```
/*Michael Amann
CSE 240 Lab 14
Richard Whitehouse
TTh @ 3:30 4:15*/

#include "Queue.h"
#include <string>

/*QueueNode constructor creates a new queueNode and initializes its
members to NULL.*/
queueNode::queueNode()
{
  string data = "";
  queueNode* next = NULL;
}

/*The queueNode member function is used to set the value of
  the node variable “data” to the value of a string input.*/
void queueNode::setData(string input)
{
  data = string;
}

/*The queueNode member function getData returns the
  value of data in a node, provided such a value exists.*/
string queueNode::getData()
{
  if(data != "")

    return data;
}

/*The queueStructure constructor creates a new queueStructure
  and initializes its members to 0 and NULL respectively by calling the
initQueue() function.*/
queueStructure::queueStructure()
{
 initQueue();
}

/*The queueStructure member function initQueue initializes the values
in a queueStructure to 0 and NULL respectively.*/
void queueStructure::initQueue()
{
  count = 0;
  front = NULL;
  end = NULL;
}

/*The queueStructure member function qEmpty is used to determine
whether the queue contains any nodes. If the queue contains at least
one node, the function will return the integer value of 1, 0
otherwise.*/
int queueStructure::qEmpty()
{
  int done = 0;
```

```
  if(end == NULL && front == NULL)
    done = 1;

  return done;
}

/*The queueStructure member function printQueue prints all of the
values contained in the queue's nodes, provided the queue is not
empty.*/
void queueStructure::printQueue()
{
  queueNode* temp = front;

  while(temp != NULL)
    {
      cout << temp->data ;
      temp = temp->next;
    }
}

/*The queueStructure member function enqueue allows string values to be
placed into new nodes that are added to the queue. Once the queue
reaches ten nodes the queue is full.*/
int queueStructure::enqueue(string input)
{//Print the values in the queue.
  int done = 0;

//Provided the queue is not full...
  if(count<10)
    {
/*Create a temporary node called temp. Assign it to the newly created
node. Copy the input string into the node's element "data." Assign the
new node's next pointer to NULL.*/
      queueNode* temp = NULL;
      temp = new queueNode();
      temp->data = input;
      temp->next = NULL;

//If this is the first node in the queue, assign both pointers to it.
      if(front == NULL)
      {
        front = temp;
        end = temp;
      }
      else if(end != NULL)
      {
        end->next = temp;
        end = temp;
      }
/*Increment the queueStructure count to keep track of the number of
nodes in the queue.*/
      count++;
      done = 1;
    }

  return done;
}
```

```
/*The queueStructure member function dequeue removes the nodes from the
queue provided the queue is not empty.*/
int queueStructure::dequeue()
{
  int done = 0;

//If the queue is not empty...
  if(front != NULL && end != NULL)
    {
/*Create a temporary node called temp and assign it to the node that
front points to. Delete what front points to then assign front to temp.
If temp is pointing to NULL such that the queue is now empty, assign
end to NULL as well.*/
      queueNode* temp;
      temp = front->next;
      delete(front);
      front = temp;
      if(temp == NULL)
      end = NULL;

/*Decrement the count to keep track of the number of nodes in the
queue.*/
      count--;
      done = 1;
    }
  return done;
}

/*The queueStructure member function qFull determines if the queue is
full. If the value of the data member count is 10, then the queue is
full and a value of 1 is returned, 0 otherwise.*/
int queueStructure::qFull()
{
  int full = 0;
  if(count == 10)
    full = 1;

  return full;
}

/*The queueStructure member function getFront returns the value
contained in the first element in the queue, provided the queue is
not empty.*/
string queueStructure::getFront()
{
  string data;
  if(front)
    data = front->data;

  return data;
}
```