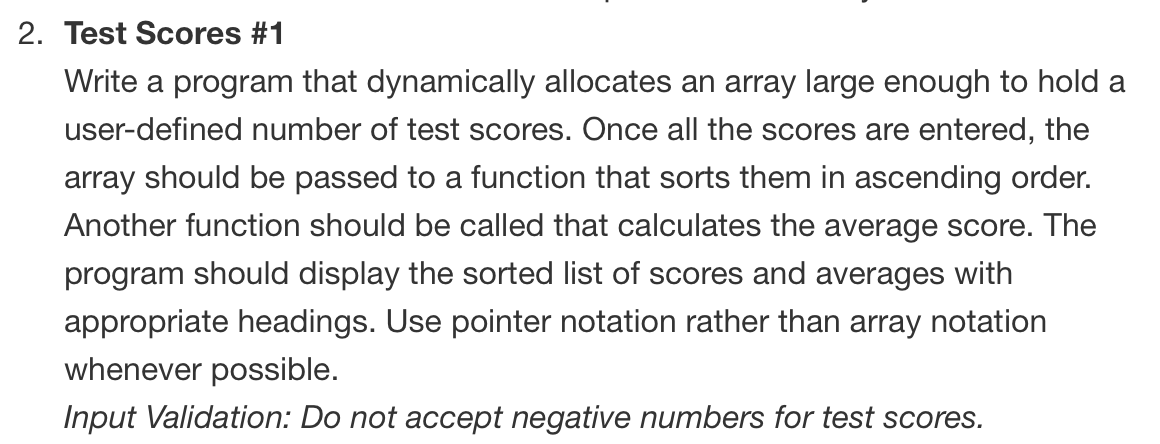
***COMSC165 Lab-2 Due: 7/01/18 11:59 PM***

Follow the lab format under the general folder in Canvas to complete this lab.



#ifndef TESTSCORES\_H

#define TESTSCORES\_H

#include <string>

class TestScores

{

public:

std::string studentName;

int numberOfTests;

double \*testScoresArrayPointer;

double average;

static const int maxTestScore = 100;

static const int minTestScore = 0;

void setStudentData();

int getNumberOfTests();

void createArray(int);

void setTestScores(double []);

void selectionSort(double [], int);

double calculateAverage();

void displayStudentData();

void deleteDynamicArray(double \*arr);

};

#endif // TESTSCORES\_H

#include "TestScores.h"

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

void TestScores::setStudentData(){

numberOfTests = 0;

cout << "Enter the student's name: ";

cin >> studentName;

cout << "\nHow many tests did " << studentName << " take? ";

cin >> numberOfTests;

}

int TestScores::getNumberOfTests(){

return numberOfTests;

}

void TestScores::createArray(int x){

testScoresArrayPointer = new double[x];

}

void TestScores::setTestScores(double arr[]){

for (int i = 0; i < numberOfTests; i++){

cout << "\nScore " << i+1 << ": " ;

cin >> testScoresArrayPointer[i];

while (testScoresArrayPointer[i] > maxTestScore || testScoresArrayPointer[i] < minTestScore){

cout << "\nInvalid score. Test score needs to be between 0 and 100." << endl;

cout << "Re-enter test score: ";

cin >> testScoresArrayPointer[i];

}

}

}

void TestScores::selectionSort(double unsortedArray[], int arraySize){

int minIndex, minValue, start, index;

cout << "\nDebug: Unsorted array: ";

for(index = 0; index < arraySize; index++){

cout << unsortedArray[index] << " ";

}

for(start = 0; start < arraySize - 1; start++){

minIndex = start;

minValue = unsortedArray[start];

for(index = start + 1; index < arraySize; index++){

if(unsortedArray[index] < minValue){

minValue = unsortedArray[index];

minIndex = index;

}

}

swap(unsortedArray[minIndex], unsortedArray[start]);

