Computer Engineering 12
Project 6: Still not Playing with a Full Deque

Due: Saturday, June 8th at 5:00 pm

1 Introduction

Professor Gosheim Loony has decided to rewrite his maze game in C++, but did not have time to finish it. (You think he would learn better time management.) Once again, you will need to write a deque ADT that conforms to Professor Loony's interface, only this time you must write it in C++.

2 Interface

The interface to your abstract data type must provide the following operations:

- `Deque::Deque();`
  constructor for the deque
- `Deque::~Deque();`
  destructor for the deque
- `int Deque::size();`
  return the number of items in the deque
- `void Deque::addFirst(int x);`
  add x as the first item in the deque
- `void Deque::addLast(int x);`
  add x as the last item in the deque
- `int Deque::removeFirst();`
  remove and return the first item in the deque, which must not be empty
- `int Deque::removeLast();`
  remove and return the last item in the deque, which must not be empty
- `int Deque::getFirst();`
  return, but do not remove, the first item in the deque, which must not be empty
- `int Deque::getLast();`
  return, but do not remove, the last item in the deque, which must not be empty

3 Implementation

As required by Professor Loony, you will use a circular, doubly-linked list with a sentinel or *dummy* node. The sentinel node is always the first node in the list, but does not itself hold data. All operations except the destructor are required to run in $O(1)$ time. As a starting point, you can use either the deque implementation in C that you wrote or the implementation that Professor Loony wrote.

4 Submission

Create a directory called `project6` to hold your solution. Call the source file for the implementation `deque.cpp` and the header file `deque.h`. Submit a `tar` file containing the `project6` directory using the online submission system.

5 Grading

Your implementation will be graded in terms of correctness, clarity of implementation, and commenting and style. Your implementation *must* compile and run on the workstations in the lab. The algorithmic complexity of each function in your deque abstract data type *must* be documented.