//INSTRUCTION:

//Look for \*\* using control-S

//Then, fill in the function body

//When done, compile queue.cpp to make sure it has no syntax errors

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

//HW#: HW1P2 queue

//Your name: Alexander Sadeghipour

//Complier: g++

//File type: \*\* queue.cpp

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

using namespace std;

#include <iostream>

#include "queue.h"

#include <string>

// constructor

queue::queue()

{

front = 0;

count = 0;

rear = -1;

}

//destructor

queue::~queue()

{ // nothing

}

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

bool queue::isEmpty()

{

typedef char el\_t; // el\_t is an alias for char for now

const int MAX\_SIZE = 10; // this is the max number of elements

if (count == 0)

{return true;}

else

{return false;}

}

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

bool queue::isFull()

{ typedef char el\_t; // el\_t is an alias for char for now

const int MAX\_SIZE = 10; // this is the max number of elements

}

// PURPOSE: if full, throws an exception Overflow

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

// PAREMETER: provide the element (newElem) to be added

void queue::add(el\_t newElem)

{

count++;

rear = (rear+1)% MAX\_SIZE; el[rear] = newElem;

}

// PURPOSE: if empty, throw Underflow

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

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

void queue::remove(el\_t& removedElem)

{

removedElem = el[front]; front = (front+1)% MAX\_SIZE; count--;

}

// PURPOSE: if empty, throws an exception Underflow

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

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

void queue::frontElem(el\_t& theElem)

{

rear = (rear+1)% MAX\_SIZE;

}

// PURPOSE: returns the current size to make it

// accessible to the client caller

int queue::getSize()

{

return count;

}

// PURPOSE: display everything in the queue horizontally

// from front to rear enclosed in [ ]

// if empty, displays [ empty ]

void queue::displayAll()

{ if (isEmpty()) { cout << "[ empty ]" << endl;}

else

{

int j = front;

cout << "{";

for (int i = 1; i <= count; i++)

{

cout << el[j] << " ";

j = (j+1)% MAX\_SIZE;

}

cout << "]" << endl;

}

}

// 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 by

//calling remove followed by add.

void queue::goToBack()

{ // \*\* comment a local variable

}