}

cout << "\nDebug: Sorted array: ";

for(index = 0; index < arraySize; index++){

cout << unsortedArray[index] << " ";

}

}

double TestScores::calculateAverage(){

average = 0;

for (int i = 0; i < numberOfTests; i++){

average += testScoresArrayPointer[i];

}

average = average/numberOfTests;

return average;

}

void TestScores::displayStudentData(){

cout << "\nStudent Name\tAverage Score" << endl;

cout.setf(ios::fixed, ios::floatfield);

cout.precision(2);

cout << studentName<< "\t\t" << setprecision(2) << average << endl;

}

void TestScores::deleteDynamicArray(double \*arr){

delete[] arr;

}

#include <iostream>

#include "TestScores.h"

#include <string>

using namespace std;

int main()

{

TestScores tso;

tso.setStudentData();

tso.createArray(tso.getNumberOfTests());

tso.setTestScores(tso.testScoresArrayPointer);

tso.selectionSort(tso.testScoresArrayPointer, tso.numberOfTests);

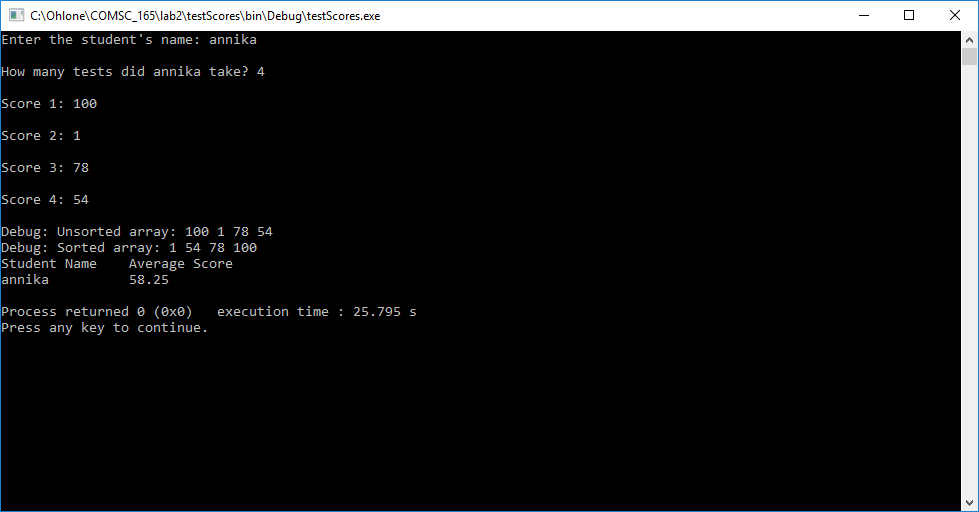
tso.calculateAverage();

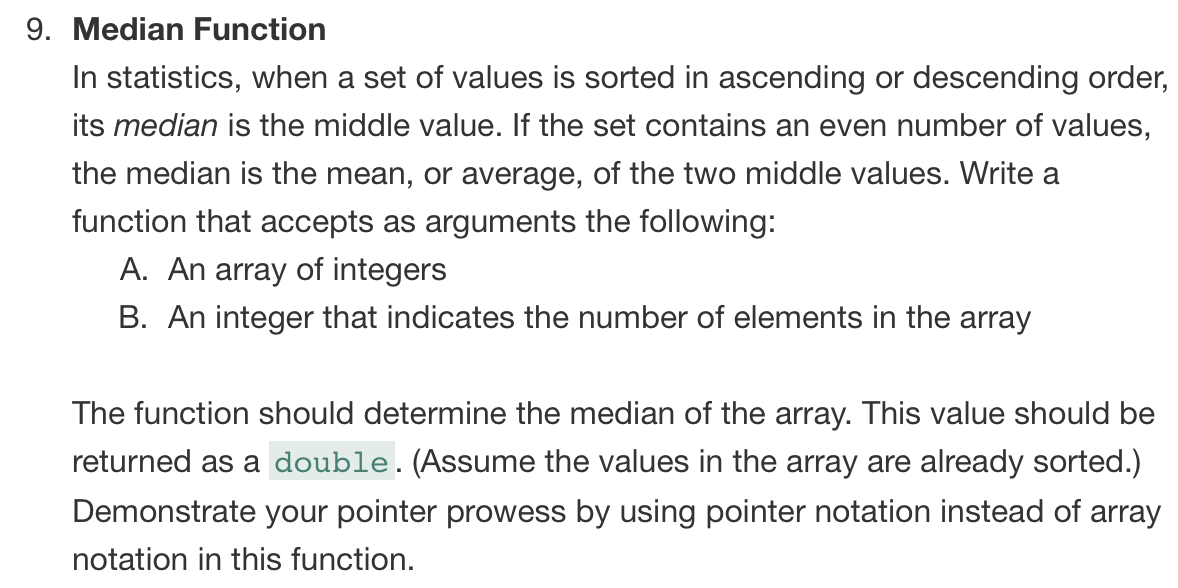
tso.displayStudentData();

