VGP240 Assignment #5

W. Dobson 11-1-2017

Create a class called `DlinkedList` stored in an `#include` file called **`dlinkedlist.h`** that can be used to create a *doubly linked list* of nodes of the following structure:

```
struct NodeType{
     string    name;
     int       value;
     NodeType  *prev;
     NodeType  *next;
}
```

1. The class will have have *private* pointers (of `NodeType`) that are called:

```
NodeType  *listhead;
NodeType  *listtail;
```

There must be a *constructor* that initializes the *head* and *tail* pointers to NULL.

---

2. The class should include *public* functions:

`void  pushtail( NodeType *ptr)` that inserts/pushes a node referenced by the `ptr` at the tail of the list. This function will dynamically allocate a new node using *new* .

`void  poptail(  NodeType *ptr)` returns the data contents of the last node in the structure referenced by the pointer and removes it from the list using *delete* .

`void  insert(  NodeType *newptr, NodeType *nodeptr )` inserts a new node referenced by `newptr` into the list after the node referenced by `nodeptr` .

`void  delete( NodeType *nodeptr )` removes a node referenced by `nodeptr` from the list. Be careful when removing the head or tail nodes to adjust the private *listhead* and *listtail* pointers correctly if one of these are deleted.

`void  pushhead( NodeType *ptr)` that inserts/pushes a node referenced by the `ptr` onto the head of the list. This function will dynamically allocate a new node using *new* .

`void  pophead(  NodeType *ptr)` returns the data contents of the head node data in the structure referenced by the pointer and removes it from the list using *delete* . Be sure to set the listhead pointer to NULL when reading the last node from the list.

`int  length()` returns the number of nodes in the list. Note if the head and tail pointers match this should be 0 otherwise use a loop to traverse the loop and count the nodes.

`NodeType * tail()` returns a pointer to the last node *(tail).*

`NodeType * head()` returns a pointer to the first node *(head).*

*** These functions from assignment 4 should be moved outside of the class as external standalone functions in your include file and should be changed to take a pointer input for the head of the list.:

`void  displaylist(NodeType *headptr)` uses an appropriate loop to traverse the list printing out the *name* and *value* to the console for each node.

`NodeType * searchlist(NodeType *headptr, string sname)` performs a linear search of the list returning a pointer to the node with a string in its *name* element that matches the input string *sname.*

---

3. Now write a *main()* code body that includes **`dlinkedlist.h`** and instantiates a linked list object then uses the `pushtail()` function to add 3 nodes to the list. Then use the `poptail()` to read data out of the linked list until the linked list is empty. Use the .length() function to cout the number of remaining items in the linked list after each *pop* operation. In this way the list is used as a *stack* or LIFO.

Now repeat the process populating the list using `pushtail()` then read out the data with the `pophead()` essentially using the list as a queue or FIFO. This can be done with simple in-line test code.

Also write some code to exercise the `.insert()` and `.delete()` class functions.

Use the `displaylist()` function to show changes made to the linked list once it is populated.