```
// file: queue.h
#ifndef QUEUE_H
#define QUEUE_H
#include "queueADT.h"
#include <iostream>

using namespace std;

template<class Type>
class queue: public queueADT<Type> {
    private:
        int maxSize;
        int queueFront, queueRear;
        Type *list;

        void copyQueue(const queue<Type>& other);

    public:
        //the function body is placed in-line for easy reading
        queue(int size=100) {

            maxSize = size;
            queueFront = queueRear = 0;
            list = new Type[maxSize];
        }

        queue(const queue<Type>& other) {
            maxSize = 0;
            list = NULL;
            copyQueue(other);
        }

        ~queue() {
            delete [] list;
        }

        void initialize()
        {
            queueFront = queueRear = 0;
        }

        bool empty() const {
            return queueFront == queueRear;
        }

        bool full() const
        {
            return (queueRear + 1) % maxSize == queueFront;
        }

        int size() const {
            return (maxSize + queueRear - queueFront) % maxSize;
        }

        const queue<Type>& operator=(const queue<Type>& other) {
            if (this != &other)
                copyQueue(other);
```

```
58              return *this;
59          }
60
61          void push(const Type& item) {
62              if (!full()) {
63                  queueRear = (queueRear + 1) % maxSize;  // wrap to circular index
64                  list[queueRear] = item;
65              } else
66                  cerr << "Queue overflow" << endl;
67          }
68
69          Type& front() {
70              //precondition: queue is not empty
71              return list[(queueFront + 1) % maxSize];
72          }
73
74          void pop() {
75              if (!empty())
76                  queueFront = (queueFront + 1) % maxSize;
77              else
78                  cerr << "Queue underflow" << endl;
79          }
80
81  };
82
83
84  //Implementation of copyQueue()
85  template<class Type>
86  void queue<Type>::copyQueue(const queue<Type>& other) {
87      if (maxSize != other.maxSize) {
88          if (list != NULL)
89              delete [] list;
90          maxSize = other.maxSize;
91          list = new Type[maxSize];
92      }
93
94      queueFront = other.queueFront;
95      queueRear = other.queueRear;
96
97      int i = queueFront;
98      while (i != queueRear) {
99          i = (i + 1) % maxSize;
100         list[i] = other.list[i];
101     }
102 }
103
104 #endif
105
106
107
```