tso.deleteDynamicArray(tso.testScoresArrayPointer);

return 0;

}





#ifndef MEDIAN\_H

#define MEDIAN\_H

class Median

{

public:

int arraySize;

double median;

double \*medianArrayPointer;

int getArraySize();

void createArray(int);

void setArrayValues(double \*arr);

void selectionSort(double \*arr, int);

void getMedian(double \*arr, int);

};

#endif // MEDIAN\_H

#include "Median.h"

#include <iostream>

using namespace std;

int Median::getArraySize(){

cout << "Enter an integer for the size of the array: ";

cin >> arraySize;

return arraySize;

}

void Median::createArray(int x){

medianArrayPointer = new double[x];

}

void Median::setArrayValues(double \*arr){

cout << "Enter an integer for each array element.\n";

for(int i = 0; i < arraySize; i++){

cout << "element " << i << ": ";

cin >> arr[i];

}

}

void Median::selectionSort(double \*unsortedArray, int arraySize){

int minIndex, minValue, start, index;

cout << "\nDebug: Unsorted array: ";

for(index = 0; index < arraySize; index++){

cout << unsortedArray[index] << " ";

}

for(start = 0; start < arraySize - 1; start++){

minIndex = start;

minValue = unsortedArray[start];

for(index = start + 1; index < arraySize; index++){

if(unsortedArray[index] < minValue){

minValue = unsortedArray[index];

minIndex = index;

}

}

swap(unsortedArray[minIndex], unsortedArray[start]);

}

cout << "\nDebug: Sorted array: ";

for(index = 0; index < arraySize; index++){

cout << unsortedArray[index] << " ";

}

}

void Median::getMedian(double \*arr, int x){

median = 0;

if(x % 2 == 0){

cout << "\nDebug: " << endl;

cout << \*(arr+(x/2)) << endl;

cout << \*(arr+(x/2+1)) << endl;

median = (\*(arr + (x/2)) + \*(arr + (x/2+1)))/2;

}else{

median = \*(arr + (x/2));

}

cout << "\n\nThe median is: " << median << endl;

}

#include "Median.h"

#include <iostream>

using namespace std;

int main(){

Median mo;

mo.createArray(mo.getArraySize());

