```
#ifndef QUEUE_H_
#define QUEUE_H_

template<typename Item_Type>
class queue {
 public:
  queue() : capacity(DEFAULT_CAPACITY), num_items(0),
    front_index(0), the_data(new Item_Type[DEFAULT_CAPACITY]) { }

  void push(const Item_Type& item) {
    if (num_items == capacity)
      reallocate();
    num_items++;
    size_t rear_index = (front_index + num_items - 1) % capacity;
    the_data[rear_index] = item;
  }

  Item_Type& front() {
    return the_data[front_index];
  }

  const Item_Type& front() const {
    return the_data[front_index];
  }

  void pop() {
    front_index = (front_index + 1) % capacity;
    num_items--;
  }

  bool empty() const {
    return num_items == 0;
  }

  size_t size() const {
    return num_items;
  }

  // Copy constructor, assignment operator, destructor, and swap
  // are similar to the vector class (but omitted in this handout).

 private:
  void reallocate() {
    size_t new_capacity = 2 * capacity;
    Item_Type* new_data = new Item_Type[new_capacity];
    size_t j = front_index;
    for (size_t i = 0; i < num_items; i++) {
      new_data[i] = the_data[j];
      j = (j + 1) % capacity;   // still based on old capacity
    }
    delete[] the_data;
    the_data = new_data;
    front_index = 0;
    capacity = new_capacity;
  }

  // Data fields
  size_t capacity;     /* The current capacity of the data array */
  size_t num_items;    /* The number of items in the queue */
  size_t front_index;  /* The index of the front of the queue */
  Item_Type* the_data; /* Pointer to the array containing the data */
  static const size_t DEFAULT_CAPACITY = 10;

}; // End class queue

#endif
```