Если понадобится, в этой задаче можно использовать std::vector и std::array. Другие стандартные контейнеры использовать нельзя. Необходимо реализовать шаблонный класс Deque<T> - сильно упрощенный аналог класса std::deque<T>. Тип T не обязан иметь конструктор по умолчанию, чтобы его можно было хранить в Deque.
Ваш Deque должен обладать следующей функциональностью:

- Конструкторы: по умолчанию, конструктор копирования, конструктор от int, конструктор от (int, const T&). Оператор присваивания deque.
- Метод size(), возвращающий текущий размер контейнера.
- Обращение по индексу: квадратными скобками (без проверок выхода за границу) и at (кидающее std::out_of_range). Должно работать за гарантированное O(1).
- Методы push_back, pop_back, push_front, pop_front. Должны работать за амортизированное O(1).
- Должен быть внутренний тип iterator (с маленькой буквы, в лучших традициях STL). Этот тип, помимо очевидного, должен поддерживать:
  - Инкремент, декремент
  - Сложение с целыми числами
  - Сравнение < > <= >= == !=
  - Взятие разности двух итераторов
  - Разыменование (унарная звездочка), результатом является T&
  - Оператор ->, результатом является T*
- Также должен быть константный итератор const_iterator. Отличие его от обычного iterator в том, что он не позволяет менять лежащий под ним элемент. Конверсия (в том числе неявная) неконстантного итератора в константный допустима, обратно - нет. Реализовать константный итератор надо так, чтобы его код не являлся копипастой кода обычного iterator.
- **Операции push_back, push_front, pop_back и pop_front должны не инвалидировать указатели и ссылки на остальные элементы дека. Операции pop_back и pop_front, кроме того, должны еще и не инвалидировать итераторы.**
- Методы begin, cbegin, end и cend, возвращающие неконстантные и константные итераторы на начало и на “элемент после конца” контейнера соответственно. Если сам контейнер константный, то методы begin и end тоже возвращают константные итераторы. Декремент end должен давать итератор на последний элемент, вычитание целых чисел из end также должно корректно работать и давать итераторы на соответствующие элементы.
- reverse-итераторы, а также методы rbegin, rend, crbegin, crend.
- Метод insert(iterator, const T&), делающий вставку в контейнер по итератору. Все элементы справа сдвигаются на один вправо, вставка работает линейное время.
- Метод erase(iterator), удаляющий элемент из контейнера по итератору. Все элементы справа сдвигаются на один влево, удаление работает линейное время.

- **Все методы вашего Deque должны быть строго безопасны относительно исключений.** Это значит, что в случае исключения в конструкторе или операторе присваивания типа T во время выполнения какого-либо метода дека, последний должен вернуться в исходное состояние, которое было до начала выполнения метода, и пробросить исключение наверх в вызывающий код.

В вашем файл должна отсутствовать функция main, а сам файл должен называться deque.h. Ваш код будет включен посредством include в программу, содержащую тесты.