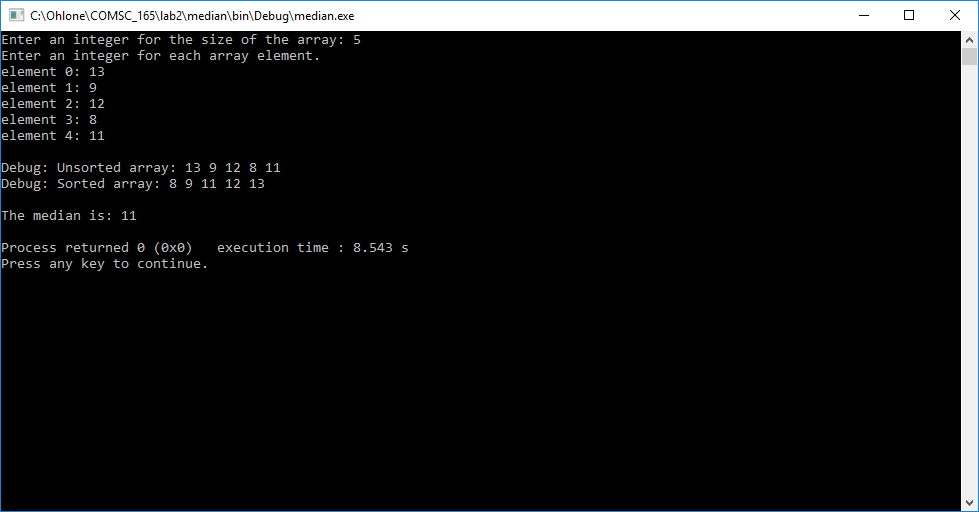
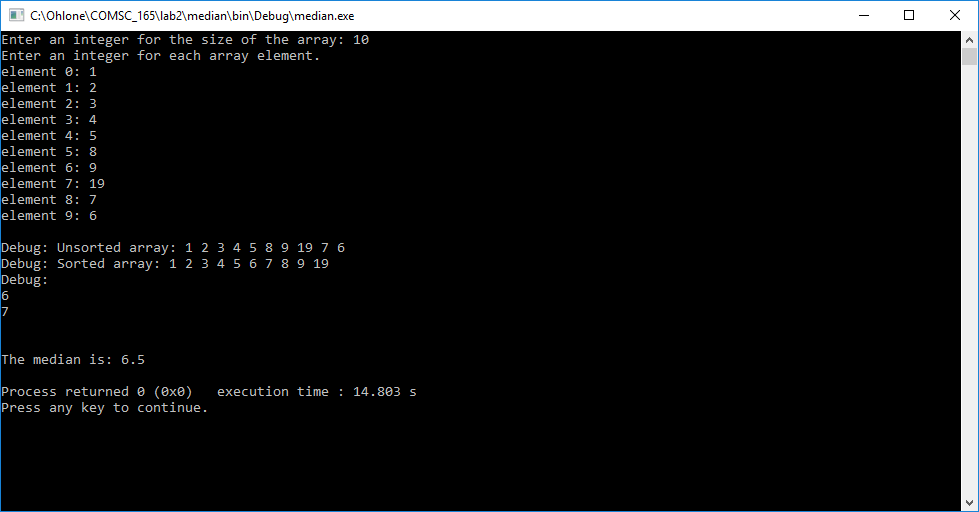
mo.setArrayValues(mo.medianArrayPointer);

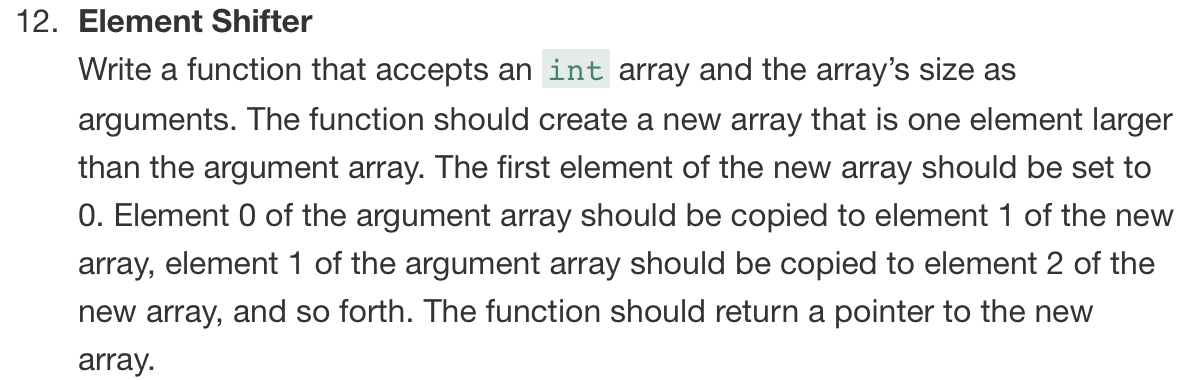
mo.selectionSort(mo.medianArrayPointer, mo.arraySize);

