//CS311 Yoshii

//INSTRUCTION:

// QUEUE class - header file template based on Notes-2A

// Look for \*\* and complete them

// =====================================================

// HW#: HW1P2 queue

// Your name: Alexander Sadeghipour

// Compiler: g++ compiler

// File type: queue header file queue.h

//=====================================================

using namespace std;

#include<string>

//----- Globally setting up the aliases ----------------

typedef string el\_t; // el\_t is an alias for the data type

// el\_t is unknown to the client

const int MAX\_SIZE = 10; // this is the max number of elements - need to change it to 50 for client2.

// MAX\_SIZE is unknown to the client

//-------------------------------------------------------

class queue

{

private:

// Data members are:

el\_t el[MAX\_SIZE]; // an array called el.

// Elements will be found between front and rear but

// front may be behind rear since the queue wraps around

int count; // how many elements do we have right now?

int front; // where the front element of the queue is.

int rear; // where the rear element of the queue is.

public: // prototypes to be used by the client

// Note that no parameter variables are given

//add exception handling classes here with comments

class Overflow{}; // when the array is full, you can't push

class Underflow{}; //when the array is empty, and trying to pop

queue(); // constructor to create an object

~queue(); //destructor to destroy an object

// PURPOSE: returns true if queue is empty, otherwise false

bool isEmpty();

// PURPOSE: returns true if queue is full, otherwise false

bool isFull();

// PURPOSE: if full, throws an exception Overflow

// if not full, enters an element at the rear

// PRAMETER: provide the element to be added

void add(el\_t);

// PURPOSE: if empty, throws an exception Underflow

// if not empty, removes the front element to give it back

// PRAMETER: provide a holder for the element removed (pass by ref)

void remove(el\_t&);

// PURPOSE: if empty, throws an exception Underflow

// if not empty, give back the front element (but does not remove it)

// PARAMETER: provide a holder for the element (pass by ref)

void frontElem(el\_t&);

// PURPOSE: returns the current size to make it

// accessible to the client caller

int getSize();

// PURPOSE: display everything in the queue from front to rear

// enclosed in []

// if empty, displays [ empty ]

void displayAll();

// PURPOSE: if empty, throws an exception Underflow

//if queue has just 1 element, does nothing

//if queue has more than 1 element, moves the front one to the rear

void goToBack();

};