Implement a Double-ended Queue

Brief Description:

Develop a double-ended queue (Deque) that holds strings using only arrays. The queue should automatically grow and shrink depending on the usage. Its size should double when it is full and shrink to half its size when the its less than $\frac{1}{4}^{th}$ full.

Your deque should implement the following functions.

- enqueue_front - Inserts an element at the front of the queue
- enqueue_back - Inserts an element at the back of the queue
- dequeue_front - Deletes the element at the front of the queue
- dequeue_back - Deletes the element at the back of the queue
- empty - checks to see if the queue is empty
- size - returns the size of the queue
- appropriate constructors and destructors

Deliverables

- Well documented source code.
- Document containing test cases and test results.
- User documentation explaining how an end user can use your implementation of the data structure.

Interface

```
class Deque{
private:
  std::string *queue;      // The array which holds the queue.
  int num_elements;        // The number of elements in the queue
  int size_of_queue;       // The capacity of the queue
  int front;               // Points to the front of the queue
  int back;                // Points to the back of the queue
  //Constructor
  Deque() {

  }
  //Destructor
  ~Deque() {
  }
  // Inserts the element at the front of the queue.
  void enqueue_front(std::string item) {
  }

  // Inserts the element at the back of the queue
  void insert_back(std::string item) {
  }

  // Deletes the element at the front of the queue.
  std::string dequeue() {
  }

  //Deletes the element at the back of the queue.
  std::string pop() {
  }

  //Returns the number of elements in the queue.
  int size() {
  }

  //Tells whether the queue is empty or not.
  bool empty() {
  }
};
```

Interface.cpp

Grading Rubric

Criteria	Points
Code compiles without error	10
Code gives correcrt answers to test cases	40
Testing (Writing proper appropriate cases and testing the code)	20
Proper code documentation	10
Writing user documentation	10
Following good programming practices	10
Total	100