mo.getMedian(mo.medianArrayPointer, mo.arraySize);

return 0;

}





#ifndef ELEMENTSHIFTER\_H

#define ELEMENTSHIFTER\_H

class ElementShifter

{

public:

int arraySize = 10;

int \*baseArray;

int \*elementShifterArrayPointer;

ElementShifter(int);

void setBaseArrayElementValues(int \*arr, int);

int \*shiftArrayElements(int \*arr, int);

void displayShiftedArray(int \*arr, int);

void deleteDynamicArray(int \*arr);

};

#endif // ELEMENTSHIFTER\_H

#include "ElementShifter.h"

#include <iostream>

using namespace std;

ElementShifter::ElementShifter(int arrSize){

cout << "Constructor." << endl;

baseArray = new int[arrSize];

}

void ElementShifter::setBaseArrayElementValues(int \*arr, int arrSize){

for(int i = 0; i < arrSize; i++){

arr[i] = i+2;

}

cout << "\nBase Array Contents: \n";

for(int i = 0; i < arrSize; i++){

cout << "Element " << i << ": " << arr[i] << endl;

}

}

int \*ElementShifter::shiftArrayElements(int \*arr, int arrSize){

elementShifterArrayPointer = new int[arrSize+1];

elementShifterArrayPointer[0] = 0;

int j = 1;

for(int i = 0; i < arrSize; i++){

elementShifterArrayPointer[j] = arr[i];

j++;

}

cout << "\nDebug new/dynamic array 0th element: " << elementShifterArrayPointer[0];

return elementShifterArrayPointer;

}

void ElementShifter::displayShiftedArray(int \*arr, int arrSize){

cout << "\nShifted Array Contents: \n";

for(int i = 0; i < arrSize; i++){

cout << "Element: " << arr[i] << endl;

}

}

void ElementShifter::deleteDynamicArray(int \*arr){

delete[] arr;

}

#include "ElementShifter.h"

#include <iostream>

using namespace std;

int main(){

ElementShifter eso(eso.arraySize);

eso.setBaseArrayElementValues(eso.baseArray, eso.arraySize);

eso.shiftArrayElements(eso.baseArray, eso.arraySize);

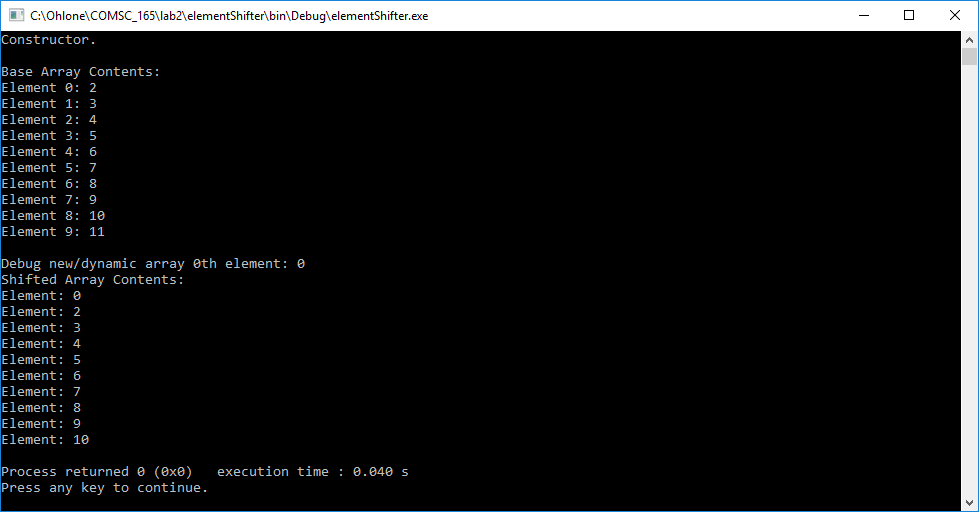
eso.displayShiftedArray(eso.elementShifterArrayPointer, eso.arraySize);

