#include <iostream>

using namespace std;

//the queue

template <class T>

class qarr

{

private:

T \* array; //array as a dynamic pointer

int size = 0; //index for the queue

int capacity = 0; //maximum capacity of the queue

public:

qarr(int s){capacity=s; array = new T[capacity];} //capacity set by main and array made in constructor

~qarr(){delete[] array;} //clear memory of the array

void enqueue(); //add value to the back of the queue

void dequeue(); //remove value from the front of the queue

void getSize(){cout << "Size of queue: " << size << endl << endl;}

void getFront(); //output front value

void getRear(); //output last value

void display(); //display the whole queue

};

template <class T>

void qarr<T>::enqueue()

{

//check if the queue is full before adding more values

if (size == capacity)

{

cout << "Queue is full!" << endl;

return;

}

//simply add the value to the next open slot

//it's like waiting in line for food

//when you order, you get the next number

//if the person before you was order 66

//you would get a ticket saying order 67

//this continues and the first person to get their food

//is going to be the first order on the list

T a;

cout << "Enter an item to queue: ";

cin >> a;

cout << endl;

array[size] = a;

size++;

}

template <class T>

void qarr<T>::dequeue()

{

//can't remove from nothing

if (size == 0)

{

cout << "Queue is empty!" << endl;

return;

}

//output what elemennt is being removed

cout << array[0] << " is being dequeued..." << endl;

for (int i = 0; i < size; i++)

{

//move each element forward by making

//each element equal to the one before itself

//which in turn removed the front elemennt from

//the queue

//example:

// < 1 | 2 | 3 | 4 | 5 >

//you would make 1 equal to 2

//make 2 equal to 3 and so on

array[i] = array[i+1];

}

//then you would shorten the size of the queue to end up with this

size--;

//< 2 | 3 | 4 | 5 >

}

template <class T>

void qarr<T>::getFront()

{

if (size == 0)

{

cout << "Queue is empty!" << endl;

return;

}

cout << "Front of queue: " << array[0] << endl << endl;

}

template <class T>

void qarr<T>::getRear()

{

if (size == 0)

{

cout << "Queue is empty!" << endl;

return;

}

cout << "Back of queue: " << array[size-1] << endl << endl;

//size - 1 because arrays start at 0

}

template <class T>

void qarr<T>::display()

{

if (size == 0)

{

cout << "Queue is empty!" << endl;

return;

}

cout << "< ";

for (int i = 0; i < size - 1; i++)

{

cout << array[i] << " | ";

}

cout << array[size-1] << " >" << endl << endl;

}

int main()

{

//Have the user set the capacity of the queue

int s = 0;

cout << "Enter the capacity of the queue: ";

do {

cin >> s;

if (s <= 0) cout << "Capacity must be > 0!" << endl;

} while(s <= 0);

cout << endl;

//Initialize the queue with the given capacity

qarr<string> b(s);

int input;

do {

cout << "-----MENU-----" << endl;

cout << "[0] - Quit." << endl;

cout << "[1] - Enqueue." << endl;

cout << "[2] - Dequeue." << endl;

cout << "[3] - Size." << endl;

cout << "[4] - Front." << endl;

cout << "[5] - Rear." << endl;

cout << "[6] - Display." << endl;

cout << "--------------" << endl;

cout << "Input: ";

cin >> input;

cout << endl;

if (input == 0) break;

if (input == 1) b.enqueue();

if (input == 2) b.dequeue();

if (input == 3) b.getSize();

if (input == 4) b.getFront();

if (input == 5) b.getRear();

if (input == 6) b.display();

} while(true);

}







