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

#include<iostream.h>
#include <string>
#include "Stack.h"

/*The stackStructure constructor creates a new stackStructure
  and initializes its members to 0 and NULL respectively.*/
stackStructure :: stackStructure()
{
  count = 0;
  stackStructure* tos = NULL;
}

/*The stackNode constructor creates a new stackNode object
  and initializes its members to empty string and NULL respectively.*/
stackNode::stackNode()
{
  data = "";
  next = NULL;
}

/*The stackNode member function getData returns the value in a
stackNode's data member.*/
string stackNode::getData()
{
  return data;
}

/*The stackNode member function setData allows the programmmer to
assign a string value to the data member of a stackNode.*/
void stackNode::setData(string input)
{
  data = input;
}

/*The stackStructure member function pop removes the first node from
the stack and resets the pointer "top of stack (tos)" to the next
available position on the stack.*/
int stackStructure::pop()
{
  int done = 0;
  stackNode* temp = NULL;

//Provided tos points to some value...
  if(tos != NULL)
    {
/*Assign temp to tos' next then delete what tos points to. Assign tos
to temp.*/
      temp = tos->next;
      delete tos;
      tos = temp;
/*Decrement the count to keep track of the number of nodes on the
stack.*/
      count--;
      done = 1;
    }
  return done;
}

/*The stackStructure member function push places a new node onto the
stackStructure and moves tos to the next available position on the
stack.*/
int stackStructure::push(string input)
{
  int done = 0;

//Provided the stack is not full..
if(count < 10)
{
/*Create a temporary node called temp and copy the string input to its
data member. Point temp's next to tos then point tos to temp.*/
  stackNode* temp = new stackNode();
  temp->data = input;
  temp->next = tos;
  tos= temp;

/*Increment the count to keep track of the number of nodes on the
stack.*/
  count++;
  done = 1;
}

  return done;
}

/*The stackStructure member function stackEmpty determines whether the
stack is empty.If there are no nodes in the stackStructure the stack is
empty and the value of 1 is returned, 0 otherwise.*/
int stackStructure::stackEmpty()
{
  int empty = 0;

  if(tos == NULL)
    empty = 1;

  return empty;
}

/*The stackStructure member function stackFull determines whether the
stack is full. If the value of the stackStructure data member is 10,
then the stack is full and the value of 1 is returned, 0 otherwise.*/
int stackStructure::stackFull()
{
  int full = 0;

  if(count == 10)
    full = 1;
  return full;
}

/*The stackStructure member function topOfStack returns the string
value that tos points to.*/
string stackStructure::topOfStack()
{
  string str;

  if (tos)
    str = tos->data;

  return str;
}

/*The stackStructure member function printStack prints the values of
all the nodes contained in the stack, provided the stack is not
empty.*/
void stackStructure::printStack()
{
  stackNode* temp = tos;

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