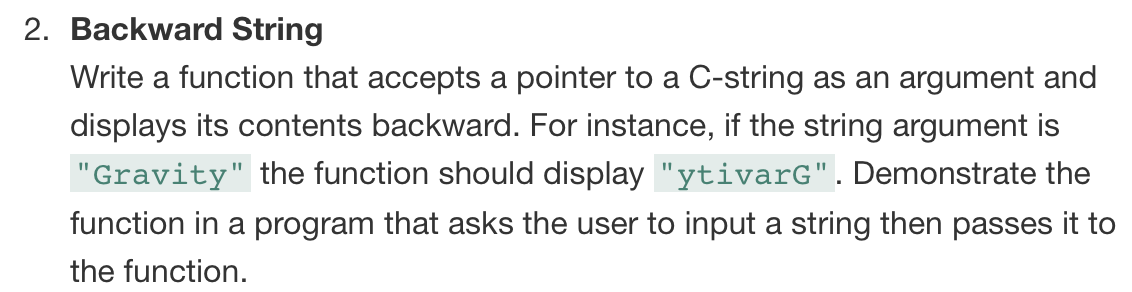
eso.deleteDynamicArray(eso.baseArray);

eso.deleteDynamicArray(eso.elementShifterArrayPointer);

return 0;

}





#ifndef BACKWARDSTRING\_H

#define BACKWARDSTRING\_H

class BackwardString

{

public:

int SIZE = 80;

int minIndex = 0;

char backwardString[];

BackwardString();

void getUserInputAndSetPointerToString();

void displayBackwardString(char \*);

};

#endif // BACKWARDSTRING\_H

#include "BackwardString.h"

#include <iostream>

using namespace std;

BackwardString::BackwardString(){

backwardString[SIZE];

}

void BackwardString::getUserInput(){

cout << "Please enter a string: " << endl;

cin.getline(backwardString, SIZE);

cout << "\n\ndebug: " << backwardString << "\n\n";

}

void BackwardString::displayBackwardString(char \*arr){

char \*inverse = arr;

while(\*inverse != '\0'){

inverse++;

}

cout << "The reverse string is: ";

while(inverse != arr){

inverse--;

cout << \*inverse;

}

}

#include "BackwardString.h"

#include <iostream>

using namespace std;

int main(){

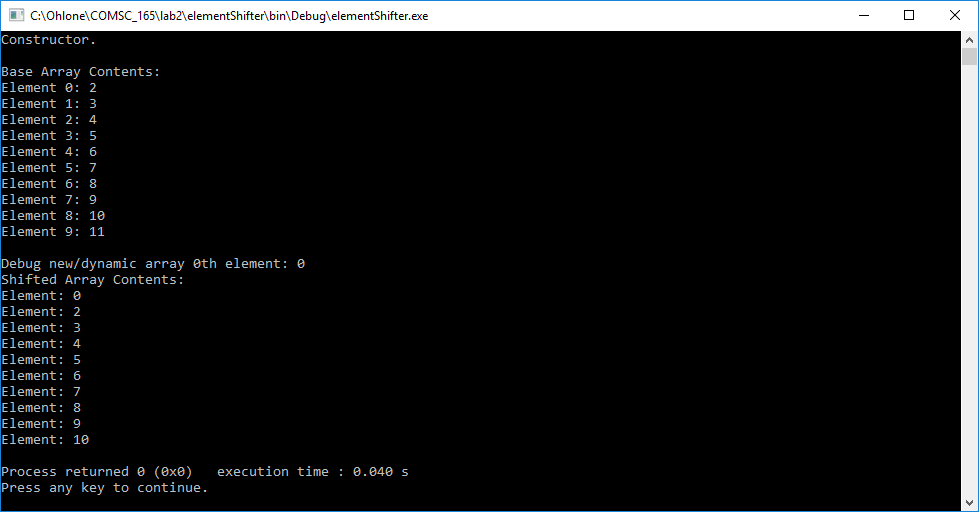
BackwardString bso;

