Neil Rolf

CSC300

Program 3 – Design

**Class Declarations**

Class StaticQueue

*public variables*

* StaticQueue() – default constructor, sets front of queue pointer to null
* StaticQueue(StaticQueue &) - copy constructor
* ~StaticQueue () – deconstructor
* bool isFull() - returns true if all elements in the queue are in use
* bool isEmpty() - returns true if the queue is empty
* bool enqueue(string, int, string) – adds data to the next available location in the queue, returns true if item was stored successfully
* bool dequeue(string &, int &, string &) - removes the item from the queue and stores its data into reference parameters, returns true if item was removed successfully

*private variables*

* struct PatientInfo – holds data values
  + string name //patient name
  + int age //patient age
  + string problem //description of patient ailments

**Driver**

int(main){

int queueSize = “size of queue”;

StaticQueue testQueueA(queueSize);

nameArray(hardcoded list of names…);

ageArray(hardcoded list of ages…);

problemArray(hardcoded list of problems…);

//function for periodic status checks

void statusCheck(testQueueA);

bool testQueueA.isFull()

bool testQueueA.isEmpty()

print queue contents

//test dequeue on empty queue

bool testQueueA.dequeue(nameArray[i], ageArray[i], problemArray[i]);

void statusCheck(testQueueA);

//test enqueue

for(int i=0, i<”some number”, i++)

bool testQueueA.enqueue(nameArray[i], ageArray[i], problemArray[i]);

void statusCheck(testQueueA);

//test enqueue past queue size

for(int i=0, i<”queueSize + 1”, i++)

bool testQueueA.enqueue(nameArray[i], ageArray[i], problemArray[i]);

void statusCheck(testQueueA);

//test partial dequeue

for(int i=0, i<”some number”, i++)

bool testQueueA.dequeue(nameArray[i], ageArray[i], problemArray[i]);

void statusCheck(testQueueA);

//test copy constructor

StaticQueue testQueueB(testQueueApointer);

void statusCheck(testQueueB);

//test enqueue on copied object

for(int i=0, i<”some number”, i++)

bool testQueueB.enqueue(nameArray[i], ageArray[i], problemArray[i]);

void statusCheck(testQueueA);

void statusCheck(testQueueB);

//test complete dequeue

for(int i=0, i<”remaining items”, i++)

bool testQueueA.dequeue(nameArray[i], ageArray[i], problemArray[i]);

void statusCheck(testQueueA);

}

Tests to run:

* enqueue
* dequeue
* isFull
* isEmpty
* enqueue past given array size
* dequeue on empty queue
* copy a queue using copy constructor
  + test integrity of original queue object