**CSC 2201 – Computer Science II**

**Shivang Trivedi**

**Lab #06**

**Due: End of the class**

**40 Points**

1. Save a copy of this document with your name and the assignment number somewhere in the file name. For example, the file name *“CSC2201\_Doe\_Jane\_Lab6.docx”*
2. Copy-and-paste your answers (e.g., C++ source code) into the document.
3. Copy-and-paste the program output window.
4. Submit the following files separately (do not compress the files) to the Canvas in one submission:

1) This document as a word document (i.e., with the extension ***.doc*** or ***.docx***).

2) All C++ source code solution file(s) (only the ***.cpp*** and ***.h*** files) to the Canvas item associated with this assignment/lab solution.

Questions:

- implement the Stack ADT using the array – based approach:

- implement constructor, copy constructor, assignment operator, destructor; (20 pts)

- push, pop; (15 pts)

- clear , isFull, isEmpty; (5 pts)

#include "stackArray.h"

#include <stdexcept>

#include <iostream>

//Name: Shivang Trivedi

//Date: Feb 20, 2019

//Lab: 06 - Stacks Array

using namespace std;

template<typename DataType>

inline StackArray<DataType>::StackArray(int maxNumber)

{

//initialize everything

maxSize = maxNumber;

dataItems = new DataType[maxSize];

top = -1;

}

template<typename DataType>

StackArray<DataType>::StackArray(const StackArray & other)

{

//set this variables to other variables

maxSize = other.maxSize;

top = other.top;

dataItems = new DataType[maxSize];

//deep copy from the other to this dataItems

for (int x = 0; x <= top; x++) {

dataItems[x] = other.dataItems[x];

}

}

template<typename DataType>

StackArray<typename DataType>& StackArray<DataType>::operator=(const StackArray & other)

{

//check if this maxSize is less than other to have enough space for data

if (other.maxSize > maxSize) {

delete[]dataItems;

dataItems = new DataType[other.maxSize];

}

//deep copy from the other to this dataItems

for (int x = 0; x <= top; x++) {

dataItems[x] = other.dataItems[x];

}

//return this

return \*this;

}

template<typename DataType>

StackArray<DataType>::~StackArray()

{

//clear memory

delete []dataItems;

}

template<typename DataType>

void StackArray<DataType>::push(const DataType & newDataItem) throw(logic\_error)

{

//if full throw error

if (isFull()) {

throw logic\_error("List is full\n");

}

//add 1 to top and set the newDataItem to that

top++;

dataItems[top] = newDataItem;

}

template<typename DataType>

DataType StackArray<DataType>::pop() throw(logic\_error)

{

//if empty throw error

if (isEmpty()) {

throw logic\_error("List is empty.");

}

//subtract top by 1 and return the dataitem at top + 1

top--;

return dataItems[top + 1];

}

template<typename DataType>

void StackArray<DataType>::clear()

{

//set top to -1, showStructor wont print numbers after the top elements

top = -1;

}

template<typename DataType>

bool StackArray<DataType>::isEmpty() const

{

//if the top is -1, return true else false

if (top == -1) return true;

else return false;

}

template<typename DataType>

bool StackArray<DataType>::isFull() const

{

//if top is maxsize the list is full

return(top == (maxSize - 1));

}

template <typename DataType>

void StackArray<DataType>::showStructure() const

// Array implementation. Outputs the data items in a stack. If the

// stack is empty, outputs "Empty stack". This operation is intended

// for testing and debugging purposes only.

{

if (isEmpty()) {

cout << "Empty stack." << endl;

}

else {

int j;

cout << "Top = " << top << endl;

for (j = 0; j < maxSize; j++)

cout << j << "\t";

cout << endl;

for (j = 0; j <= top; j++)

{

if (j == top)

{

cout << '[' << dataItems[j] << ']' << "\t"; // Identify top

}

else

{

cout << dataItems[j] << "\t";

}

}

cout << endl;

}

cout << endl;

}