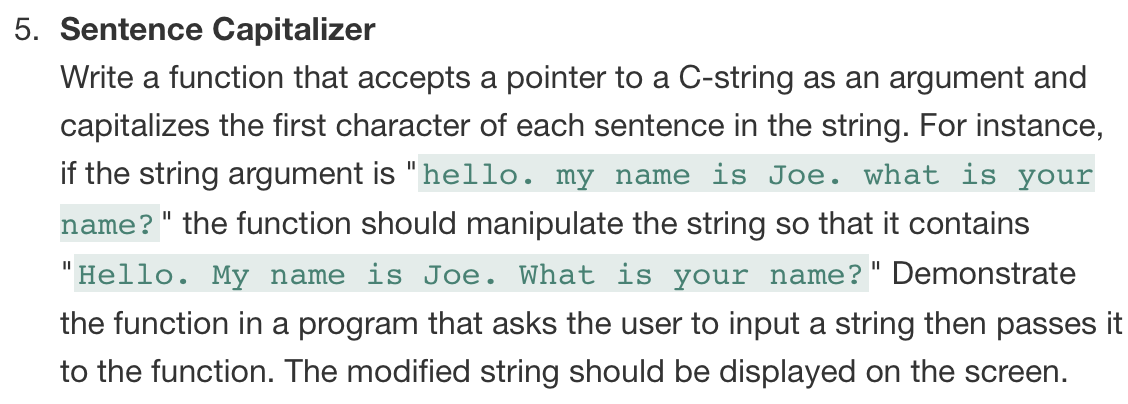
bso.getUserInput();

bso.displayBackwardString(bso.backwardString);

return 0;

}





#ifndef SENTENCECAPITALIZER\_H

#define SENTENCECAPITALIZER\_H

class SentenceCapitalizer

{

public:

int SIZE = 80;

int minIndex = 0;

char unconvertedSentence[];

SentenceCapitalizer();

void getUserInput();

void displayAndConvertSentence(char \*);

};

#endif // SENTENCECAPITALIZER\_H

#include "SentenceCapitalizer.h"

#include <iostream>

#include <cctype>

using namespace std;

SentenceCapitalizer::SentenceCapitalizer(){

unconvertedSentence[SIZE];

}

void SentenceCapitalizer::getUserInput(){

cout << "Please enter a string: " << endl;

cin.getline(unconvertedSentence, SIZE);

cout << "\ndebug: " << unconvertedSentence << "\n";

}

void SentenceCapitalizer::displayAndConvertSentence(char \*arr){

cout << arr << endl;

if(arr[0] != ' '){

arr[0] = toupper(arr[0]);

}

for(int i = 0; i < SIZE; i++){

if(arr[i] == '.'){

if(arr[i+1] != ' '){

arr[i+1] = toupper(arr[i+1]);

}else if(arr[i+2] != ' '){

arr[i+2] = toupper(arr[i+2]);

}

}

}

cout << "\nCapitalized Sentence:" << endl;

cout << arr << endl;

}

#include "SentenceCapitalizer.h"

#include <iostream>

#include <cctype>

using namespace std;

int main(){

SentenceCapitalizer sco;

sco.getUserInput();

sco.displayAndConvertSentence(sco.unconvertedSentence);

return 0;

}

