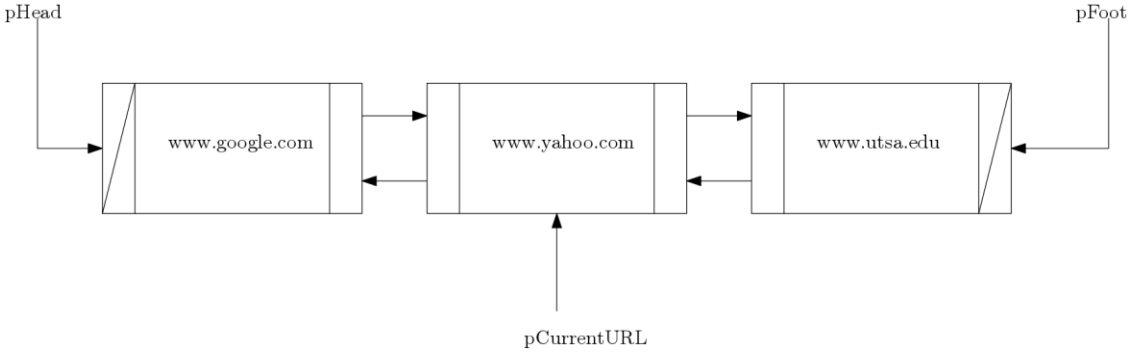
2) Then we go to www.yahoo.com:



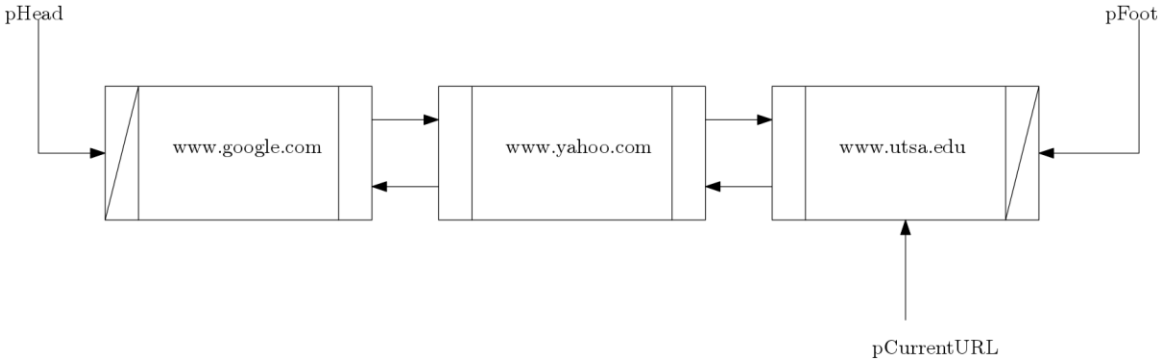
3) Then we go to www.utsa.edu:



4) Then we go “back”:



5) Then we go “forward”:



Files provided in the project:

1) DoublyLinkedList.h. This file contains the structs needed for the DoublyLinkedList class as well as the prototypes for several functions that you will need to implement. In particular you need to implement the following functions in the file DoublyLinkedList.c:
    a. **DoublyLinkedList newDoublyLinkedList()** which should allocate the memory for a new doubly linked list, initialize the variables, and return the address.
    b. **void freeDoublyLinkedList(DoublyLinkedList list)** which should free the linked list (including any nodes currently in the list).
    c. **NodeDL *allocateNode(Element value)** which should allocate memory for a new node, store "value" inside this node, and return the address of the node.
    d. **void append(DoublyLinkedList list, Element value)** which should add a new node (which stores value) to the end of the list. Notice that often one would want to insert into the middle of a linked list, but for this project it suffices to only insert at the end of a linked list.

2) BrowserList.h. This file contains the struct for the BrowserList as well as the prototypes for the functions that you will need to implement. In particular you need to implement the following functions in the file BrowserList.c:
    a. **BrowserList newBrowserList()** which allocates the memory for a new BrowserList, initializes its variables, and returns its address.
    b. **void freeBrowserList(BrowserList browserList)** which frees the memory used for the BrowserList (including its DoublyLinkedList).
    c. **void goToURL(BrowserList browserList, Element element)** which will add a node in the DoublyLinkedList that stores element after pCurrentURL (any nodes that were in the list after pCurrentURL should be removed from the list) and updates pCurrentURL to point to the new node.
    d. **int back(BrowserList browserList)** which moves pCurrentURL "back" one page. It returns TRUE if this completed successfully and returns false otherwise (e.g. the node doesn't exist).
    e. **int forward(BrowserList browserList)** which moves pCurrentURL "forward" one page. It returns TRUE if this completed successfully and returns false otherwise (e.g. the node doesn't exist).
    f. **void printBrowserList(BrowserList browserList)** which prints the contents of the BrowserList with one URL per line, placing *'s around the current URL. If we printed the final list in the example, we would have:

        www.google.com
        www.yahoo.com
        www.utsa.edu

3) p3Input.txt. This file contains a sample input file that you should process. Each line will contain either BACK, FORWARD, PRINT, or a URL (don't worry about checking the URL for errors). If the line contains a URL, then you should call the goToURL() function to update the BrowserList. If the line contains one of the other terms, then you should call the corresponding function. The input file should be passed to the program like so: **./p3 < p3Input.txt**

4) abc123p3.c. Rename this file to your abc123. This file is mostly empty right now. It contains the main() function that you should complete. In main, you should read a line from p3Input.txt (you can do this however you would like), detect if it is a URL or some other command, and update the BrowserList accordingly.

5) Makefile. Update the makefile to reflect your abc123. Compile using **make p3**.

Submitting:

Please submit to the blackboard dropbox a zipped folder containing each of the files we have provided you along with the new DoublyLinkedList.c and BrowserList.c files that you create.