CPSC 2430 Data Structures

Homework Assignments #2

11 points

1. Problem

You are required to design an Abstract Data Type (ADT) called **Deque** (double-ended queue). A Deque is a linear data structure that allows insertion and deletion of elements from **both the front and the back**. We assume that elements stored in Deque are integers.

For this assignment, you must:

1. Implement the Deque using a **doubly linked list** (not arrays or other data structures).
2. Provide operations as shown in Table 1.

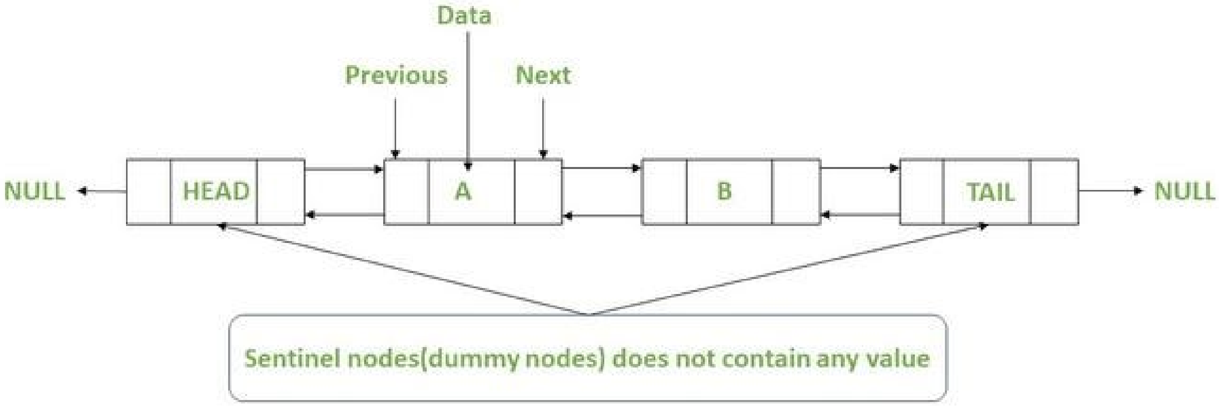
.

Table 1 Required Operations

`Deque()`	Default constructor: constructs an empty container, with no elements.
`Deque(const Deque& x)`	Copy constructor: constructs a container with a copy of each of the elements in *x*, in the same order.
`~Deque()`	Destructor: destroys all container elements, and deallocates all the dynamic memory allocated by the Deque container.
`Deque& operator=(const Deque& x)`	Assignment operator: assigns new contents from *x* to the container, replacing its current contents, and modifying its accordingly.
`unsigned int size() const`	Returns the number of elements in the Deque container.
`bool empty() const`	Returns whether the Deque container is empty (i.e. whether its size is 0).
`int front() const`	Returns the value of the first element in Deque container. Assume that this operation will not be performed if the Deque container is empty.
`int back() const`	Returns the value of the last element in Deque container. Assume that this operation will not be performed if the Deque container is empty.
`void push_back(const int& val)`	Adds a new element at the end of the Deque container, after its current last element. The content of *val* is copied to the new element.

	This effectively increases the container size by one.
`void push_front(const int& val)`	Inserts a new element at the beginning of the Deque container, right before its current first element. The content of *val* is copied to the inserted element. This effectively increases the container size by one.
`void pop_back()`	Removes the last element in the Deque container, effectively reducing the container size by one. This destroys the removed element. If the container is empty, no operation is performed.
`void pop_front()`	Removes the first element in the Deque container, effectively reducing its size by one. This destroys the removed element. If the container is empty, no operation is performed.

You may consider using a doubly linked list with sentinel (dummy) nodes at both ends in your implementation. This approach can simplify your work significantly. An example of such doubly-linked lists is shown below (also briefly discussed in class).



2. The Client (or Driver) Program

A `client.cpp` will be provided for this assignment. However, feel free to implement your own `client.cpp` during testing and debugging. Please note that the provided `client.cpp` needs to be included in your final submission.

3. The Makefile

You are required to provide a Makefile for this assignment. You may adapt the Makefile used in HW #1, as the modifications should be straightforward.

4. Submission

You need to submit the following files:

- deque.h
- deque.cpp
- client.cpp: the provided client program.
- Makefile

Before submission, you should ensure your program has been compiled and tested (extensively) in cs1.seattleu.edu. Your assignment receives zero if your code cannot be compiled and executed.

You can submit your program multiple times before the deadline. The last submission will be used for grading.

To submit your assignment, you should follow two steps below (assuming your files are on cs1.seattleu.edu):

- Pack all your files into a package named ***hw2.tar***

```
tar -cvf hw2.tar deque.h deque.cpp client.cpp Makefile
```

- Submit the package ***hw2.tar*** as the second programming assignment ***HW2***

```
/home/fac/zhuy/class/submit2430 HW2 hw2.tar
```

The message similar to the following one will be displayed if the submission is successful.

```
=======================Copyright(C)Yingwu Zhu============================
Sun Mar 28 23:01:52 PDT 2021
Welcome testzhuy!
You are submitting hw1.tar for assignment HW1.
Transferring file.....
Congrats! You have successfully submitted your assignment! Thank you!
Email: zhuy@seattleu.edu

========================================================================
```

5. Grading Criteria

Label	**Notes**
[1] Submission (1 pt)	All required files are submitted.
[2] Makefile (1 pt)	Makefile compiles the code and generates the executable.
[3] Presentation (1 pt)	Clean, well-commented code. No unsolicited output messages such as testing & debugging messages.
[4] Functionality (8 pts)	Default and copy constructors: 1 pt Assignment operator: 1 pt empty() and size(): 1 pt front() and back(): 1 pt push_back(): 1 pt push_front(): 1 pt pop_front() & pop_back(): 1 pt Destructor: 1 pt
[5] Overriding policy	If the code cannot be compiled or executed (segmentation faults instantly, for instance), it results in zero point for Functionality. No further investigation will be conducted on your program.
[6] Late submission	Please refer to the Syllabus for details.